applied sciences

MDPI

Article

Non-Iterative Cluster Routing: Analysis and Implementation Strategies

Huong Pham [1] and Samuel Cheng [2,*]

[1] Norman Campus, School of Electrical and Computer Engineering, University of Oklahoma, Norman, OK 73019, USA; huong.n.pham01@gmail.com
[2] Tulsa Campus, School of Electrical and Computer Engineering, University of Oklahoma, Tulsa, OK 74135, USA
* Correspondence: samuel.cheng@ou.edu

Abstract: In conventional routing, a capsule network employs routing algorithms for bidirectional information flow between layers through iterative processes. In contrast, the cluster routingtechnique utilizes a non-iterative process and can outperform state-of-the-art models with fewer parameters, while preserving the part–whole relationship and demonstrating robust generalization to novel viewpoints. This paper aims to further analyze and clarify this concept, providing insights that allow users to implement the cluster routing technique efficiently. Additionally, we expand the technique and propose variations based on the routing principle of achieving consensus among votes in distinct clusters. In some cases, these variations have the potential to enhance and boost the cluster routing performance while utilizing similar memory and computing resources.

Keywords: data-dependent routing; capsnet; capsule network

Citation: Pham, H.; Cheng, S. Non-Iterative Cluster Routing: Analysis and Implementation Strategies. *Appl. Sci.* **2024**, *14*, 1706. https://doi.org/10.3390/app14051706

Academic Editor: Nektarios A. Valous

Received: 12 December 2023
Revised: 8 February 2024
Accepted: 13 February 2024
Published: 20 February 2024

1. Introduction

A Capsule Network, often referred to as CapsNet, is an advanced type of neural network that employs neuron clusters known as "capsules". Unlike traditional Convolutional Neural Networks (CNNs), which output scalar values, these capsules produce vector outputs. These vectors represent not only the probability of a feature's existence but also its instantiation parameters, such as pose (position, size, orientation), deformation, texture, and so on. This richer representation allows the network to capture and maintain spatial relationships between features more effectively than traditional methods. CapsNet introduces a unique mechanism, known as "routing-by-agreement", which replaces the pooling layers found in conventional CNNs. This routing process enables capsules at one level to send their outputs to higher-level capsules only if there is a strong agreement (i.e., high probability) that the higher-level capsule's entity is present in the input. This agreement is determined according to the dot product between the output of a lower-level capsule and the predicted output of a higher-level capsule, iteratively refined through a routing process. This architecture ensures that, during forward propagation, information flows through the network in a way that preserves spatial relationships, making it inherently more capable of handling variations in viewpoint, scale, and rotation without the need for extensive data augmentation [1].

Capsule networks aim to address some fundamental limitations of CNNs, especially in terms of preserving spatial hierarchies between features within an image. As CNNs rely on the scalar output of neurons within layers for feature detection and representation, they sometimes fail to recognize objects captured from different viewpoints if they are not covered in the training data [2,3]. CapsNets, in contrast, can better preserve the pose information (position and orientation) of features, thus making them more robust to variations in the input data [1,4,5].

There are several main distinctions between CapsNets and CNNs. First, CapsNets use a vector or group of neurons as a basic unit, whereas CNNs use a single neuron. These vectors, called capsules, can potentially represent different parts of an object. Second, while CNNs capture hierarchical features through the depth of the network, with each layer learning different levels of abstract and complex features, CapsNets model hierarchical relationships between parts and whole objects, providing more interpretable representations of learned features. Third, unlike CNNs—which lack a dedicated mechanism for routing information between layers—CapsNets use data-dependent routing to determine the flow of information between capsules, allowing for better modeling of part–whole relationships.

In classic routing procedures, a CapsNet begins with a set of primary capsules that represent low-level features extracted from the input data. Each primary capsule makes predictions (or votes) by computing an affine transformation of its output and sending their votes to the capsules of the next layer. These capsules at the higher level compute a weighted sum of the predictions received from capsules at the lower level. Routing weights are normalized through layer normalization [6] or a squashing function [1]. An iterative routing process is used to determine an agreement on how capsules at one level should connect to capsules in the next level by updating the weights [1,4]. In contrast, a non-iterative routing procedure computes the routing weights and information of capsules only once [5,7]. By simplifying the process into a single forward pass, non-iterative routing methods alleviate the computational load associated with iteration.

Unlike conventional routing methods [1,4], the cluster routing paradigm involves capsules generating vote clusters (instead of individual votes) for the subsequent layer's capsules [7]. Each vote cluster consists of multiple votes, with each vote potentially originating from a distinct capsule in the previous layer. The proximity of votes within a cluster signifies the information extracted from the same part of an object from capsules in previous layers. Consequently, the variance within a vote cluster can serve as an indicator of the confidence in the encoded information that the vote clusters represent. This suggests that vote clusters with lower variance are more reliable in encoding information related to a specific part of an object. Thus, greater weights are assigned to centroids originating from clusters with lower variance.

We offer a comprehensive explanation of the definitions and terminologies presented in the existing literature [7]. Our objective is to minimize confusion and prevent misinterpretations surrounding the cluster routing technique, thereby enhancing its utilization across different scenarios.

Moreover, we carry out extensive experiments to investigate the impact of agreement indicators and hyperparameters. Our goal is to determine whether variance can serve as the sole dependable agreement indicator, or if increasing the number of votes per cluster invariably results in better performance.

Expanding upon the non-iterative cluster routing concept [7], we propose two variations of cluster routing: one features a streamlined routing process with the potential to reduce the number of parameters while maintaining performance levels comparable to those of existing methods, and the other is designed to effectively handle an increase in the number of votes per cluster, thus improving the overall efficiency of cluster routing.

Empirical evaluations across several data sets show that our proposed methods utilizing cluster routing consistently yield higher accuracy, when compared to existing capsule networks. Moreover, our capsule network maintains the beneficial attributes of previous CapsNet versions, including strong generalization to images taken from new viewpoints.

The contributions of our work are outlined in the following:

- The definitions and examining the implementations within the cluster routing literature are clarified to support the effective deployment of cluster routing techniques.
- Comprehensive experiments are performed to investigate the impacts of agreement indicators.
- Two novel variations of cluster routing are proposed [7] based on the routing principle of achieving consensus among votes of clusters.

