Scientific Research Publishing

Open Journal of Applied Sciences, 2026, 16(4), 1017-1033
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Lightweight Capsule Network Based on Weight Sharing and Top-K Routing

Dazhong Mu*, Ran Li#

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, China
Email: sdnymdz123@163.com, #ran89@usst.edu.cn

How to cite this paper: Mu, D.Z. and Li, R. (2026) Lightweight Capsule Network Based on Weight Sharing and Top-K Routing. *Open Journal of Applied Sciences*, **16**, 1017-1033.
https://doi.org/10.4236/ojapps.2026.164060

Received: March 6, 2026
Accepted: March 30, 2026
Published: April 2, 2026

Abstract

Capsule networks can effectively model the spatial hierarchical relationships among features; however, the dynamic routing mechanism introduces a large number of parameters and high computational complexity, which limits their practical deployment in real-world engineering applications. To address these issues, this paper proposes a lightweight capsule network named WTCaps, based on weight sharing and Top-K routing pruning. Specifically, a weight-sharing strategy is introduced in the capsule transformation stage to significantly reduce the number of model parameters. Meanwhile, a Top-K routing pruning mechanism is employed to perform dynamic routing only among a small subset of highly relevant higher-level capsules, thereby effectively decreasing computational complexity. In addition, a lightweight spatial attention mechanism is incorporated to enhance the representation of critical regions, improving classification performance while maintaining model efficiency. Experimental results on multiple public datasets demonstrate that WTCaps achieves classification accuracy comparable to or better than existing capsule networks, with substantial reductions in both parameter count and computational cost. Furthermore, the proposed model exhibits strong robustness in rotated digit recognition and overlapping digit recognition tasks. This study provides a feasible solution for the efficient design and practical application of capsule networks.

Keywords

Capsule Network, Lightweight Model, Deep Learning, Weight Sharing, Dynamic Routing, Attention Mechanism

*First author.
#Corresponding author.

1. Introduction

With the continuous development of deep learning techniques, neural network-based visual models have achieved remarkable success in tasks such as image classification and object recognition. Among them, Convolutional Neural Networks (CNNs) [1] have become the mainstream approach in current visual tasks due to their local connectivity and weight-sharing mechanisms, which enable effective extraction of local image features. However, traditional CNNs primarily represent features in the form of scalars and rely on pooling operations to achieve spatial dimensionality reduction. This process weakens the ability to capture spatial relationships among features to a certain extent. When objects exhibit variations in pose, scale, or partial occlusion, CNN-based models often require large amounts of training data and deeper network architectures to achieve satisfactory robustness.

To address these limitations, Capsule Network (CapsNet) was proposed by Geoffrey Hinton and colleagues [2]. In this framework, capsules represented as vectors or matrices are used to encode features. Such representations not only indicate the probability of feature existence but also explicitly capture the pose information of objects. Capsule networks employ a dynamic routing mechanism to establish connections between lower-level capsules and higher-level capsules, thereby enabling more effective modeling of part-whole relationships. In tasks such as handwritten digit recognition, capsule networks have demonstrated strong structural representation capability and certain robustness to interference, which has attracted widespread attention. However, the original capsule network still faces significant efficiency challenges in practical applications. On the one hand, independent transformation matrices are typically used for feature mapping between lower-level and higher-level capsules, causing the number of model parameters to grow linearly or even faster with the number of capsules. On the other hand, the dynamic routing process repeatedly computes coupling coefficients and agreement measures among all capsule pairs, leading to high computational complexity and long inference time. These issues limit the application of capsule networks in resource-constrained scenarios and constitute a major obstacle to their broader adoption.

To address the issues of large parameter size and high computational cost in capsule networks, existing studies have explored various strategies. For example, some works introduce weight-sharing strategies to reduce the number of transformation matrices [3], simplify or replace the dynamic routing algorithm to lower computational cost [4], or incorporate attention mechanisms to enhance the representation of critical information [5]. Although these approaches alleviate the complexity of capsule networks to some extent, several limitations remain. Some methods mainly focus on parameter compression while having limited impact on the overall computational cost, whereas others fail to fully exploit the differences in spatial importance among features within lower-level capsules, resulting in a trade-off between model efficiency and performance.

Based on the above analysis, this paper investigates the problem of lightweight design and efficient inference for capsule networks. While maintaining the structural advantages of capsule networks as much as possible, improvements are

made to their key components. Specifically, this work redesigns the transformation mechanism between capsules and the routing strategy, and incorporates a feature enhancement mechanism along the spatial dimension to explore a more balanced capsule network architecture in terms of parameter scale and computational efficiency. Experimental results demonstrate that the proposed method can effectively reduce the number of model parameters while maintaining strong classification performance, and it also shows advantages in practical inference efficiency.

2. Related Work

2.1. Capsule Networks

Capsule Network was proposed by Geoffrey Hinton and Sara Sabour, aiming to characterize both the existence probability of features and their spatial attributes through vectorized representations, thereby compensating for the limitations of traditional Convolutional Neural Networks in modeling spatial hierarchical relationships. Compared with scalar neurons, capsules are capable of simultaneously representing whether a feature exists as well as its pose, orientation, and other spatial information, which provides advantages in tasks such as image classification and object recognition.

