International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

An Improved Routing based Capsule Network for Hyperspectral Image Classification

B. Thiyagarajan[1], M. Thenmozhi[2], K. Revathy[3]

Submitted: 18/09/2023 **Revised**: 18/11/2023 **Accepted**: 28/11/2023

Abstract: Capsule networks have emerged as a solution to the limitations faced by convolutional neural networks. This innovative architecture focuses on encoding features and capturing spatial relationships within images. Instead of employing max pooling, capsule networks introduce a dynamic routing process. The Capsule network is trained to classify each pixel in a hyperspectral image to predefined categories with suitable loss functions and techniques for optimization. By effectively modeling the complex information embodied in hyperspectral data, capsule networks have the potential to improve the accuracy of hyperspectral image classification, making them an invaluable tool for applications such as remote sensing that rely significantly on spectral information. However, the original three-layer capsule network with dynamic routing exhibits subpar performance on intricate datasets like CIFAR-10/100, SVHN and PaviaU HIS dataset, primarily due to the computationally intensive nature of the dynamic routing algorithm. To tackle these challenges, an enhanced capsule network has been proposed, integrating a dense block layer and an "Improved Routing" algorithm. This improved capsule network configuration has undergone testing on CIFAR-10 and SVHN datasets, resulting in notable enhancements such as improved accuracy, reduced loss rates, and decreased time complexity.

Keywords: *Capsule network, Routing algorithm, Improved Routing algorithm, image classification, Convolution Neural Network, Deep Learning.*

1. Introduction

The widespread application of deep learning in computer vision has been driven by the immense volume of generated images. Within this domain, a multitude of tasks span various domains, including but not limited to image classification [1]–[3], scene recognition [4], object detection [6], image segmentation [7], and a host of other endeavors.

The hyperspectral imaging spectrometer is used to produce hyperspectral images (HSIs).Numerous practical applications, include military recognition of targets, discovering minerals, and agricultural production, may profit from the high level of sensitivity of the HSI, as each pixel comprises hundreds of elements of reflected information at various frequencies[8].

Hyperspectral imaging is becoming more and more essential in remote sensing owing to the advancement of the technology and the distinctive features of HSI data. Furthermore, HSI classification—which involves allocating a class label to each pixel—is emerging one of the primary subjects of HSI research [9].Conversely, it is challenging to extract discriminative information from HSI for classification given complicated noise effects and spectrum variability [10], high dimensionality, inadequately labeled training samples ,and major spectral mixing of materials

Convolutional neural networks (CNNs) are among the most crucial structures for accomplishing these tasks. Starting with the inaugural deep learning network model, LeNet, researchers have since crafted a multitude of additional CNN architectures, each building upon the last. [10], and new versions continue to emerge for performing novel visual tasks. These advancements involve the conception, design, oversight, and investigation of these models.

In a typical CNN, pooling can significantly reduce computational complexity, but it comes at a cost as it retains only the maximum or average value of each pixel, causing the network to lose precise location information of the target.

Moreover, the network exclusively learns to identify the target within the input image and isn't trained to emphasize the exact positional details of the target.

While CNNs have proven effective in accomplishing various image tasks, convolution, as the fundamental architecture of deep learning image processing, still requires improvement. When presented with different angled samples of the same object, humans form coordinate systems to recognize the images and remember the patterns they have learned. In contrast, For Convolutional Neural Networks (CNNs) to proficiently

[1]*Research Scholar, Department of Computer Science and Engineering, Puducherry Technological University, Puducherry – 605014, India.*
[2]*Associate Professor, Department of Computer Science and Engineering, Puducherry Technological University, Puducherry – 605014, India.*
[3]*PG Student, Department of Computer Science and Engineering, Puducherry Technological University, Puducherry – 605014, India.*
**Corresponding Author Email: thiyagarajan.b@pec.edu*

identify such variations, they need to learn new parameters.

Although the dynamic routing approach for processing capsule information has certain drawbacks in terms of training duration and efficiency, particularly when confronted with a substantial quantity of capsules, it shares similarities with max-pooling, a technique employed for feature map handling.[1] This similarity demonstrates that the routing method can effectively transfer low-level capsule information. A unique routing technique called 'self-routing' [13] avoids iterative approaches by using an additional matrix. It also applies the convolution sliding window technique to the input capsule, greatly accelerating routing. However, it processes certain information repeatedly while ignoring some edge capsule data.

On traditional simple datasets like MNIST, the original CapsNet works admirably [14]. However, the network's convolution layer only employs two 9x9 convolution kernels, leading to insufficient feature extraction. Additionally, a large number of capsules constitute a background data entity, which can deceive routing results and slow down dynamic routing.

A new arrangement featuring a trio of dense block layers is suggested by the multi-level dense CapsNet.[15]. In comparison to ResNet, it can extract richer features from complex datasets and reduce the number of parameters by linking the blocks of each layer [3].

The original CapsNet had the following drawbacks: Two central issues require attention. Firstly, the capsule's performance is compromised by the features extracted from the shallow convolutional network. Secondly, the routing technique encounters challenges in achieving convergence due to the iterative calculation of the coupling coefficient, prompting a reduction in the artificially specified number of iterations. Elevating the processing efficiency of primary capsules emerges as a critical and essential avenue for CapsNet improvement.

While capsule networks (CapsNets) offer promising solutions to address the limitations of traditional convolutional neural networks (CNNs), they also face their own challenges and limitations. One limitation is the additional computational expense introduced by the iterations involved in the routing process. The iterative nature of dynamic routing increases the computational cost compared to the feed-forward process of CNNs. This can result in longer training and inference times, making CapsNets less efficient for real-time applications or large-scale datasets [33].