2. Related Works

2.1. Agreement Routing in Capsule Network

CapsNets play a crucial role in feature encoding. They transform the extracted features into capsules, which are vectors capable of characterizing parts of an object, unlike the singular units used in conventional CNNs. These capsules possess the ability to represent more intricate aspects of an object, such as its pose, orientation, and texture style. The capsule system proposed in the dynamic routing paper titled "Dynamic Routing Between Capsules" [1] employs a routing-by-agreement mechanism. This mechanism iteratively determines its predictions based on the agreement between lower- and higher-level capsules, which is achieved through adjusting the weights connecting them. Inspired by the ideas presented in this paper, numerous subsequent research papers have explored similar concepts with the aim of enhancing the capsule network while preserving the critical notion of the part–whole relationship. These endeavors have led to diverse approaches and innovations in the field. The matrix capsules with EM routing approach [4] uses high-dimensional coincidence filtering and a fast iterative process to determine the routing weights, resulting in better performance and higher robustness to adversarial attacks than baseline CNNs. RS-CapsNet [8] integrates *Res2Net* and *Squeeze-and-Excitation* blocks to extract multi-scale features, emphasizes useful information, and employs linear combinations between capsules in order to enhance object representation while reducing the capsule count. Cluster routing [7] uses variance as the fundamental indicator of agreement and confidence in the information encoded within the vote cluster. Group normalization [9] also utilizes the mean and standard deviation of a group. It splits the output channels of a convolutional layer into several groups and normalizes the features within each group according to the mean and standard deviation. However, as it does not use routing weights based on the agreement between capsules (data-dependent routing) to produce the input for the next capsule layer, as in cluster routing, it is not classified as a routing algorithm.

CapsNets can outperform conventional CNNs in various applications, utilizing fewer parameters [10–12]. This advantage is particularly notable in medical image processing, where the main obstacles include detecting small lesions and overcoming class imbalances. Unlike CNNs, which often require substantial amounts of labeled data (that may be scarce in medical contexts), CapsNets can achieve similar levels of performance with a smaller data set [13,14]. CapsNets employ capsules that encapsulate richer feature vectors, unlike CNNs that use scalar neurons, making them more effective in addressing these challenges [13]. Furthermore, CapsNets are superior in maintaining part–whole relationships and geometric details, significantly improving their performance in medical segmentation tasks [11,12,15,16].

AI-generated deepfakes, including face-swapping videos and images, have been proliferating across the internet, driven by significant advancements in graphics processing units and AI algorithms. These technologies enable individuals to effortlessly create manipulated and unethical media. In areas such as deepfake detection, where novel and unforeseeable attacks are frequent, the strong generalization capability of capsules becomes vital. The nature of deepfake technology allows attackers to continuously develop new methods to bypass detection systems, making it imperative for defense mechanisms to possess the ability to generalize from known attacks to novel ones effectively. CapsNets are particularly suited to this task, due to their ability to understand the underlying structure of the data in a way that mirrors human visual perception. This understanding includes recognizing when an image or video deviates from the norm in a manner that suggests manipulation, even if the specific technique used for manipulation has not been encountered by the system before. A notable application of CapsNets in this context is Capsule-Forensic, which has been shown to be effective in identifying altered or synthetically produced images and videos [17,18]. The efficacy of CapsNets in this application stems from their unique ability to encode hierarchical relationships between objects and their components, including detailed pose information. This makes it an invaluable tool in the fight against the unethical use of AI for media manipulation [19].

2.2. Attention Routing

The ideas behind CapsNets share similarities with those of attention mechanisms, initially introduced in transformers [20]. Attention between capsules [21] replaces dynamic routing with a convolutional transform and attention routing, thus utilizing fewer parameters. The inclusion of a dual attention mechanism after the convolution layer and primary capsules in [22] enhanced the performance of the CapsNets. The use of a self-attention mechanism in [23] allowed for alternative non-iterative routing, efficiently reducing the number of parameters while maintaining effectiveness.

3. Methods

3.1. Notation and Terminology

Several terms and definitions are used in the relevant literature, defined as follows:

- **v**, *Vote*: A vote is the prediction of a capsule, resulting from the multiplication of a weight matrix with a capsule.
- **D**, *Depth or Length of a Capsule*: The number of convolutional units in a capsule.
- **u**, *Capsule*: A capsule with length or depth of **D**.
- **K**, *Number of Votes per Cluster*: Each cluster has **K** votes.
- **N**, *Number of Clusters*: The number of vote clusters in each channel of a capsule layer.
- **W**, *Weight Matrix*: A weight matrix is multiplied with a capsule to create a vote **v**. There are **K** matrices to produce **K** votes in a cluster.
- **C**, *Number of Channels*: Each capsule layer has **C** channels.
- **G**, *Number of Groups*: Each group has a number of clusters.

In cluster routing, a capsule layer has **C** channels, each channel has **N** clusters, and each cluster has **K** votes.

3.2. Original Non-Iterative Cluster Routing

Cluster routing [7] generates clusters of votes through a non-iterative approach, where the standard deviation is employed as the fundamental indicator of agreement and confidence for the information encoded within the vote cluster. Clusters with higher variance indicate lower confidence in their encoded information or decision; consequently, they are weighted less and passed to the next layer. The agreement vector is computed based on the standard deviation of the cluster's individual votes. The centroids or means of the vote clusters are weighted with their corresponding agreement vectors. The weighted sum of these centroids integrates and aggregates the information from all vote clusters across the capsules in the current layer, which is then normalized and transmitted to the next layer. By leveraging agreement and weighted summation, cluster routing can efficiently and dynamically route information flow between each layer of capsules in a single pass. A vote cluster $\hat{\mathbf{u}}_i^{\{1,\cdots,K\}}$ can be generated by multiplying a capsule $\mathbf{u}_i$ with each weight matrix $\mathbf{W}$ of a weight cluster $\mathbf{W}_i^{\{1,\cdots,K\}}$,

$$\hat{\mathbf{u}}_i^k = \mathbf{W}_i^k \mathbf{u}_i. \tag{1}$$

To enhance the receptive fields of each vote, the neighborhood of $\mathbf{u}_i$ in Equation (1) is expanded to 3×3 in practice. $\mathbf{u}_i$ becomes the concatenation $N(\mathbf{u}_i) \in \mathbb{R}^{9D}$ of its neighborhood. Then, the kth vote in the cluster is produced as follows,

$$\hat{\mathbf{u}}_i^k = \mathbf{W}_i^k N(\mathbf{u}_i). \tag{2}$$

A vote cluster $\hat{\mathbf{u}}_i^{\{1,\cdots,K\}}$ will send its centroid or mean $\mathbf{m}_i = \frac{1}{K}\sum_{k=1}^{K} \hat{\mathbf{u}}_i^k$ to the next layer, with weighting determined by the agreement vector $\mathbf{a}_i$. The computation of $\mathbf{a}_i$ is outlined as follows,

$$\mathbf{a}_i = -\log(\sigma_i), \tag{3}$$

where

$$\sigma_i = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(\hat{\mathbf{u}}_i^k - \mathbf{m}_i^k) \times (\hat{\mathbf{u}}_i^k - \mathbf{m}_i^k)} \tag{4}$$