In recent years, research on capsule networks has mainly focused on network architecture design, routing mechanism improvements, and the expansion of application scenarios. From the perspective of network structure, researchers have attempted to enhance the feature representation capability of models by stacking multiple capsule layers or introducing different forms of capsule modules [6]. Regarding routing mechanisms, dynamic routing serves as a core component of capsule networks. Through multiple iterations, it aggregates information from lower-level capsules to higher-level capsules and is considered a key factor contributing to the modeling capability of capsule networks. For example, [4] introduces sparse attention routing to improve both efficiency and accuracy.

However, as the number and dimensionality of capsules increase, the dynamic routing mechanism requires iterative computation over a large number of capsule pairs, resulting in a significant growth in both model parameters and computational complexity. This issue limits the practical deployment of capsule networks, especially in scenarios with constrained computational resources.

2.2. Lightweight Research on Capsule Networks

To address the large parameter scale and high computational cost of capsule networks, existing studies have explored lightweight improvements from perspectives such as network architecture design and routing strategies. Among these approaches, parameter compression and weight sharing are common techniques for reducing model complexity. By sharing transformation parameters among multiple capsules, the number of transformation matrices can be effectively reduced, thereby alleviating the parameter growth problem as the capsule scale increases.

For instance, [3] proposed the concept of shared-weight capsules, where all primary capsules share a single transformation matrix when mapping to a higher-level capsule. This approach significantly reduces the number of parameters and lowers computational complexity.

In addition, some studies attempt to simplify the dynamic routing process. One approach is to reduce the number of routing iterations. For example, [7] replaces traditional dynamic routing with a non-iterative self-attention routing mechanism, thereby reducing both computation and parameter requirements. Another direction focuses on optimizing routing computation through parallelization. For instance, [8] proposes a parallelized dynamic routing strategy that reduces computational complexity and demonstrates significant advantages over the traditional CapsNet in terms of hardware resource utilization, inference speed, and energy consumption. Although these methods improve operational efficiency to some extent, they may also affect the precision of agreement modeling between capsules.

Overall, existing lightweight research has made certain progress in reducing the number of parameters, but limitations remain in controlling computational complexity. Some methods compress parameters without significantly reducing the computational overhead in the routing stage. Therefore, how to simultaneously balance parameter scale and computational efficiency while maintaining model performance remains a key issue in lightweight capsule network research.

2.3. Attention Mechanism

The Attention Mechanism has been widely applied in convolutional neural networks and sequence modeling tasks as an effective method for feature selection and information weighting [9]. Its core idea is to assign different weights to different features, guiding the model to focus on information that is more important for the current task, thereby improving overall representation capability.

In recent years, some studies have explored integrating attention mechanisms into capsule networks to enhance the ability of capsules to perceive critical information. Such methods typically introduce attention modules either during the feature extraction stage or the capsule aggregation stage. By weighting lower-level features or capsule outputs, these approaches highlight important regions or key capsules. For example, [10] employs a multi-branch attention mechanism to facilitate information transmission between capsule layers.

Although the attention mechanism can improve the feature representation capability of capsule networks to a certain extent, its introduction is often accompanied by additional parameters and computational overhead, which conflicts with the goal of lightweight capsule networks. Therefore, how to effectively integrate attention mechanisms with capsule network architectures while controlling model complexity—so that useful feature representations are enhanced without significantly increasing computational burden—remains an open research problem.

3. Proposed Model

3.1. Overall Model Architecture Design

To address the issues of large parameter scale and high computational complexity in traditional Capsule Network architectures, this paper proposes a lightweight improved model based on the classical capsule network framework. The complete architecture of the proposed model is illustrated in Figure 1. The overall structure still follows the fundamental design of convolutional feature extraction, the PrimaryCaps layer, and the higher-level capsule classification layer, in order to preserve the capability of capsule networks to model spatial hierarchical relationships and pose information. On this basis, targeted improvements are made to the original architecture from three aspects: model parameter organization, feature representation enhancement, and dynamic routing computation strategy.

First, a parameter-sharing mechanism is introduced during the transformation process of higher-level capsules. This mechanism reduces redundant linear transformation parameters between lower-level capsules and higher-level capsules, thereby effectively compressing the model size. Second, a lightweight spatial attention mechanism is incorporated at the PrimaryCaps stage to adaptively reweight feature responses at different spatial locations, enhancing the discriminative power of the capsule input features. Finally, during the dynamic routing stage, a score-based routing pruning strategy is adopted. Dynamic routing computation is performed only among a small number of higher-level capsules with high matching scores, thereby reducing the computational overhead of the routing process.

Through these improvements, the proposed model maintains the structural advantages of the original capsule network while effectively reducing the parameter scale and providing a feasible approach for further lowering computational complexity.