Another limitation is the sparsity of capsule layers in CapsNets.Within the traditional CapsNet structure, low-level capsules are restricted to selecting a sole high-level capsule for voting, while ignoring the rest..This can lead to an uneven distribution of capsules, with some high-level capsules being highly activated while others remain dormant.

Furthermore, CapsNets may struggle to achieve optimal performance on datasets containing more complex items. While CapsNets have shown good performance on simple datasets like MNIST, they may face challenges with more intricate and diverse datasets. This is because traditional CapsNets have limited feature extraction capabilities, and adding more capsule layers to improve feature extraction can further increase computational costs and memory usage due to the complexity of matrix multiplications [31].

To address these limitations, researchers have proposed various improvements and extensions to CapsNets. For example, there have been efforts to optimize the routing algorithm to reduce computational costs and improve scalability. Additionally, architectures such as dynamic routing with spatially transformed capsules (DynamicRoutings), matrix capsules with EM routing (Matrix Capsules), and others have been introduced to enhance the feature extraction capabilities and performance of CapsNets on complex datasets [1] [31]. It's worth noting that the field of CapsNets is still evolving, and ongoing research aims to overcome these limitations and further refine the architecture for improved performance and efficiency.

This article centers around augmenting the classification accuracy of CapsNet while maintaining optimal time utilization. To achieve this goal, we must surmount three technical hurdles outlined as follows:

- Mitigating the iteration load of the Dynamic Routing (DR) algorithm and proficiently harnessing the insights embedded in low-level capsules.
- Integrating the attributes of capsule vectors into the routing algorithm to expedite model training and obtain favorable performance.
- Facilitating feature reutilization when dense blocks are employed as feature extractors within CapsNets.

Considering these challenges, our article offers significant contributions:

We introduce a novel routing algorithm called "swift routing," which functions as a substitute for the DR algorithm. This innovative strategy notably curtails execution time by 71.2% on the MNIST dataset, while marginally enhancing classification accuracy in contrast to conventional CapsNets. Furthermore, the deployment of CapsNet with the swift routing mechanism showcases its resilience when facing affine distortions.

Dense blocks are incorporated prior to capsule layers to serve as feature extractors. This choice is based on the superior computational efficiency and feature extraction

capabilities of convolutional layers within dense blocks compared to capsule layers.

In summary, our research aims to enhance the classification accuracy of CapsNet while maintaining favorable time efficiency. Our notable contributions involve introducing the improved routing algorithm and employing dense blocks as efficient feature extractors within CapsNets.

2. Related Literature

Capsule networks have been introduced as a potential alternative to traditional convolutional neural networks, as they offer a better understanding of the hierarchical relationships between different features and entities. Capsules in different layers work together to represent specific features or entities, and they are connected to each other through routing algorithms that determine their similarity and probability of connection [1]. Dynamic routing is a popular routing algorithm used in capsule networks. It uses matrix multiplication with transformation matrices to predict the similarity between capsules in different layers. The utilization of cosine similarity facilitates the assessment of similarity between the primary capsule layer and the digit capsule layer. While this algorithm offers precise predictions and enhances the comprehension of hierarchical relationships among capsules, it may entail a significant computational cost [2].

EM routing is another routing algorithm used in capsule networks that is more computationally efficient than dynamic routing. It comprises a duo of stages: the M-step and the E-step. Activation of the high-level capsule transpires when the variance between the low-level capsule and the high-level capsule is minimal, while it remains inactive when the variance is elevated. [3]. Several studies have proposed modifications and improvements to capsule networks for specific applications. For instance, a modified capsule network has been used to diagnose COVID-19 from medical imaging [4].

Another study proposed capsule filter routing between the primary capsule layer and the dense block to improve the network's performance [5]. Another modification to capsule networks is the addition of attention mechanisms to improve their interpretability and robustness. Attention mechanisms allow the network to focus on specific regions of input and assign different weights to different features. This can improve the network's ability to distinguish between different objects or classes [6]. In these applications, capsules represent different aspects of language, such as grammar, syntax, and semantics, and are connected to each other through routing algorithms to capture the relationships between different aspects of language [7].

Some researchers have delved into amalgamating capsule networks with other categories of neural networks,, such as recurrent neural networks (RNNs) and attention networks, to improve their performance on specific tasks. For instance, a capsule-based RNN has been used for time series prediction [8], while a capsule-based attention network has been used for document classification [9].

One recent paper that explores the potential of capsule networks for a specific application is "Improved capsule routing for weakly labelled sound event detection." In this study, parallel convolution layers were used for feature extraction, and the output was fed into two capsule layers. Subsequently, a recurrent layer was introduced following the capsule layer to grasp temporal context information. This recurrent layer, in tandem with a fully connected layer, was leveraged to compute event activity probabilities and acquire understanding of temporal context details.[10]. These modifications and improvements to capsule networks highlight their versatility and potential for a wide range of applications. Through enabling a more comprehensive comprehension of the hierarchical connections among distinct features and entities, capsule networks have the potential to enhance the precision and interpretability of neural networks across a wide array of domains, spanning from medical imaging to natural language processing. Moreover, by combining capsule networks with other types of neural networks, researchers can create even more powerful models for specific tasks, further expanding the range of applications for these innovative networks.