The centroids (or means) of these vote clusters are weighted summed as follows,

$$\mathbf{s} = \sum_{i=1}^{C} \mathbf{c}_i \times \mathbf{m}_i, \tag{5}$$

where

$$\mathbf{c}_i = \frac{\exp(\mathbf{a}_i)}{\sum_{j=1}^{C}\exp(\mathbf{a}_j)} \tag{6}$$

Each of the **C** capsule channels contributes a vote cluster to the capsules in the next layer, making the number of vote clusters (**N**) equivalent to the number of channels (**C**) in the multiple-channel version. Layer normalization [6] is applied to **s** to produce the output of the capsule layer. The computational task in Equation (2) can be efficiently executed through utilization of PyTorch's pre-existing optimized convolutional operations [24], similar to the technique employed in [21].

Notably, this mechanism utilizes significantly fewer parameters than both attention routing [21] and dynamic routing [1]. The absence of iterations in the process enhances its computational efficiency and reduces training resource requirements.

3.3. Cluster Routing Implementation

3.3.1. Weighting Computation

Feature maps were extracted from the original image through a convolutional layer to form primary capsules. This layer is a convolutional capsule layer with **C** channels of convolutional capsules. Each primary capsule comprises **D** convolutional units with a 3×3 kernel and a stride of 1. To create a vote, cluster routing utilizes a weight matrix for element-wise multiplication with a capsule. Capsules within the same channel can share weights to produce a vote, where each capsule has dimensions of $1 \times 1 \times$ D. However, in cluster routing, the surrounding neighborhood is included to increase the receptive field for producing a vote, resulting in the weight dimension of a capsule becoming $3 \times 3 \times$ D.

To handle the weight multiplication process, we leverage an optimized function in PyTorch named `conv2d`. There are two approaches to produce votes from **C** separate channels in the primary capsule layer.

First, a single `conv2d` layer can be used to generate votes from all **C** channels, employing group convolution through the `conv2d` function in PyTorch. The `groups` parameter in `conv2d` controls the connections between inputs and outputs; for example, if `groups` $= 1$, all inputs are convolved to all outputs. In cluster routing with $C = 3$, we can use `groups` $= 3$ to consider three convolutional layers side by side, creating three capsule channels. Each layer sees one-third of the input tensor and produces one-third of the output, which are subsequently concatenated.

Second, the feature maps can be segmented into individual channels, with each channel employing its own `conv2d` layer to generate votes. It is important to note that each channel utilizes a distinct `conv2d` layer for the generation of votes, which is currently implemented in the code.

In iterative routing [1,4], the routing weights are computed through an iterative process. The weight update frequency can have significant effects on the performance [1]. However, in our non-iterative approach, the weight only needs to be computed once per epoch based on the agreement vectors. Hence, the weight update frequency does not impact the performance. In fact, without iteration, our method can reduce the training time to approximately 15%.

3.3.2. Agreement Vector

In cluster routing, the weighting coefficients c_i in Equation (6) can be computed efficiently using the `softmax` function in Pytorch. However, when we substitute Equation (3) into Equation (6), the c_i can be simplified into the following form,

$$\mathbf{c}_i = \frac{\exp(\mathbf{a}_i)}{\sum_{j=1}^{C} \exp(\mathbf{a}_j)} = \frac{\frac{1}{\sigma_i}}{\sum_{j=1}^{C} (\frac{1}{\sigma_j})} \tag{7}$$

where σ_i is the standard deviation of the vote cluster.

The agreement vector in Equation (6) now can be redefined as $\mathbf{a}_i = \frac{1}{\sigma_i}$. This can be explained by stating that the agreement is inversely proportional to the standard deviation of the vote cluster: with a higher standard deviation, less agreement exists between the votes, resulting in less weight being assigned to the centroid of the cluster.

3.3.3. Cluster Routing with Single Channel

In cluster routing, optimal performance is typically achieved through utilization of the single-channel version, as illustrated in Table 1. Technically, this approach can be summarized as creating multiple versions of the original capsule channel, increasing its dimension by one. Next, the `.view()` function is utilized to divide the additional dimension into two dimensions, **K** and **N**, as depicted in Figure 1. Subsequently, the mean and standard deviation are computed along the **K** dimension to determine the centroid and agreement vector, reducing the dimension by one. Following this, the mean tensor is multiplied with the weight and then summed to derive the output capsule channel, further reducing the dimension by one. This process involves expanding the dimension and subsequently reducing it to preserve the original dimensionality of the input capsule channel. Further details and the code implementation of this version can be found in our GitHub repository.

The averaging and summing components of this technique can be conceptualized as a type of vector dimension reduction, in which the dimensionality is reduced while preserving essential information. However, additional experiments (which are not addressed in this study) are required to validate this hypothesis.

Table 1. Comparative analysis of test error rates in the capsule networks literature [7].

Method	smallNORB		Fashion-MNIST		SVHN		CIFAR-10	
	Error (%)	Param	Error (%)	Param	Error (%)	Param	Error (%)	Param
Baseline CNN	3.76	3.30M	5.21	3.38M	3.30	3.38M	7.90	3.38M
Dynamic [1]	2.7	8.2M	-	-	4.3	≃1.8M	10.6	8.2M
EM-Routing [4]	1.8	310K	-	-	-	-	11.9	≃460K
RS-CapsNet [8]	-	-	6.49	5.00M	3.5	5.01M	10.19	5.01M
Attention Routing [21]	-	-	-	-	-	-	11.39	9.6M
DA-CapsNet [22]	1.74	-	6.02	-	5.18	-	14.53	-
Efficient-CapsNet [23]	2.54	151K	-	-	-	-	-	-
RVC [25]	-	-	5.87	-	2.97	-	8.3	-
Inverted dot-product attention routing [26]	-	-	-	-	-	-	14.83	560K
VB-Routing [27]	1.6	169K	5.2	172K	3.9	323K	11.2	≃323k
MoCapsNet [28]	-	-	-	-	7.00	-	27.83	-
SRM-CapsNet [29]	8.54	-	8.35	-	6.90	-	28.26	-
ER-CapsNet [30]	-	-	-	-	4.33	-	11.46	9.4M
Metric Learning [31]	-	-	9.59	-	7.24	-	26.76	9.4M