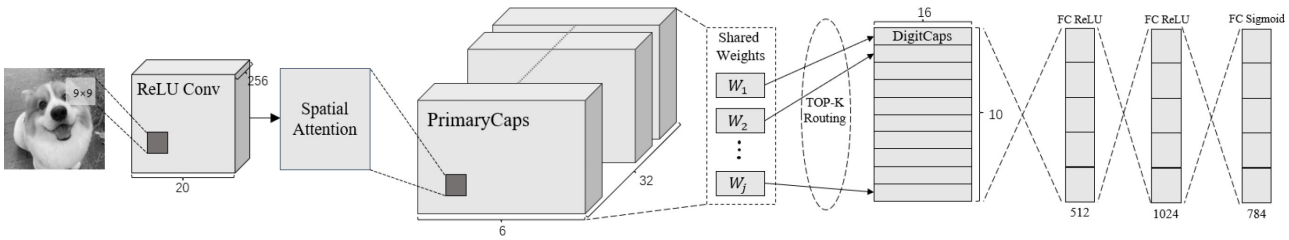

Figure 1. WTCaps network model diagram.

3.2. Shared Transformation Matrix

In the classical Capsule Network architecture, feature mapping from lower-level capsules to higher-level capsules is achieved through a set of independent linear transformation matrices. Specifically, for each pair consisting of a lower-level capsule and a higher-level capsule, the network learns a dedicated transformation matrix to map the pose vector of the lower-level capsule into the prediction space of the corresponding higher-level capsule. Let the output vector of the *iii*-th lower-

level capsule be:

$$u_i \in \mathbb{R}^d \tag{1}$$

The prediction vector corresponding to the j higher-level capsule is computed using an independent transformation matrix:

$$\hat{u}_{j|i} = W_{ij} u_i \tag{2}$$

where W_{ij} denotes the transformation matrix between the *iii*-th lower-level capsule and the *jjj*-th higher-level capsule. This design implies that each capsule pair requires its own set of parameters. As the number of capsules increases, the number of parameters grows rapidly, leading to significant parameter redundancy.

To address this issue, this paper redesigns the transformation mechanism between capsules by introducing a parameter-sharing strategy based on higher-level capsules. Instead of assigning an independent transformation matrix to each pair of lower-level and higher-level capsules, only one shared linear transformation matrix is learned for each higher-level capsule. All lower-level capsules use this shared parameter when mapping to that higher-level capsule:

$$\hat{u}_{j|i} = W_j u_i \tag{3}$$

where $W_j \in \mathbb{R}^{d' \times d}$ represents the shared transformation matrix corresponding to the *jjj*-th higher-level capsule. Through this approach, lower-level capsules no longer rely on independent parameters for differentiation, thereby significantly reducing the parameter scale of the model.

After the pose mapping stage, the input vector of a higher-level capsule is obtained by computing the weighted sum of prediction vectors from all lower-level capsules:

$$s_j = \sum_i a_{ij} \hat{u}_{j|i} \tag{4}$$

where a_{ij} represents the contribution coefficient of the *i* lower-level capsule to the *j* higher-level capsule, which measures the degree of agreement between them. Functionally, this coefficient is similar to the routing coefficient in the original capsule network. However, it no longer relies on the coupling modeled by independent transformation matrices; instead, it acts as an independent weighting factor in constructing the input of higher-level capsules.

Finally, the output of the higher-level capsule is obtained through a nonlinear squashing function:

$$v_j = \text{squash}\left(s_j\right) \tag{5}$$

which ensures that the length of the output vector reflects the probability of the existence of the corresponding class.

Through the proposed parameter-sharing mechanism, the model substantially reduces redundant linear transformation parameters between lower-level and higher-level capsules while preserving the core principles of capsule networks.

3.3. Top-K Routing Pruning Mechanism

In the classical Capsule Network, the dynamic routing process computes coupling coefficients between each lower-level capsule and all higher-level capsules, and continuously updates them through multiple iterations. Although this fully connected routing strategy can effectively model the relationship between parts and wholes, its computational complexity grows linearly with the number of lower-level and higher-level capsules, resulting in considerable computational overhead in practical applications. In the PrimaryCaps layer, a large number of lower-level capsules are typically generated. However, many of them contribute little to the final classification result but still fully participate in the routing computation, leading to significant redundancy.

To address this issue, this paper introduces a Top-K routing pruning mechanism in the dynamic routing stage. By pre-screening the matching relationships between lower-level and higher-level capsules, only a subset of routing connections with higher contributions is retained for subsequent dynamic routing computation. In this way, the computational cost of the routing stage can be reduced without significantly affecting model performance. The schematic diagram of the Top-K routing process is shown in **Figure 2**.

Specifically, under the shared transformation matrix setting, a lower-level capsule obtains a prediction vector for the j higher-level capsule after linear transformation. Based on the consistency relationship between the prediction vector and the current output vector of the higher-level capsule, the matching score between them can be defined as:

$$s_{ij} = \hat{u}_{j|i} \cdot v_j \tag{6}$$

This score essentially measures the degree of support that the *iii*-th lower-level capsule provides to the *jjj*-th higher-level capsule, and it is consistent in form with the agreement term used to update routing logits in the original dynamic routing algorithm.

In the Top-K routing pruning strategy, routing coefficients are not computed and updated for all capsule pairs. Instead, for each lower-level capsule, the top K candidate higher-level capsules with the highest matching scores are selected from all higher-level capsules, forming a candidate set K_i. Only when $j \in K_i$ does the lower-level capsule participate in the dynamic routing computation of the corresponding higher-level capsule. All other routing connections are directly pruned for the current sample.