In conclusion, capsule networks offer a promising alternative to traditional neural networks, as they provide a better understanding of the hierarchical relationships between different features and entities. Dynamic routing and EM routing are two popular routing algorithms used in capsule networks, with EM routing being more computationally efficient. Finally, combining capsule networks with other types of neural networks, such as RNNs and attention networks, has shown improved performance on specific tasks. With ongoing research and advancements, capsule networks are poised to play an important role in the future of deep learning.

CapsNet has numerous uses in various areas [38][40]. Examples include few-shot learning [41], unsupervised learning [42], and GANs [43]. In a study by [44], better dynamic routing was employed for legal judgment in charge prediction. Yang et al. [45] utilized CapsNet to aid in the extraction of hierarchical graph features using

3. Proposed Methodology

Consider an input image x0 with dimensions 32x32x3. The network consists of four layers. The initial layer is a convolutional one, characterized by a filter size of 3, a

kernel size of 3, a stride of 1, and padding of 1. This specific layer generates an output featuring 32 channels.

The second layer is an Improved Dense Block layer, which contains three blocks. Each block consists of three layers, and each layer performs two convolutional operations. In the first convolution, the kernel size is 1, the stride is 1, and the number of output channels is calculated by multiplying bn_sizewithgrowth_rate. The subsequent convolution employs a kernel size of 3, a stride of 1, and padding of 1. The output of each layer in the dense block is calculated using the following formula (Eq. 1).

$$k*(\beta /(n*l)) \quad \ldots\ldots\ldots \quad (1)$$

Where k is the input feature maps, β is hyper parameter (it should be selected in the range from 2 to 6), n is number of layers, l is current layer and also constant values are for bn_size(hyperparameter)=4, growth rate = 32. After each dense block, there is a transition layer, which includes a convolutional layer and an average pooling layer.

The number of output features from the DenseNet is 462.Subsequently, the output of the DenseNet is fed into the T_caps layer, featuring a kernel size of 3, a stride of 2,no padding. and a caps_size of 9 (the output channel should be multiples of caps_size). The resultant of this layer has 32x9 channels. The output of the T_caps layer is then given to the high-level capsule layer, which has a kernel size of 3, a caps_in_size of 9, and a caps_out_size of 9. The resultant of this layer has ten channels, representing the ten classes of the output.

The high-level capsule layer refers to a specific layer in a neural network architecture that utilizes capsule networks. Capsule networks are designed to overcome some limitations of traditional convolutional neural networks (CNNs) in capturing spatial hierarchies and handling viewpoint variations.

In a high-level capsule layer, the outputs of lower-level capsules are grouped together to form higher-level capsules. Each capsule represents a specific

3.1. Proposed Routing Algorithm

3.1.1 Dynamic Routing

The Dynamic Routing (DR) mechanism in CapsNet aims to dynamically generate a sparse tree-like structure, imposing a constraint on the weights of capsules. During iterative iterations, capsules in a layer allocate their votes to a singular parent capsule in the subsequent layer, excluding others. This procedure is known as "routing." However, this strategy has drawbacks, including potential information loss as the routing algorithm may prioritize specific capsules over others. For example, when considering hand capsules, the algorithm might favor finger capsules while overlooking toe capsules. Moreover, the routing process imposes a significant computational burden due to its iterative nature.

To address these routing concerns, our goal is to create an innovative algorithm that reduces information loss and minimizes the computational load tied to iterative procedures.

3.1.2 Improved Routing

Precisely, the total of votes cast by a low-level capsule towards high-level capsules is required to equate to 1. To attain this, the softmax function is employed, enabling a low-level capsule to primarily allocate its vote to the high-level capsule with the most significant resemblance.

The concept of utilizing variance as a measure to evaluate the consistency among predictions is easy to grasp. When the predictions from lower-level capsules are tightly grouped, it implies a consensus in their connection with a specific higher-level capsule. In contrast, higher variance could signify greater ambiguity or reduced consensus among the predictions. This approach provides a straightforward way to gauge the level of agreement within a system of capsules, offering insights into their overall reliability and cohesion.