Table 1. *Cont.*

Method	smallNORB		Fashion-MNIST		SVHN		CIFAR-10	
	Error (%)	Param	Error (%)	Param	Error (%)	Param	Error (%)	Param
MASK DR [32]	-	-	6.32	1.4M	-	-	9.99	1.4M
ML-CapsNet [33]	-	-	7.35	-	-	-	14.28	-
PT-CapsNet [34]	-	-	7.77	-	-	-	8.79	-
NASCaps [35]	-	-	6.13	-	3.41	-	23.54	-
DenseCaps [36]	-	-	-	-	4.01	-	10.59	16M
Cluster Routing [7]								
CM [1]*456* (C4K5D6)	2.98 ± 0.24	150K	5.17 ± 0.07	146K	3.94 ± 0.07	154K	12.16 ± 0.30	154K
CM458 (C4K5D8)	3.09 ± 0.19	246K	5.02 ± 0.04	240K	3.63 ± 0.11	252K	11.11 ± 0.09	252K
CM4816 (C4K8D16)	1.92 ± 0.12	1.32M	4.84 ± 0.07	1.30M	3.56 ± 0.07	1.34M	8.55 ± 0.12	1.34M
CM4824 (C4K8D24)	1.95 ± 0.12	2.87M	**4.64** ± 0.03	2.84M	3.48 ± 0.14	2.89M	7.89 ± 0.11	2.89M
CS [2] (N8K8D32)	**1.57** ± 0.13	2.53 M	4.68 ± 0.01	2.51M	**3.37** ± 0.03	2.55M	**7.37** ± 0.06	2.55M
Ours								
V1M [3]*456* (K20D6G1)	3.32 ± 0.46	150K	5.36 ± 0.07	146K	3.74 ± 0.03	154K	10.78 ± 0.34	154K
V1M458 (K20D8G1)	3.11 ± 0.41	246K	5.29 ± 0.08	240K	3.71 ± 0.06	252K	10.21 ± 0.24	252K
V1M4816 (K32D16G4)	2.67 ± 0.3	1.32M	5.06 ± 0.04	1.30M	3.62 ± 0.14	1.34M	8.22 ± 0.3	1.34M
V1M4324 (K12D24G8)	**1.95** ± 0.43	1.11M	4.95 ± 0.03	1.09M	**3.48** ± 0.08	1.11M	**7.83** ± 0.12	1.11M
V1M4824 (K32D24G8)	2.1 ± 0.22	2.87M	4.86 ± 0.02	2.84M	3.73 ± 0.14	2.89M	7.53 ± 0.11	2.89M
V2S [4] (N4K2N4K2D32)	**1.46** ± 0.11	2.53 M	**4.63** ± 0.07	2.51M	**3.21** ± 0.08	2.55M	**7.33** ± 0.03	2.55M

[1] Cluster routing with multiple channels. [2] Cluster routing with single channel. [3] First variation with multiple channels. [4] Second variation with single channel.

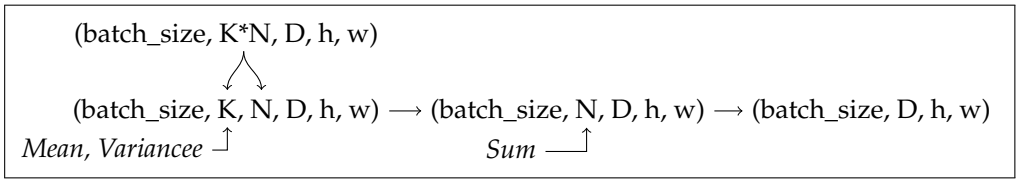

Figure 1. Single-channel quick implementation of cluster routing.

3.3.4. Cluster Routing with Multiple Channels

Cluster routing can also be implemented with multiple channels. Each capsule layer consists of **C** channels and, within each channel, there are **N** clusters. As each channel contributes a vote cluster, there are **C** vote clusters per channel, making **N** equal to **C** in this case. For simplicity, this variation involves multiple channels, with each channel representing the single-channel version of cluster routing, as previously presented.

3.4. Proposed Routing Variations

3.4.1. Our First Routing Variation

This deviated version originates from the multiple-channel version of cluster routing. Nevertheless, it remains compatible with the single-channel version, and we adhered to the agreement based on standard deviation. However, we omit the step of summation over the weighted centroids, thereby eliminating the need for the hyperparameter **N**, representing the number of clusters per channel. The process is illustrated in Figure 2.