Based on this mechanism, the normalization process of routing coefficients is restricted within the Top-K subspace:

$$a_{ij} \begin{cases} \dfrac{\exp\left(b_{ij}\right)}{\sum_{k \in K_i} \exp\left(b_{ik}\right)}, & j \in K_i \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where b_{ij} denotes the routing logits, whose update rule still follows the iterative

procedure of the original dynamic routing algorithm. It can be seen that the Top-K routing pruning strategy does not alter the core principle of dynamic routing; instead, it explicitly constrains the routing search space, thereby reducing unnecessary interactions between irrelevant capsules.

Through this mechanism, the number of higher-level capsules involved in the routing computation for each lower-level capsule is reduced to K. As a result, the computational cost of operations such as vector inner products and weighted summations in the routing stage is significantly decreased. Meanwhile, since the retained routing connections correspond to higher matching scores, the pruning strategy can also suppress noisy routing to some extent, which helps improve the stability of the dynamic routing process.

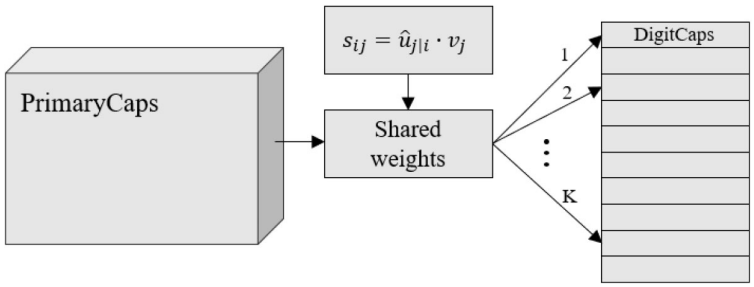

Figure 2. TOP-K routing diagram.

3.4. Spatial Attention Mechanism

Under the shared transformation matrix and the Top-K routing pruning mechanism, the number of parameters and the computational complexity in the higher-level capsule stage are significantly reduced. However, this lightweight design also places higher demands on the quality of feature representations produced by lower-level capsules. If the lower-level capsules contain excessive irrelevant regions or background noise, it may negatively affect the consistency computation of prediction vectors during the subsequent routing process.

To address this issue, this paper introduces a lightweight spatial attention mechanism into the PrimaryCaps layer to adaptively reweight spatial features before generating lower-level capsules. The attention structure is inspired by the spatial attention component of Convolutional Block Attention Module (CBAM) [11]. By modeling the importance of different spatial locations, this mechanism guides the network to focus on more discriminative regional features.

It should be noted that this spatial attention module only performs weighting on feature maps without altering their spatial dimensions or the number of channels. Moreover, it does not introduce additional capsule structures. Therefore, it does not increase the number of capsules nor interfere with the subsequent capsule reshaping and routing mechanisms. The attention-enhanced feature maps are then used to construct lower-level capsule vectors, thereby improving their stability and effectiveness under the shared-weight transformation and Top-K routing pruning settings.

By introducing spatial attention at the lower-level capsule stage, the proposed method maintains the lightweight advantage of the model while effectively alleviating the potential performance degradation caused by parameter sharing and routing pruning.

4. Experiments

In the experimental design, the performance of the proposed model is evaluated from multiple perspectives. First, the dataset and model parameters are set up. Subsequently, the model is assessed in terms of image classification performance, robustness to affine-transformed images, and computational efficiency.

The experiments are conducted on a personal computer with the following hardware configuration: an NVIDIA GeForce GTX 4090 GPU, Windows 11 operating system, Intel® Core™ i9-10900X CPU @ 3.70 GHz. The models are implemented in Python using the PyTorch deep learning framework. The datasets are divided into training, validation, and test sets, with the validation set consisting of 20% of the training samples. The hyperparameters are set as follows: batch size = 128, optimizer = Adam, initial learning rate = 0.001 with an adaptive learning rate schedule monitored by validation loss. If the validation loss does not improve for 15 consecutive epochs, the learning rate is multiplied by 0.1, but not reduced below 1e-6. Models are trained for 200 epochs, and the model with the highest validation accuracy is selected for testing.

4.1. Dataset Description

This study evaluates the classification performance of the proposed model on five publicly available datasets: MNIST, Fashion-MNIST, CIFAR-10, SVHN, and SmallNORB.

MNIST: A grayscale handwritten digit dataset containing classes 0 - 9, with images of size 28 × 28. It consists of 60,000 training images and 10,000 test images.

Fashion-MNIST: A grayscale image dataset of 70,000 fashion products in 10 categories, with images of size 28 × 28.

CIFAR-10: A dataset of 60,000 RGB natural images across 10 categories, with image size 32 × 32. Each category contains 5,000 training images and 1,000 test images.

SVHN: A dataset for digit classification, containing over 600,000 real-world RGB images of printed digits (0 - 9), each of size 32 × 32.

SmallNORB: A dataset of 3D toy objects across 5 categories, with each object captured at 18 different azimuth angles (0˚ - 340˚) and 9 different elevations, with image size 96 × 96.

4.2. Classification Results

To verify the classification performance and lightweight efficiency of the proposed WTCaps model, comparative experiments were conducted on MNIST, Fashion-MNIST, SVHN, CIFAR-10, and SmallNORB datasets against several representative

capsule network models. The experimental results are summarized in Table 1.