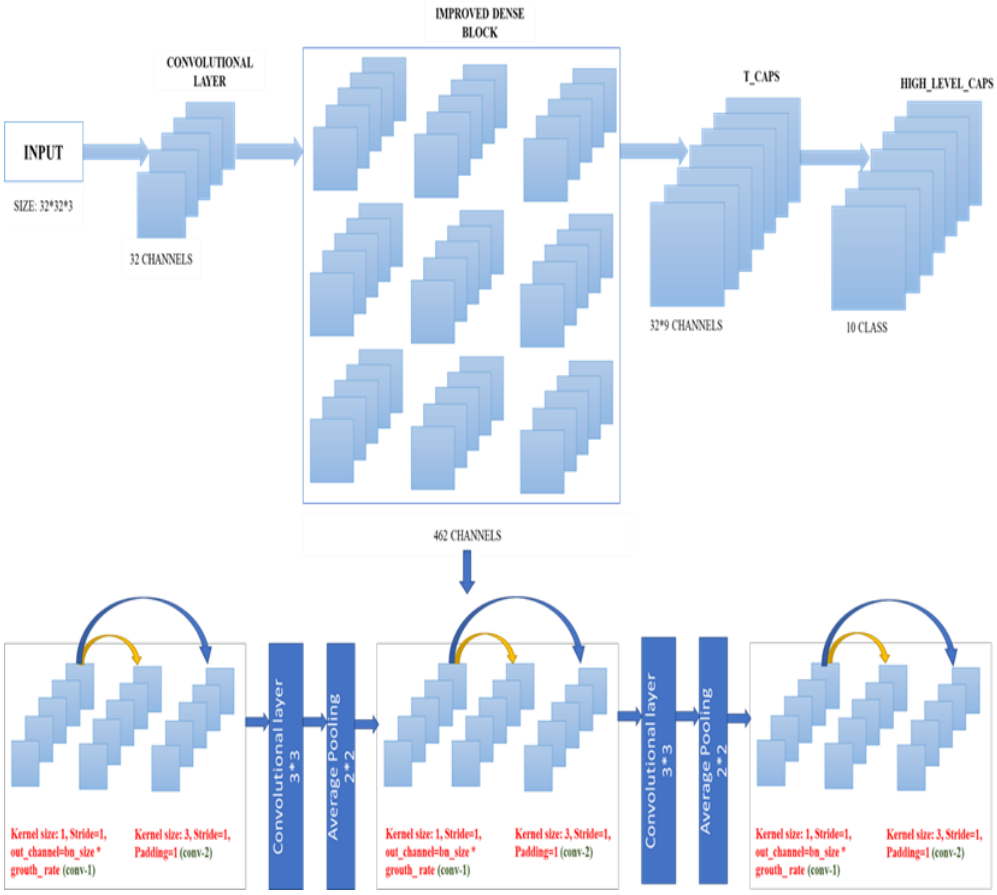

Fig. 1. Proposed Capsule Network Architecture

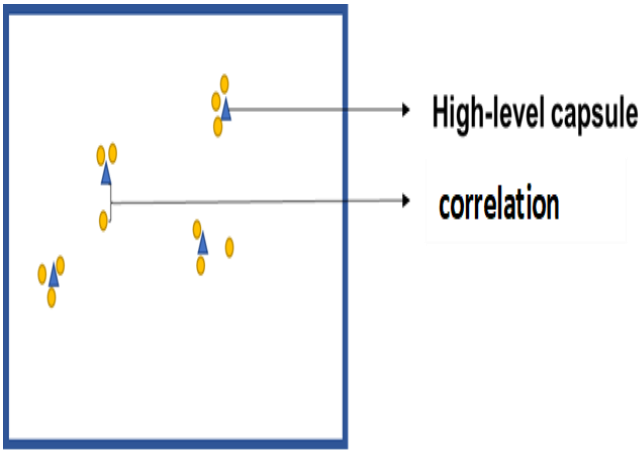

Fig. 2. Capsule Network Activation Filter

3.1.3 Improved Routing Algorithm

According to Improved Routing, the variance of votes is used to determine the activation of high-level capsules in the proposed "Improved Routing" method. Thus, the sum of votes and the distribution of votes are considered simultaneously.

Algorithm 1 Improved Routing

Input: u_i

Output: v_j

1. Require:c_i,p_i, $\Phi_{i,j}$
2. $pri,_j,\Phi_{i,j}c_i$
3. for r iteration, do
4. $meanj = \sum_i p_i pr_{i,j} / \sum_i p_i$
5. $std\_dev_j = \sum_i,(p_{i,j} - mean_j)2 \ / \sum_i,p_i$
6. $correlation = \sum i,(p_{i,j} - mean_j)/std\_dev_j$
7. $p_j = correlation_j$
8. $c_j = mean_j$
9. Return: c_j,p_j

3.2 Components of Improved Routing

Input: The algorithm takes the child capsule vectors (c_i) from the previous layer and the transformation matrix ($\Phi_{i,j}$) that provides spatial information about the features. The transformation matrix is a 4x4 matrix with 16 values representing features such as position, size, orientation, colour, and texture. These values act as weights for the low-level capsules in relation to the high-level capsules.

Computation of $pr_{i,j}$: The algorithm calculates $pr_{i,j}$by multiplying the child capsule vectors (c_i) with the corresponding transformation matrix ($\Phi_{i,j}$). This step combines the information from the child capsules with the spatial information provided by the transformation matrix.

Iterative computation of mean, standard deviation, and correlation: The algorithm iteratively computes the mean (mean), standard deviation (std_dev_j), and correlation (correlation) for each high-level capsule. The mean is calculated as the sum of the products of $pr_{i,j}$and the high-level capsule values, divided by the sum of $pr_{i,j}$. The standard deviation is calculated based on the deviation of each $pr_{i,j}$from the mean, and the correlation is the ratio of the deviation to the standard deviation.

Calculation of p_j and c_j: The algorithm sets the values of the high-level capsule (p_j) as the calculated correlation (correlation) and the mean (c_j) as the calculated mean (mean). These values represent the direction and degree of the relationship between the high-level capsule and the child capsules.

Output: The algorithm returns the values of cj and pj for each high-level capsule. The final layer is the Parent Capsule Layer, which is activated if the predictions of the child capsule cluster tightly in the high-dimensional space of the parent capsule. For example, if the features present in the child capsules match the features of a cat image, then the cat image in the parent capsule will be activated.

Additionally, the output of the DenseNet is fed into the T_caps layer, also known as the Primary layer. This layer uses a kernel size of 3, a stride of 2, padding of 0, and a caps_size of 9 (the output channel should be multiples of caps_size). The T_caps layer outputs 32x9 channels. The output of this layer is then fed into the high-level capsule layer, which uses a kernel size of 3, a caps_in_size of 9, and a caps_out_size of 9. The resulting output of this layer has ten channels, representing the ten classes of the output.

3.3 Advantages of the Improved Routing

Improved Routing: The algorithm introduces a new routing process that utilizes correlation instead of variance used in existing routing algorithms. Correlation measures the strength of the relationship between capsules, providing a more accurate representation of the capsule clusters.