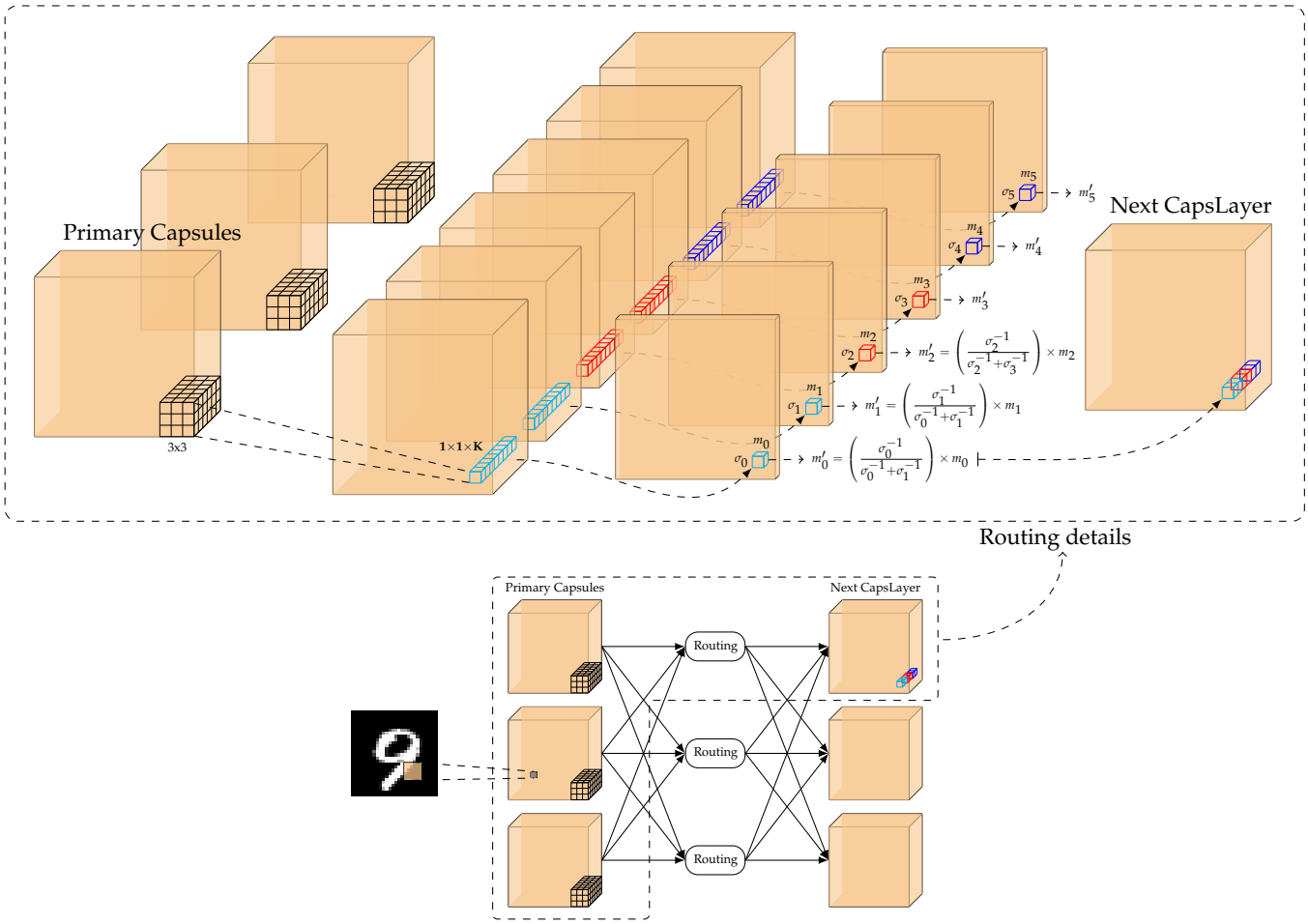

Figure 2. Our first routing variation with multiple channels. The bottom of the figure demonstrates the connections between different capsule layers. At the top of the figure, **3** × **3** primary capsules are multiplied with each weight matrix of a weight cluster comprising **K** matrices $\mathbf{W}_i^{\{1,\cdots,K\}}$ to generate a vote cluster $\mathbf{u}_i^{\{1,\cdots,K\}}$. This vote cluster consists of **K** votes, depicted as a 1 × 1 × **K** block (colored cyan) in the right corner of the front middle block. Each cluster sends its weighted centroid $\mathbf{m}_i$ based on the agreement vector $\mathbf{a}_i$ to the next capsule layer. The agreement vector is the result of comparing the standard deviation of each vote cluster with other vote clusters within their smaller respective group, distinguished using different colors, where the number of groups is denoted as **G**. In the current scenario, there are three groups (G = 3), each with two centroids (denoted as $\boldsymbol{m_0}$ and $\boldsymbol{m_1}$). These centroids are weighted using their corresponding standard deviations $\boldsymbol{\sigma_0}$ and $\boldsymbol{\sigma_1}$, resulting in transformed values (labeled as $\boldsymbol{m_0'}$ and $\boldsymbol{m_1'}$ for the cyan-colored group). This procedure is replicated for the other (red and blue) groups. The resultant weighted centroids are concatenated together to form the next capsule channel layer.

The agreement vector $\mathbf{a}_i$ in Equation (3) is simplified as the reciprocal of its standard deviation, as follows:

$$\mathbf{a}_i = \frac{1}{\sqrt{\frac{1}{K}\sum_{k=1}^{K}(\hat{\mathbf{u}}_i^k - \mathbf{m}_i^k)^2}} \tag{8}$$

Apart from the weighted sum in Equation (5), the centroids of these vote groups are only weighted and fast-forwarded to the next layers, as follows,

$$\mathbf{s} = \mathbf{c}_i \times \mathbf{m}_i, \tag{9}$$

where,

$$\mathbf{c}_i = \frac{\mathbf{a}_i}{\sum_{j=1}^{C} \mathbf{a}_j} \tag{10}$$

Then, layer normalization [6] is applied to the weighted centroids **s**. The fast-forward mechanism, when experimentally applied, exhibited comparable performance with fewer parameters than the weighted sum approach used in cluster routing for the *SVHN* and *CIFAR-10* data sets, as illustrated in Table 1. This method has the potential to reduce the number of parameters while preserving network performance.

3.4.2. Our Second Routing Variation

While conducting extensive experiments with various hyperparameters in cluster routing, we observed that increasing the number of votes per cluster (denoted as **K**) in both single- and multiple-channel versions can lead to a decrease in performance, as demonstrated in Figures 3 and 4, respectively. Therefore, we introduce a second variation of the routing mechanism to mitigate the impact of increasing **K** while enhancing its performance.

In Figure 5, the **K** and **N** dimensions are split only once in the original cluster routing, using the `.view()` method in PyTorch. This second variation suggests that the breakdown is not limited to only two times (as depicted in Figure 5) but, rather, depends on the dimension of the tensor, which must satisfy the condition $K1 \times N1 \times K2 \times N2 = K \times N$.

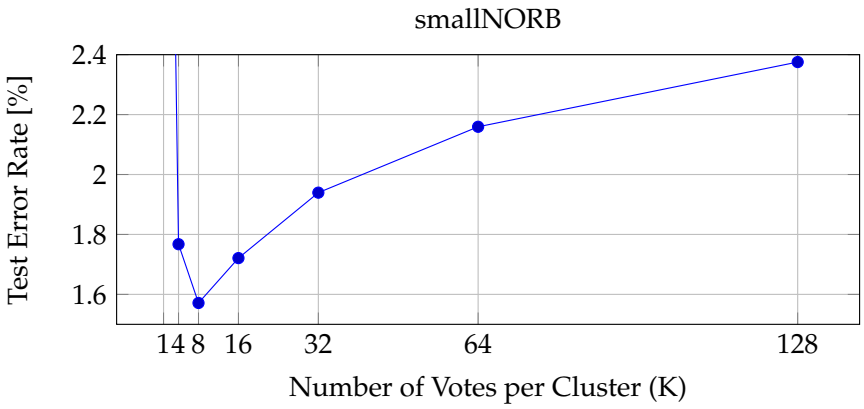

Figure 3. Performance of the single-channel version of cluster routing with **N** = 8 and **D** = 32.