The results indicate that WTCaps achieves stable classification performance across all five datasets. Specifically, the error rates on MNIST, SVHN, and CIFAR-10 are comparable to those of current mainstream capsule network methods. In terms of parameter scale, WTCaps contains 4.20 M parameters, significantly fewer than models such as DRCaps, AWRCaps, Aff-CapsNet, and Self-Routing. Meanwhile, WTCaps achieves classification performance that is better than or comparable to other lightweight methods, including CAPCaps and GraCapsnets.

Overall, the experimental results demonstrate that the proposed WTCaps model, based on weight sharing and Top-K routing pruning, effectively reduces model parameter size while maintaining strong classification performance, achieving a reasonable balance between accuracy and model complexity.

Table 1. Classification error rates and parameters of different models on five datasets.

Model	MNIST		FashionMNIST		SVHN		CIFAR10		SmallNorb	
	Errors /%	Params/M	Errors /%	Params/M	Errors /%	Params/M	Errors /%	Params/M	Errors /%	Params/M
Attn-Routing [12]	0.54	5.31	-	-	-	-	12.71	9.60	-	-
EM-Routing [13]	-	-	-	-	-	-	11.90	0.45	1.80	0.30
VB-Routing [14]	-	-	5.20	0.17	3.90	0.32	11.20	0.32	1.60	0.17
Self-Routing [15]	-	-	-	-	3.12	140.30	7.86	140.30	-	-
DRCaps [2]	0.32	6.10	5.45	6.10	3.85	6.10	9.94	6.10	3.52	6.10
Aff-CapsNet [16]	0.46	8.20	7.47	8.20	7.85	8.20	23.72	8.20	-	-
MS-CapsNet [17]	-	-	7.30	10.80	-	-	24.30	11.20	-	-
FRMS [18]	0.42	1.20	6.00	1.20	-	-	15.60	1.20	-	-
GraCapsnets [19]	-	-	-	-	2.98	2.48	7.99	2.48	-	-
AWRCaps [20]	0.32	5.80	5.18	5.8	3.69	5.8	9.54	5.80	2.99	5.8
CAPCaps [21]	0.29	3.00	5.41	3.00	3.46	3.54	7.40	3.54	4.90	3.54
WTCaps	0.33	4.20	5.25	4.20	3.52	4.20	7.66	4.20	3.27	4.20

4.3. Ablation Study

To analyze the impact of each improved module on model performance and complexity, ablation experiments were conducted on the MNIST and CIFAR-10 datasets, as shown in Table 2, with the baseline model being the traditional capsule network model [2]. The results indicate the following: Weight Sharing (WS): In-

troducing the weight-sharing mechanism reduces the number of model parameters from 6.10 M to 4.30 M. However, a slight decrease in classification accuracy is observed, indicating that while weight sharing effectively compresses the model size, it imposes some constraints on feature representation capability. Top-K Routing Pruning (TR): When only the Top-K routing pruning is applied, the model parameter count remains unchanged, but FLOPs decrease significantly from 552.38 M to 260.15 M, demonstrating a clear reduction in computational complexity. Classification accuracy, however, also decreases slightly, suggesting that excessive pruning of routing connections may result in the loss of discriminative information. Spatial Attention (CA): Introducing the spatial attention mechanism alone has little effect on model parameters or computational complexity, but classification accuracy improves on both datasets, validating the effectiveness of spatial attention in enhancing feature discriminability.

Combining TR and CA leads to a substantial reduction in computational complexity while achieving better classification performance compared to using TR alone, indicating that spatial attention can partially mitigate performance loss caused by routing pruning.

Finally, the full WTCaps model, incorporating WS, TR, and CA, reduces the parameter count and FLOPs to 4.20 M and 200.90 M, respectively, while achieving classification accuracies of 99.67% on MNIST and 92.34% on CIFAR-10. These results demonstrate that the proposed method achieves a favorable balance between model lightweight efficiency and classification performance.

Table 2. Generalization performance from MNIST and CIFAR10 datasets.

Model	Params/M	Flops/MFlops	MNIST Accuracy/%	CIFAR10 Accuracy/%
Baseline	6.10	552.38	99.68	90.06
Baseline + WS	4.30	552.38	99.52	88.41
Baseline + TR	6.10	260.15	99.50	89.02
Baseline + CA	6.10	560.10	99.71	91.18
Baseline + WS + CA	4.30	560.10	99.60	90.02
Baseline + TR + CA	6.10	200.90	99.61	91.05
WTCaps	4.20	200.90	99.67	92.34

4.4. Robustness to Affine-Transformed Images

To evaluate the robustness of the proposed algorithm to affine transformations, all models were trained on the MNIST dataset and then tested on the affNIST dataset. In this setup, the training images were generated by first padding the original MNIST images (28×28) to 40×40 pixels and then randomly placing the digits within the 40×40 background. This procedure introduces variations in position and scale, allowing assessment of the model's ability to maintain accurate classification under affine transformations.

Table 3. Generalization performance from MNIST to affNIST dataset.