Reduced Loss and Time Efficiency: By using correlation and the iterative process, the algorithm aims to reduce loss and improve time efficiency compared to traditional routing algorithms. This is achieved by better capturing the relationships and dependencies between capsules, leading to more effective clustering.

Improvised Dense Block: The algorithm proposes an improved dense block that addresses the issue of redundant features in traditional DenseNet. By utilizing Eq. (1), the algorithm removes redundant features, leading to a more efficient and compact representation of features.

The new Routing algorithm has been proposed and tested on PaviaU, CIFAR-10 and SVHN datasets. In this Routing algorithm, we have used correlation and iteration processes by comparison with the Existing Routing algorithm. In existing routing algorithms, they have used the variance to form the cluster [3].

The variance measures the spread between the capsules. But the correlation measures the strength of the relationship between the capsules. By this algorithm, we have reduced the Loss and Time efficiency on the datasets PaviaU, CIFAR-10/100 ,FMNIST and SVHN. An Improvised Dense block has been proposed to reduce the redundant features, which is the main drawback of traditional denseness.

Table 1. Comparison of Algorithms for Time Efficiency

Datasets	Dynamic Routing	EM Routing	Fast Routing	Improved Routing
CIFAR-10	**2.85**	**3.15**	**3.00**	**2.73**
SVHN	**3.79**	**3.32**	**3.44**	**2.32**
CIFAR-100	**2.85**	**3.15**	**3.00**	**2.73**
PaviaU	**3.55**	**3.85**	**3.47**	**2.64**

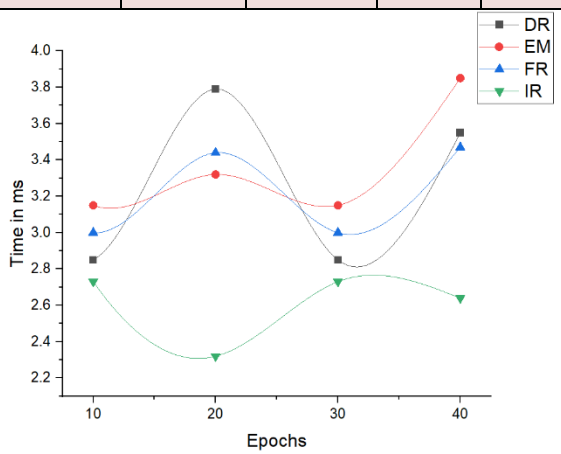

Fig. 3. Comparison of Algorithms for Time Efficiency

When comparing time efficiency on the CIFAR-10 dataset, the Improved Routing algorithm exhibits increased computation time compared to the Fast Routing algorithm. However, on the PaviaU and SVHN datasets, the Improved Routing algorithm demonstrates less computation time.

Table 2. Accuracy of Algorithms

Datasets	Dynami Routing	EM Routing	Fast Routing	Improved Routing
CIFAR-10	0.69	0.46	0.91	0.97
SVHN	0.83	0.55	0.95	0.95
CIFAR-100	0.69	0.46	0.91	0.94
PaviaU	0.72	0.52	0.86	0.93

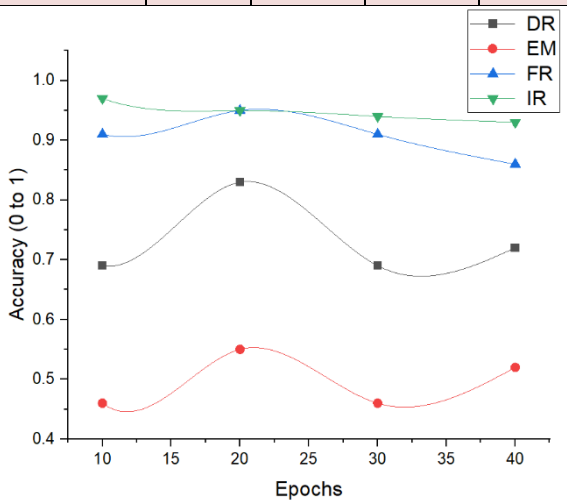

Fig. 4. Accuracy of Algorithms

By comparing the accuracy on the CIFAR-10/100 dataset, the Improved Routing algorithm achieves a higher accuracy, increasing by 0.2 percentage points compared to the Fast Routing algorithm. Similarly, on the PaviaU, SVHN datasets, the Improved Routing algorithm demonstrates improved accuracy, increasing by 0.1 percentage points compared to the Fast Routing algorithm.

By comparing the error rate on both the PaviaU, CIFAR-10/100 , FMNIST and SVHN datasets, the Improved Routing algorithm yields the same results as the Fast Routing algorithm. There is no significant difference in the error rates between the two algorithms for these datasets.

Table 3 . Loss

.Datasets	Dynamic Routing	EM Routing	Fast Routing	Improve Routing
CIFAR-1	7.51	5.12	1.23	0.47
SVHN	6.36	4.95	0.41	0.05
CIFAR - 100	7.50	5.09	1.11	0.49
Pavia U	5.37	5.02	1.05	0.09

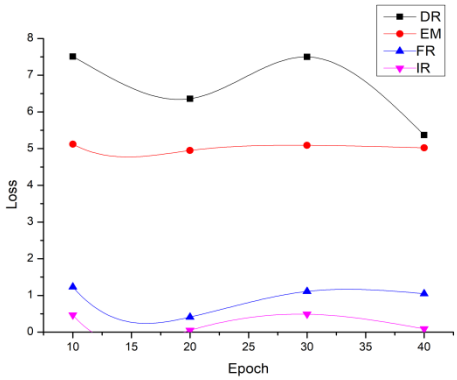

Fig. 5 . Loss Rate

By comparing the loss on the CIFAR-10 dataset, the Improved Routing algorithm shows improved results, reducing the loss by a difference of 0.76 compared to the Fast Routing algorithm. Similarly, on the PaviaU and SVHN datasets, the Improved Routing algorithm demonstrates improved results, reducing the loss by a difference of 0.36 compared to the Fast Routing algorithm.