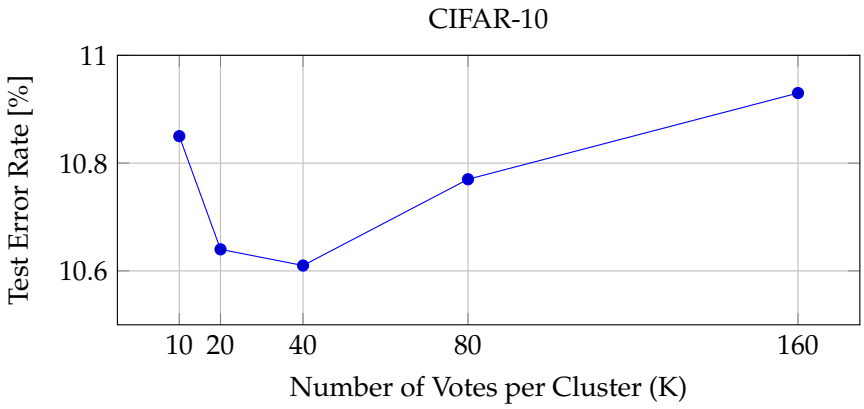

Figure 4. Performance of the multiple-channel version of cluster routing with **C** = 4 and **D** = 6.

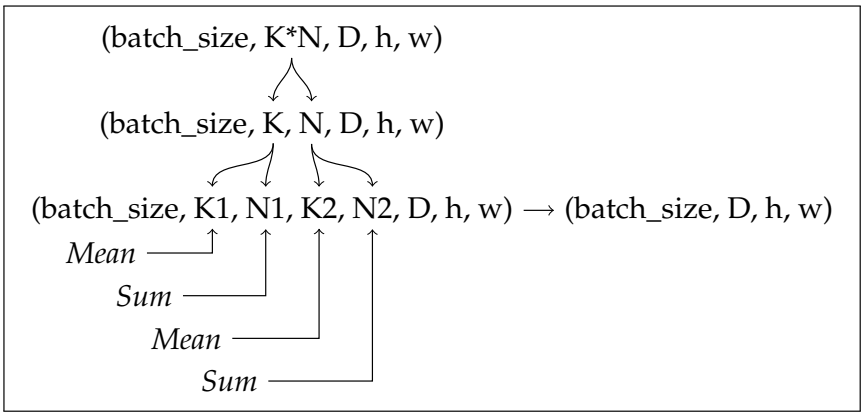

Figure 5. Second variation scheme.

This approach can be utilized to effectively manage capsule layers with large dimensions, resulting in improved performance and faster convergence, as illustrated in Figures 6 and 7.

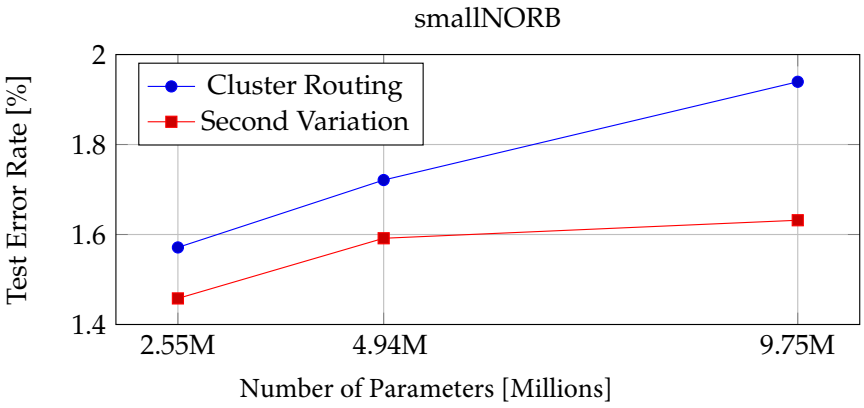

Figure 6. Performance of cluster routing and the second routing variation, along with their respective hyperparameters shown in Table A1.

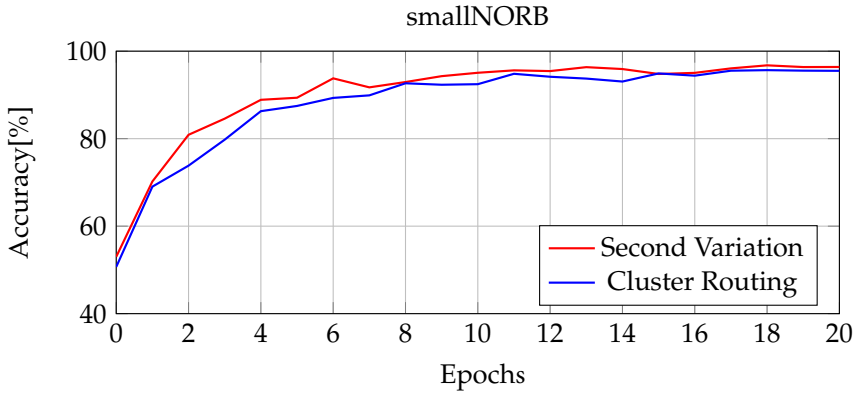

Figure 7. Training curves for cluster routing and the second routing variation.

4. Experimental Setup

In all experiments, we set the number of capsule layers to 5 and used a stride of 2 in the second and fourth layers. The models were all trained with the Stochastic Gradient Descent optimizer and cross-entropy loss. Training was performed with a batch size of 64 over 300 epochs, with an initial learning rate of 0.1 and a step decay of 0.1 at every 100 epochs. The hyperparameters for augmentation were tuned using a validation set comprising one-fifth of the training images.

The reference CNN employed for comparison was composed of five ReLU convolutional layers. Each layer incorporates ReLU activation, 256 filters, and layer normalization, resulting in a total of 3.38 million parameters.

Data Pre-Processing

For each data set, we employed brightness and contrast jitter to augment the training data. We used α and β to control the contrast and brightness $f(x)$ of a pixel at position x by $g(x) = \alpha f(x) + \beta$. Specifically, random brightness and contrast were applied with a factor of 0.2. The value of α was randomly chosen from a uniform distribution in [0.8, 1.2], and the value of β from $[-0.2\frac{1}{N_x}\sum_x f(x), 0.2\frac{1}{N_x}\sum_x f(x)]$, with N_x denoting the total number of pixels and $\frac{1}{N_x}\sum_x f(x)$ denoting the mean pixel value.