Model	MNIST Accuracy	affNIST Accuracy
CNN [1]	99.2%	85.90%
DRCaps [2]	99.2%	79.0%
HitNet [22]	99.6%	83.3%
GCaps [23]	98.4%	89.1%
SparseCaps [24]	99.0%	90.1%
Attn-Routing [11]	99.5%	91.6%
SCAE [25]	98.5%	92.2%
Aff-CapsNet [16]	99.23%	93.12%
EM-Routing [13]	99.2%	93.1%
AWRCaps [20]	99.2%	96.73%
WTCaps	99.2%	96.69%

The test set was generated by first padding the original MNIST images to 40 × 40 pixels, and then applying affine transformations to the digits. These transformations included rotations within ±20˚, shearing within ±45˚, horizontal and vertical scaling in the range of 0.8 - 1.2, and random translations up to 8 pixels in each direction, resulting in the affNIST dataset. For fair comparison, only models that achieved at least 99.2% test accuracy on the original MNIST dataset were evaluated on affNIST. The proposed capsule network model achieved a classification accuracy of 96.69% on affNIST, as shown in Table 3. These results demonstrate that the proposed method achieves state-of-the-art generalization performance on affine-transformed images.

4.5. Robustness to Rotated Digit Images

To further evaluate the robustness of the proposed WTCaps model under complex geometric transformations, comparative experiments were conducted on the rotated MNIST and rotated SVHN datasets, with rotation angles set to ranges of $\pm\theta_1$ and $\pm\theta_2$, respectively, as summarized in Table 4.

The experimental results show that under standard testing conditions without rotation, all models achieve high classification accuracy. As the rotation range increases, the classification performance of all models declines to varying degrees; however, capsule network-based methods generally exhibit stronger robustness. Compared to ResNet18, WTCaps shows a smaller decrease in accuracy under large-angle rotations, indicating better adaptability to pose variations. In comparison with other capsule network models such as DRCaps, DeepCaps, and CAP-Caps, WTCaps maintains comparable classification performance on rotated MNIST and rotated SVHN datasets, and in some rotation ranges, achieves slightly better or equivalent results.

These results indicate that, even with significantly reduced model parameters

and computational complexity, WTCaps effectively preserves the inherent robustness of capsule networks to spatial transformations such as rotation.

Table 4. Classification performance of different algorithms on Rotated MNIST and Rotated SVHN.

Model	MNIST Accuracy/%			SVHN Accuracy/%		
	0	(−30˚, 30˚)	(60˚, 60˚)	0	(−30˚, 30˚)	(−60˚, 60˚)
DRCaps [2]	99.68	97.70	89.14	96. 15	92.89	85.25
Resnet18	99.27	97.10	80.75	95.78	75.17	70.76
DeepCaps [26]	99.72	97.82	88.20	97.16	91.80	83.55
CAPCaps [20]	99.71	97.88	89.26	96.54	94.70	87.51
WTCaps	99.67	97.86	89.28	96.48	94.68	87.65

4.6. Performance on Overlapping Digit Recognition

To evaluate the model's ability to model and separate overlapping targets in complex scenarios, comparative experiments were conducted on the MultiMNIST dataset using several representative methods. The MultiMNIST dataset contains images with multiple overlapping digits, placing higher demands on the model's feature disentanglement and object perception capabilities, as shown in Table 5.

The results indicate that traditional CNNs achieve relatively low recognition accuracy on this task, whereas methods incorporating capsule structures and dynamic routing mechanisms significantly improve classification performance. Compared with capsule network models such as Dynamic Routing and Aff-CapsNet, WTCaps achieves comparable classification accuracy to AWRCaps while drastically reducing the number of parameters. Specifically, WTCaps achieves a test accuracy of 95.95% with only 4.20 M parameters.

Although AWRCaps attains slightly higher accuracy, its larger parameter scale makes the model less compact. WTCaps strikes a better balance between model compactness and recognition performance. These results demonstrate that the proposed weight-sharing and Top-K routing pruning mechanisms maintain the capsule network's strong modeling capability for overlapping objects while effectively reducing model size, highlighting the practical applicability of WTCaps in complex visual scenarios.

Table 5. Classification performance of different algorithms on MultiMNIST.

Model	MutliMNIST Accuracy	Params
CNN [1]	91.99%	24.56 M
DRCaps [2]	94.80%	11.36 M
Aff-CapsNet [15]	95.49%	8.20 M
AWRCaps [19]	95.97% ± 0.18	5.80 M
WTCaps	95.95% ± 0.22	4.20 M

4.7. Visualization of Reconstruction Results

To further verify the feature representation capability of the proposed WTCaps model during the lightweighting process, experiments were conducted to analyze its reconstruction performance. In capsule networks, a reconstruction branch is introduced after the classification capsules, using the input image as a supervisory signal to reconstruct the original sample. This process provides an intuitive visualization of the model's ability to capture the pose and structural information of the target object. The reconstruction results are shown in Figure 3.

The experimental results indicate that, compared with baseline capsule networks, WTCaps is still able to reconstruct the overall contours and key structural features of input digits fairly accurately, even under significantly reduced parameters and computational complexity. No noticeable shape distortions or semantic omissions were observed. This demonstrates that the proposed weight-sharing and Top-K routing pruning strategies do not compromise the capsule network's ability to model spatial hierarchical relationships. These results further confirm that WTCaps maintains strong feature representation and structural reconstruction capabilities while achieving model lightweighting.