3.4 Comparison results on Improvised Dense Block by varying Bn_size value

In the evaluation of the Improvised Dense Block, we assessed the impact of varying the Bn_size value on the performance. The Bn_size parameter determines the number of channels in the bottleneck layer of the block. By altering this value, we can adjust the capacity and complexity of the block.

Our goal was to examine how different Bn_size values affect the accuracy, error rate, loss, and number of parameters in the model. By analyzing these metrics, we can understand the trade-offs and determine the optimal Bn_size value for the specific task.

We conducted the evaluation on the PaviaU, CIFAR-10 and SVHN datasets using the proposed Improved Routing algorithm. We compared the performance of the model

across different Bn_size values, including 2, 4, and 6. This range of values allowed us to assess the impact of increasing Bn_size on the model's performance.

Now, let's explore the comparison results obtained by varying the Bn_size value in the Improvised Dense Block.

Table 4. Time Efficiency of Learning Parameters

Datasets	Bn_size=2	Bn_size=4	Bn_size=6
CIFAR-10	5.35	3.00	2.71
SVHN	3.48	2.90	2.34
CIFAR - 100	5.38	3.07	2.74
PaviaU	2.78	2.56	2.62

Table 5. Accuracy

Datasets	Bn_size=2	Bn_size=4	Bn_size=6
CIFAR-10	0.91	0.91	0.97
SVHN	0.95	0.95	0.95
CIFAR-100	0.87	0.87	0.94
PaviaU	0.95	0.95	0.93

Table 6. Error Rate

Datasets	Bn_size=2	Bn_size=4	Bn_size=6
CIFAR-10	0.08	1.23	0.08
SVHN	0.07	0.04	0.04
CIFAR - 100	0.09	1.10	0.08
PaviaU	0.08	0.09	0.06

Table 7. Loss

Datasets	Bn_size=2	Bn_size=4	Bn_size=6
CIFAR-10	0.47	0.42	0.47
SVHN	0.10	0.31	0.05
CIFAR-100	0.45	0.40	0.49
PaviaU	0.12	0.12	0.09

When the Bn_size value is increased in the Improvised Dense Block, it allows for capturing and processing more information within each block. This increased capacity can lead to improved performance as the model can learn more intricate patterns and features. However, it comes at the cost of increased computational and memory requirements due to the larger number of parameters.

By increasing the Bn_size, the model becomes more capable of capturing fine-grained details and nuances in the data, which can contribute to a reduction in both loss and error rates. The additional parameters enable the model to better fit the training data, resulting in improved generalization and lower error rates. It's important to strike a balance between model complexity and performance.

While a higher Bn_size may lead to better results in terms of loss and error rate, it also increases the risk of overfitting, especially if the dataset is limited. Therefore, it's crucial to consider the available resources, computational constraints, and the specific requirements of the application when deciding on the appropriate Bn_size value for the Improvised Dense Block.

4 Results and Discussions

4.1 Dataset

Numerous experiments were conducted on two datasets to substantiate the efficacy of our model. In place of MNIST, we opted for the CIFAR-10 dataset, which shares the same image size, training set division, and test set division. The CIFAR-10 dataset encompasses 60,000 training images and 10,000 test images, each measuring 32x32 pixels. Diverging from MNIST, CIFAR-10 presents a greater complexity due to its diverse object composition. Furthermore, the SVHN dataset, a collection of digit images, was also employed, featuring 73,257 training images and 26,032 test images. The Pavia University Hyperspectral Image (HSI) dataset finds widespread application in the domain of remote sensing and analysis of hyperspectral imagery. It is frequently employed for the assessment and validation of algorithms pertaining to tasks

such as image classification, identification of targets, and various other analyses within the framework of hyperspectral imaging. The spectral composition differs between Pavia Centre, with 102 bands, and Pavia University, with 103 bands. Pavia Centre presents an image size of 1096 by 1096 pixels, while Pavia University is 610 by 610 pixels.

4.2 Experimental Setup

Within our model, we designated the total number of epochs as 40, set the batch size at 50, and initialized the learning rate to 0.001. The complete network architecture was implemented using PyTorch, executed on a Tesla T4 with 16GB RAM within Google Colaboratory.Employing data augmentation methods, we subjected the input images to techniques such as random horizontal flipping. To ensure robustness, we conducted numerous experiments for each scenario, omitting the extreme values from the outcomes and subsequently computing the average value to establish the final result.

4.3 Experiment Metrics

When conducting research or comparing routing algorithms in capsule networks, some common metrics to consider might be:

1. **Classification Accuracy:** The proportion of correctly classified instances (e.g., images) in the test set.
2. **Reconstruction Loss:** Capsule networks often include a reconstruction task to ensure that important information is preserved during encoding.The reconstruction loss gauges the difference between the input data and the data reconstructed from the capsule representations.
3. **Routing Iterations:** The number of routing iterations used during the dynamic routing process. More iterations might lead to more accurate results, but it could also increase computational overhead.
4. **Training Time:** The time it takes to train the capsule network with a specific routing algorithm. This metric is important for practical implementation considerations.