The *smallNORB* [37] data set contains 24,300 images at 96 × 96 resolution, classified into 5 classes. Pre-processing, as per [4], involved down-sampling to 48 × 48, normalization and, for training, a contrast factor of 0.2, random shifts of 0.2, padding to 56 × 56, and random cropping to 32 × 32. For testing, images were centrally cropped to 32 × 32.

The *Fashion-MNIST* [38] data set contains 70,000 images with a resolution of 28 × 28 pixels, categorized into 10 classes. The data set was split into training and testing sets with a ratio of 6:1. During training, various data augmentation techniques were applied to the images, including random brightness adjustments, random flips with a 0.5 probability, and contrast adjustments with a factor of 0.2. Additionally, the images were padded to a size of 36 × 36 pixels, then randomly cropped to a final size of 32 × 32 pixels.

The *CIFAR-10* [39] data set consists of a collection of 60,000 color images, each measuring 32 × 32 pixels, and is divided into 10 different classes, with each class representing a distinct object or category. Each class contains 6000 images. The training and testing sets were divided with a 5:1 ratio. In the training phase, random adjustments to brightness and contrast were implemented with a factor of 0.2. Padding to achieve a size of 40 × 40 was conducted, then random cropping was applied to obtain a 32 × 32 portion. Additionally, random horizontal flips were introduced with a 0.5 probability.

The *SVHN* [40] data set comprises 10 digit classes represented as 32 × 32 real-world house numbers. The core training set, consisting of 73,257 images, was used for training, while evaluation was performed on the 26,032 test set images. Throughout training, random brightness and contrast adjustments with a factor of 0.2 were applied. Images were padded to 40 × 40, and random 32 × 32 crops were extracted for further processing.

5. Results

In Table 1, we evaluate and compare the performance of our proposed capsule network in classification tasks (although it is not exclusively confined to such tasks). We investigated its generalization to novel viewpoints and experiments through the various indicators of agreement between cluster votes. For each data set, we conducted a minimum of 8 experiments using 8 random seeds under identical conditions, and the average test error rate, standard deviation, and the number of parameters are presented in Table 1.

5.1. Comparison with the Baseline CNN

In our study, our proposed cluster routing variations demonstrated superior performance when compared to the baseline CNN in two key aspects: the number of parameters and test error rates. Notably, when tested on the smallNORB data set, which is known for its variations in viewpoint, the proposed variations achieved higher accuracy while employing fewer parameters, as illustrated in Table 1. The CNN baseline, with 3.3 million parameters, achieved a test error rate of 3.76% on the smallNORB data set, while our strategy halved the test error rate to 1.95% while using approximately one-third of the parameters (1.11 M). The proposed variation achieved the best performance on this data set, with a test error rate of 1.46%. This proves the robustness of CapsNets in handling viewpoint variations in object recognition and pose estimation tasks. This accomplishment can be attributed to the capsule network's inherent capability to retain and utilize pose

information within the features, enabling robust generalization even when confronted with images captured from viewpoints that have not been previously encountered [25] While demonstrating superior performance and robustness on the smallNORB data set, our proposed variations also exhibited improved performance with a reduction in the number of parameters across image processing tasks, including the real-world digit recognition task in the SVHN data set, as well as image classification in the Fashion-MNIST and CIFAR-10 data sets, as illustrated in Table 1. This can be useful in real-world scenarios, where objects may be partially occluded or where the viewpoint of the object may vary.

CNNs are adept at a wide array of tasks, but encounter difficulties in maintaining the spatial relationships between objects due to the incorporation of max-pooling layers [13,15,41]. These layers, which are designed for feature extraction and dimensionality reduction, inadvertently result in the loss of spatial information among objects. Consequently, CNNs may demonstrate decreased robustness in scenarios involving changes in viewpoint, scale, and rotation [14]. Capsule networks offer a solution to this issue by eliminating the need for pooling layers. Instead of relying on scalar values for feature representation—as is common in CNNs—CapsNets employ vectors or "capsules". Furthermore, they utilize a "routing-by-agreement" mechanism instead of pooling, which allows capsules to collaboratively decide on the existence of higher-level features [10]. This capability to preserve spatial relationships renders CapsNets particularly beneficial in the field of medical image analysis for tasks such as tumor detection, semantic segmentation, and disease classification, especially considering the variability in the body shapes and sizes of patients, as well as tumor locations.

Despite their potential, CapsNets face inherent challenges; notably, they depend on a sophisticated routing algorithm that requires extensive agreement computations among capsules. In seminal CapsNet studies, such as those cited by [1,4], routing iterations have been shown to be crucial for updating capsule agreements, considerably lengthening training and inference times [10]. This issue could limit the ability of CapsNets to match the depth and scalability of CNNs [12]. Nevertheless, our novel approaches leverage a non-iterative strategy to calculate capsule agreement, effectively reducing the training duration by an average of 15%, when compared to conventional iterative methods [1,4].

5.2. Comparisons with the State-of-the-Art

For a fair comparison in Table 1, we employed identical data processing and training settings, ensuring the selection of configurations with a similar number of parameters.

Our first variation, characterized by a lower number of parameters, achieved performance comparable to that of the original cluster routing approach. Specifically, the *V1M4324* variation with fewer parameters (1.11 million in total) demonstrated a test error rate of 3.48% on the *SVHN* data set. This is comparable to that of the original cluster routing approach, which has 2.89 million parameters and a similar error rate. Similarly, in the context of the *CIFAR-10* data set, the outcomes of the *V1M4324* variation closely matched those of the original cluster routing, despite employing fewer parameters, with test error rates of 7.83% and 7.89%, respectively. However, it slightly fell short in performance on the *Fashion-MNIST* data set, exhibiting a test error rate of 4.95% with 1.1M parameters, in contrast to the original cluster routing, which achieved 4.84% with 1.3M parameters. Despite this result, this variation demonstrated the potential to achieve comparable performance with fewer parameters. Furthermore, our variation employed a simpler agreement calculation, leading to a modest reduction in training time (averaging 5%), when compared to the original cluster routing approach.