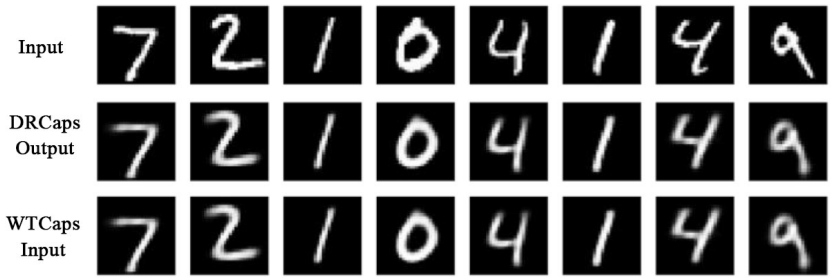

Figure 3. Reconstruction results of DRCaps and WTCaps.

4.8. Comparison of Computational Efficiency

To more comprehensively evaluate the characteristics of the proposed algorithm, a comparative experiment on computational efficiency was conducted. Classic capsule network models and recent lightweight capsule network models were tested using input images of the same size and number of channels, and their parameter counts and floating-point operations (FLOPs) were measured to assess computational performance, as shown in Table 6.

The results indicate that traditional capsule networks, such as DRCaps and AWRCaps, maintain good performance but suffer from high parameter counts and FLOPs. In contrast, GraCapsNet and CAPCaps reduce model parameters to some extent through structural simplification, but their computational complexity remains relatively high.

The proposed WTCaps model achieves a more balanced performance in both parameter scale and computational complexity. Specifically, WTCaps contains only 4.20M parameters—significantly fewer than DRCaps and AWRCaps—and reduces the computation to 200.90 MFLOPs, approximately 63.6% lower than

DRCaps. These results demonstrate that WTCaps effectively reduces redundant computations in the dynamic routing stage without significantly increasing structural complexity, thereby significantly improving overall computational efficiency while preserving the capsule network's modeling capabilities.

Table 6. Comparison of the computational efficiency of different algorithms.

Model	Params/M	Flops/MFlops
DRCaps [2]	6.10	552.38
GraCapsnet [18]	2.48	380.00
CAPCaps [20]	3.00	380.00
AWRCaps [19]	5.80	552.38
WTCaps	4.20	200.90

5. Conclusions

This paper addresses the challenges faced by capsule networks in practical applications, including large model parameter sizes, high computational complexity, and inefficient dynamic routing. We propose a lightweight capsule network, WTCaps, based on weight sharing and a Top-K routing mechanism. By systematically improving the traditional capsule network architecture, WTCaps effectively reduces model parameters and computational overhead while preserving the advantages of capsule networks in modeling spatial relationships, thereby enhancing its practical applicability.

Specifically, the proposed approach introduces a weight-sharing mechanism in the digit capsule layer, allowing higher-level capsules of the same category to share transformation matrices, which significantly reduces the number of model parameters. Next, a Top-K routing pruning strategy is employed to limit the number of candidate capsules participating in dynamic routing, thereby reducing redundant computations during the routing stage and lowering overall computational complexity. Additionally, a lightweight spatial attention mechanism is incorporated in the PrimaryCaps layer to enhance the model's focus on key local features, mitigating potential performance loss caused by model lightweighting.

Extensive experiments demonstrate that WTCaps achieves competitive classification performance across multiple datasets, while outperforming several existing lightweight capsule network methods in terms of parameter size and computational cost. Moreover, WTCaps maintains strong robustness in rotated and overlapping digit recognition tasks, validating the effectiveness and generalization capability of the proposed method.

Although WTCaps achieves a balance between efficiency and performance, there remains room for further optimization. Future work may explore more refined routing pruning strategies, adaptive routing mechanisms, and integration with other lightweight techniques to further enhance the performance and practical value of capsule networks in complex visual tasks.

Funding

Shanghai Municipl Education Commission AI Program-SHJWAIJK241201.
Key R&D Program of the Ministry of Science and Technology (2022YFF0607701).
National Natural Science Foundation of China (12372384, 12072200).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., *et al.* (2018) Recent Advances in Convolutional Neural Networks. *Pattern Recognition*, **77**, 354-377. https://doi.org/10.1016/j.patcog.2017.10.013

[2] Sabour, S., Frosst, N. and Hinton, G.E. (2017) Dynamic Routing between Capsules. arXiv: 1710.09829.

[3] Huang, R., Li, J., Wang, S., Li, G. and Li, W. (2020) A Robust Weight-Shared Capsule Network for Intelligent Machinery Fault Diagnosis. *IEEE Transactions on Industrial Informatics*, **16**, 6466-6475. https://doi.org/10.1109/tii.2020.2964117

[4] Geng, X., Wang, J., Gong, J., Xue, Y., Xu, J., Chen, F., *et al.* (2024) OrthCaps: An Orthogonal CapsNet with Sparse Attention Routing and Pruning. 2024 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Seattle, 16-22 June 2024, 6037-6046. https://doi.org/10.1109/cvpr52733.2024.00577

[5] Jiao, Y., Zhao, L. and Xu, H. (2021) Research on Capsule Network Based on Attention Mechanism. *International Journal of Advanced Network, Monitoring and Controls*, **6**, 1-8. https://doi.org/10.21307/ijanmc-2021-011