4.4 Experiment Results

We applied the improved Routing algorithm in conjunction with refined dense blocks to the CIFAR-10 and SVHN datasets. The CIFAR-10 dataset encompasses 60,000 color images, each measuring 32x32 pixels, divided across ten classes with 6,000 images per class. This dataset is further segregated into 50,000 training images and 10,000 test images. On the other hand, the SVHN dataset is tailored for digit classification and comprises 600,000 color images, also measuring 32x32 pixels.

We implemented and compared the proposed algorithm with an improvised Dense Block against the CIFAR-10 and SVHN datasets. The assessment encompassed conventional Dynamic Routing, EM Routing, Fast Routing, and Improved Routing, with evaluations conducted using metrics such as time efficiency, error rate, accuracy, and loss. The outcomes conclusively illustrated that the proposed method surpassed the performance of the other techniques.

5 Conclusion and future work

As Capsule Networks can extract contextual and hierarchical information from hyperspectral data they are able to offer enhanced processing efficiency for hyperspectral images (HSI).The time and effort needed for data preparation can be decreased by using feature learning process to generate preprocessing which is more automated and efficient.By inevitably supporting the spectral parameters, CapsNets can better take advantage of the multi-dimensional characteristics of HSI, resulting in enhanced information extraction. Additionally, they only focus on extracting features from images and do not consider their position, size, and orientation.

To tackle these challenges, we suggest the utilization of capsule networks for image classification. Nevertheless, the dynamic routing algorithm employed between the primary and digit capsule layers within capsule networks necessitates intricate computations. To reduce the computational burden, we propose an Improved Routing algorithm.

By comparing the existing and proposed algorithms, we infer that the capsule network with the proposed algorithm outperforms traditional dynamic routing, EM routing, and fast routing algorithms. We observe an improvement in classification performance compared to the original capsule network. In forthcoming endeavors, our objective is to refine the feature extraction phase of the capsule network to effectively manage intricate datasets.

Acknowledgements

None.

Author contributions

B. Thiyagarajan and K.Revathy : Conceptualization, research, writing, first draft revision, data collection, analysis, and result interpretation. **M. Thenmozhi:** creation, supervision, and examination of the study.

Conflicts of interest

The authors declare no conflict of interest.

References

[1] Zhang Y, Li W, Zhang M, Qu Y, Tao R, Qi H. Topological structure and semantic information

transfer network for cross-scene hyperspectral image classification. IEEE Transact Neural Networks Learn Syst 2021. In press.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘‘ImageNet classification with deep convolutional neural networks,’’ in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.

[3] S. Huang, F. Lee, R. Miao, Q. Si, C. Lu, and Q. Chen, "A deep convolutional neural network architecture for interstitial lung disease pattern classification," *Med. Biol. Eng. Comput.*, vol. 58, no. 4, pp. 725–737, Jan. 2020.

[4] K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Deep residual learning for image recognition,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[5] L. Xie, F. Lee, L. Liu, K. Kotani, and Q. Chen, "Scene recognition: A comprehensive survey," *Pattern Recognit.*, vol. 102, Jun. 2020, Art. no. 107205.

[6] L. Xie, F. Lee, L. Liu, Z. Yin, and Q. Chen, ‘‘Hierarchical coding of convolutional features for scene recognition,’’ *IEEE Trans. Multimedia*, vol. 22, no. 5, pp. 1182–1192, May 2020.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ‘‘You only look once: Unified, real-time object detection,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[8] J. Cai, F. Lee, S. Yang, C. Lin, H. Chen, K. Kotani, and Q. Chen, "Pedes- train as points: An improved anchor-free method for centre-based pedestrian detection," *IEEE Access*, vol. 8, pp. 179666–179677, Sep. 2020.

[9] E. Shelhamer, J. Long, and T. Darrell, ‘‘Fully convolutional networks for semantic segmentation,’’ *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, ‘‘Gradient-based learn- ing applied to document recognition,’’ *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan,V. Vanhoucke, and A. Rabinovich, ‘‘Going deeper with convolutions,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[12] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. Int. Conf. Artif. Neural Netw.*, 2011, pp. 44–51.

[13] S. Sabour, N. Frosst, and G. E. Hinton, ‘‘Dynamic routing between capsules,’’ in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3859–3869.

[14] T. Hahn, M. Pyeon, and G. Kim, ‘‘Self-routing capsule networks,’’ in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7656–7665.

[15] Y. LeCun, C. Cortes, and C. J. Burges. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[16] S. S. R. Phaye, A. Sikka, A. Dhall, and D. R. Bathula, ‘‘Multi-level dense capsule networks,’’ in *Proc. Asian Conf. Comput. Vis.*, Dec. 2018, pp. 577–592.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, ‘‘Densely connected convolutional networks,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[18] X. Wang, R. Girshick, A. Gupta, and K. He, ‘‘Non-local neural networks,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7794–7803.

[19] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, ‘‘CBAM: Convolutional block attention module,’’ in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[20] J. Hu, L. Shen, and G. Sun, ‘‘Squeeze-and-excitation networks,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[21] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, pp. 1–18, 2017.

[22] J.-H. Luo, J. Wu, and W. Lin, ‘‘ThiNet: A filter level pruning method for deep neural network compression,’’ in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and

R. Salakhutdinov, ‘‘Dropout: A simple way to prevent neural networks from overfitting,’’ *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[24] S. Yang, F. Lee, R. Miao, J. Cai, L. Chen, W. Yao, K. Kotani, and Q. Chen, ‘‘RS-CapsNet: An advanced capsule network,’’ *IEEE Access*, vol. 8, pp. 85007–85018, 2020.

[25] Sara Sabour, Nicholas Frosst and Geoffrey E Hinton, “Dynamic Routing Between Capsules” in arXiv, https://doi.org/10.48550/arXiv.1710.09829.

[26] Geoffrey Hinton, Sara Sabour and Nicholas Frosst“ Matrix Capsules with EM Routing” Published as a conference paper at ICLR 2018, vol. 15, https://openreview.net/pdf?id=HJWLfGWRb.

[27] BodhisatwaMandal, SwarnenduGhosh, RiteshSarkhel, Nibaran Das, MitaNasipuri "Using dynamic routing to extract intermediate features for developing scalable capsule network" in the international conference on advanced computational and communication, 2018

[28] S. K. Sahu, P. Kumar and A. P. Singh, "Dynamic Routing Using Inter Capsule Routing Protocol between Capsules," 2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim), Cambridge, UK, 2018, pp. 1-5, doi: 10.1109/UKSim.2018.00012.

[29] J. Chen and Z. Liu, "Mask Dynamic Routing to Combined Model of Deep Capsule Network and U-Net," in IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 7, pp. 2653-2664, July 2020, doi: 10.1109/TNNLS.2020.2984686.

[30] F. M. Saif, T. Imtiaz, S. Rifat, C. Shahnaz, W. -P. Zhu and M. O. Ahmad, "CapsCovNet: A Modified Capsule Network to Diagnose COVID-19 From Multimodal Medical Imaging," in IEEE Transactions on Artificial Intelligence, vol. 2, no. 6, pp. 608-617, Dec. 2021, doi: 10.1109/TAI.2021.3104791.

[31] W. Wang, F. Lee, S. Yang and Q. Chen, "An Improved Capsule Network Based on Capsule Filter Routing," in IEEE Access, vol. 9, pp. 109374-109383, 2021, doi: 10.1109/ACCESS.2021.3102489.

[32] Mandal, B., Sarkhel, R., Ghosh, S., Das, N., &Nasipuri, M. (2021). Two-phase Dynamic Routing for Micro and Macro-level Equivariance in Multi-Column Capsule Networks. Pattern Recognition, 109. https://doi.org/10.1016/j.patcog.2020.107595

[33] R. Zeng and Y. Song, "A Fast Routing Capsule Network With Improved Dense Blocks," in IEEE Transactions on Industrial Informatics, vol. 18, no. 7, pp. 4383-4392, July 2022, doi: 10.1109/TII.2021.3128412.

[34] Li, H., Yang, S. & Wang, W. Improved capsule routing for weakly labelled sound event detection. J AUDIO SPEECH MUSIC PROC. 2022, 5 (2022). https://doi.org/10.1186/s13636-022-00239-6

[35] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger “Densely Connected Convolutional Networks” 2018 https://doi.org/10.48550/arXiv.1608.06993

[36] https://blog.paperspace.com/capsule-networks/

[37] https://towardsdatascience.com/capsule-networks-the-new-deep-learning-network-bd917e6818e8

[38] https://www.researchgate.net/figure/a-Routing-by-agreement-in-CapsNets-A-capsule-is-a-group-of-neurons-whose-activity_fig2_343116827#:~:text=In%20this%20case%2C%20capsules%20agree,activity%20in%20the%20boat%20capsule.

[39] T. Liu, X. Lin, W. Jia, M. Zhou, and W. Zhao, ``Regularized attentive capsule network for overlapped relation extraction," in Proc. 28th Int.Conf. Comput. Linguistics, 2020, pp. 6388_6398.

[40] H. Lin, F. Meng, J. Su, Y. Yin, Z. Yang, Y. Ge, J. Zhou, and J. Luo, ``Dynamic context-guided capsule network for multimodal machine translation," in Proc. 28th ACM Int. Conf. Multimedia, Oct. 2020, pp. 1320_1329.

[41]] M. Edraki, N. Rahnavard, and M. Shah, ``Subspace capsule network," in Proc. AAAI Conf. Artif. Intell., Apr. 2020, vol. 34, no. 7, pp. 10745_10753.

[42] F. Wu, J. S. Smith, W. Lu, C. Pang, and B. Zhang, ``Attentive prototype few-shot learning with capsule network-based embedding," in Proc. Eur.Conf. Comput. Vis., Aug. 2020, pp. 237_253.

[43] S. Sabour, A. Tagliasacchi, S. Yazdani, G. E. Hinton, and D. J. Fleet, ``Unsupervised part representation by _ow capsules," 2020,

[44] arXiv:2011.13920. [Online]. Available: http://arxiv.org/abs/2011.13920

[45] A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan, ``CapsuleGAN: Generative adversarial capsule network," in Proc. Eur. Conf. Comput.Vis. (ECCV), 2018, pp. 526_535.

[46] Y. Le, C. He, M. Chen, Y. Wu, X. He, and B. Zhou, ``Learning to predict charges for legal judgment via self-attentive capsule network," Frontiers Artif. Intell. Appl., vol. 325, pp. 1802_1809, May 2020.

[47] J. Yang, P. Zhao, Y. Rong, C. Yan, C. Li, H. Ma, and J. Huang, ``Hierarchical graph capsule network," 2020, arXiv:2012.08734. [Online]. Available: http://arxiv.org/abs/2012.08734