[6] Garau, N., Bisagno, N., Sambugaro, Z. and Conci, N. (2022). Interpretable Part-Whole Hierarchies and Conceptual-Semantic Relationships in Neural Networks. 2022 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), New Orleans, 18-24 June 2022, 13679-13688. https://doi.org/10.1109/cvpr52688.2022.01332

[7] Mazzia, V., Salvetti, F. and Chiaberge, M. (2021) Efficient-CapsNet: Capsule Network with Self-Attention Routing. *Scientific Reports*, **11**, Article No. 14634. https://doi.org/10.1038/s41598-021-93977-0

[8] Javadinia, S. and Baniasadi, A. (2023) PDR-CapsNet: An Energy-Efficient Parallel Approach to Dynamic Routing in Capsule Networks. arXiv: 2310.03212.

[9] Brauwers, G. and Frasincar, F. (2023) A General Survey on Attention Mechanisms in Deep Learning. *IEEE Transactions on Knowledge and Data Engineering*, **35**, 3279-3298. https://doi.org/10.1109/tkde.2021.3126456

[10] Zeng, R., Qin, Y. and Song, Y. (2024) A Non-Iterative Capsule Network with Interdependent Agreement Routing. *Expert Systems with Applications*, **238**, Article ID: 122284. https://doi.org/10.1016/j.eswa.2023.122284

[11] Woo, S., Park, J., Lee, J. and Kweon, I.S. (2018) CBAM: Convolutional Block Attention Module. In: Ferrari, V., Hebert, M., Sminchisescu, C. and Weiss, Y., eds., *Computer Vision—ECCV* 2018, Springer, 3-19. https://doi.org/10.1007/978-3-030-01234-2_1

[12] Choi, J., Seo, H., Im, S. and Kang, M. (2019) Attention Routing between Capsules. 2019 *IEEE/CVF International Conference on Computer Vision Workshop* (*ICCVW*), Seoul, 27-28 October 2019, 1981-1989. https://doi.org/10.1109/iccvw.2019.00247

[13] Hinton, G.E., Sabour, S. and Frosst, N. (2018) Matrix Capsules with EM Routing. *International Conference on Learning Representations* 2018, Vancouver, 30 April-3 May 2018, 1-15.

[14] De Sousa Ribeiro, F., Leontidis, G. and Kollias, S. (2020) Capsule Routing via Variational Bayes. *Proceedings of the AAAI Conference on Artificial Intelligence*, **34**, 3749-3756. https://doi.org/10.1609/aaai.v34i04.5785

[15] Hahn, T., Pyeon, M. and Kim, G. (2019) Self-Routing Capsule Networks. *Proceedings of the* 33*rd International Conference on Neural Information Processing Systems*, Vancouver, 8-14 December 2019, 7658-7667.

[16] Gu, J. and Tresp, V. (2020) Improving the Robustness of Capsule Networks to Image Affine Transformations. 2020 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Seattle, 13-19 June 2020, 7283-7291. https://doi.org/10.1109/cvpr42600.2020.00731

[17] 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**, 1850-1854. https://doi.org/10.1109/lsp.2018.2873892

[18] Zhang, S., Zhao, W., Wu, X. and Zhou, Q. (2021) Fast Dynamic Routing Based on Weighted Kernel Density Estimation. *Concurrency and Computation*: *Practice and Experience*, **33**, e5281. https://doi.org/10.1002/cpe.5281

[19] Gu, J. (2021) Interpretable Graph Capsule Networks for Object Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, **35**, 1469-1477. https://doi.org/10.1609/aaai.v35i2.16237

[20] Song, Y., Qin, Y.Z. and Zeng, R. (2023) Attention-Based Capsule Network with Shared Parameters. *Control and Decision*, **38**, 1577-1585.

[21] Zhu, Z.H. and Song, Y. (2024) Lightweight Capsule Network Fusing Attention and Capsule Pooling. *Electronics Science and Technology*, **37**, 1-8, 31.

[22] Deliege, A., Cioppa, A. and Van Droogenbroeck, M. (2019) An Effective Hit-Or-Miss Layer Favoring Feature Interpretation as Learned Prototypes Deformations. arXiv: 1911.05588.

[23] Lenssen, J.E., Fey, M. and Libuschewski, P. (2018) Group Equivariant Capsule Networks. *Proceedings of the* 32*nd International Conference on Neural Information Processing Systems*, Montréal, 3-8 December 2018, 8858-8867.

[24] Rawlinson, D., Ahmed, A. and Kowadlo, G. (2018) Sparse Unsupervised Capsules Generalize Better. arXiv: 1804.06094.

[25] Kosiorek, A., Sabour, S., The, Y.W., *et al.* (2019) Stacked Capsule Autoencoders. *Proceedings of the* 33*rd International Conference on Neural Information Processing Systems*, Vancouver, 8-14 December 2019, 15512-15522.

[26] Rajasegaran, J., Jayasundara, V., Jayasekara, S., Jayasekara, H., Seneviratne, S. and Rodrigo, R. (2019) DeepCaps: Going Deeper with Capsule Networks. 2019 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Long Beach, 15-20 June 2019, 10717-10725. https://doi.org/10.1109/cvpr.2019.01098