Our second variation model, *V2S*, demonstrated significant performance enhancements compared to existing capsule networks with an equivalent number of parameters to the original cluster routing, as detailed in Table 1. Our analysis revealed that an increase in the number of votes per cluster, denoted as **K**, led to a degradation in the performance of the original cluster routing in both single- and multi-channel configurations, as depicted in Figures 3 and 4, respectively. The *V2S* model addresses this issue effectively, offering

a strategic approach to manage capsule layers with significant dimensions and alleviate the negative impact of increasing **K**. In the context of the *smallNORB* data set, *V2S* not only achieved superior performance, as demonstrated in Figure 6, but also exhibited faster convergence during network training, as illustrated in Figure 7. These characteristics significantly enhance the cluster routing efficacy, underscoring the potential of the *V2S* model in improving cluster routing capabilities.

Despite nuanced variations, the two proposed routing alternatives emerge as viable options, showcasing notable strengths in differentiating data sets with diverse characteristics.

5.3. Classification across Various Viewpoints

In this section, we evaluate the generalization capabilities of our proposed methods to novel viewpoints using the *smallNORB* data set. To guarantee a fair comparison, we also retrained the cluster routing algorithm, applying the same data augmentation techniques as those used in our methods.

The network was trained on the training set with varying capture angles or azimuths (300, 320, 340, 0, 20, 40). The test set comprised different instances in the same categories, with unseen angles ranging from 60 to 280. To evaluate the network's performance under varying elevation viewpoints, we selected three smaller elevations from the training set and tested the model on six larger elevations from the test set.

From the novel azimuth viewpoint, our first variation achieved the highest accuracy of 85.27%. The multiple-channel version of the original cluster routing and our first variation demonstrated an advantage over the single-channel version. Regarding the novel elevation viewpoint, the single version of our second variation attained the highest accuracy of 84.44%, as depicted in Table 2. The performance of both variations was on par with the original cluster routing, maintaining robust generalization to novel viewpoints. In certain instances, they even exhibited a slight advantage.

Table 2. Comparison of generalization capabilities on the *smallNORB* data set across various viewpoints.

	Azimuth (%)				**Elevation (%)**			
Model	**CS**	**CM**	**V2S**	**V1M**	**CS**	**CM**	**V2S**	**V1M**
#Params	**2.55M**	**2.89M**	**2.55M**	**2.89M**	**2.55M**	**2.89M**	**2.55M**	**2.89M**
Novel View	83.31	84.89	83.84	85.27	83.79	84.25	84.44	83.71

5.4. Agreement Indicators

In cluster routing, the standard deviation of the votes within a cluster serves as a metric for assessing the consensus among votes surrounding the cluster centroids. Specifically, $\frac{1}{\sigma}$ is utilized to calculate the agreement weighting for the cluster centroids in cluster routing.

We performed experiments to investigate alternative agreement indicators, including $\frac{1}{\sqrt{\sigma}}$, $\frac{1}{\sigma^2}$, $\frac{1}{\sigma^3}$, and the mean of absolute difference. However, no significant variations in network performance were observed. Additionally, we explored alternative indicators, such as the L1 norm (Manhattan norm) and the L2 norm (Euclidean norm), utilizing a moving 3D kernel for norm calculations. Despite these efforts, such alternatives led to a decrease in performance.

The summarized performance results for these investigations are provided in Table 3. Based on these findings, we concluded that $\frac{1}{\sigma}$ stands out as the optimal choice for the agreement indicator, due to its simplicity and efficiency.

Table 3. Performance of *second variation* on CIFAR-10 with diverse agreement indicators.

$\sqrt{\sigma}$	σ	σ^2	σ^3	**MAD** *	**L1 Norm**	**L2 Norm**
10.92 ± 0.33	10.78 ± 0.34	10.75 ± 0.37	11.05 ± 0.54	11.26 ± 0.39	34.41 ± 0.16	29.44 ± 0.25

* Mean of absolute difference.

6. Conclusions

This research aimed to clarify the key definitions and concepts integral to cluster routing, with an emphasis on demystifying the terminologies and principles underlying the original cluster routing algorithm. Additionally, we provided a comprehensive guide for the implementation of different versions of the original cluster routing algorithm. Our goal was to streamline the deployment process, making it easier for users to adopt and effectively utilize the cluster routing approach.

We conducted extensive experiments to assess the limitations associated with increasing the number of votes per cluster and explore alternative agreement indicators for cluster routing. Our findings indicated that the standard deviation of the votes within a cluster stands out as the optimal choice for the agreement indicator, due to its simplicity and efficiency. Additionally, we observed that increasing the number of votes per cluster, denoted as **K**, led to decreased performance in the original cluster routing for both single- and multiple-channel versions.

Furthermore, we proposed two innovative cluster routing variations, which were designed to enhance its capabilities. Our first variation attempts to simplify the original cluster routing computation process through omitting the summation over weighted centroids, thereby reducing the training time. In some cases, it achieved superior performance while utilizing fewer parameters. This simplification is beneficial, as computational complexity and memory requirements pose challenges for constructing deeper and larger capsule networks. Our second variation addresses the impact of increasing the number of votes per cluster, denoted as **K**, in the original cluster routing. This variation enables us to manage capsule layers with substantial dimensions and mitigate the effects of increasing the number of votes per cluster, leading to improved performance when compared to the original cluster routing approach. Additionally, it offers the advantage of faster convergence throughout the network training phase.

Our proposed novel methods improved the effectiveness of cluster routing and appear promising for numerous applications. Future research will aim to evaluate the applicability of our variations in various fields, including medical image analysis and deepfake detection.

Author Contributions: Supervision, S.C.; Writing—original draft, H.P.; Writing—review & editing, H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Vice President for Research and Partnerhsips, the Data Institute for Societal Challenges, and the Stephenson Cancer Center at the University of Oklahoma through the 2021 DISC/SCC Seed Grant Award.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All research data used in this study were sourced from publicly available data on public websites. The source codes developed for this study are accessible at https://github.com/hngpham/cluster_routing_variations starting from 16 March 2023.

Acknowledgments: The authors extend their gratitude to Zhihao Zhao for valuable discussions. Additionally, the authors would like to acknowledge the partial provision of computational resources for this project by the OU Supercomputing Center for Education & Research (OSCER) at the University of Oklahoma (OU).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Experimental configuration used for Figure 6.

Parameter (Million)	Cluster Routing (K, N, D)	Second Variation (K1, N1, K2, N2, D)
2.55M	(8, 8, 32)	(4, 2, 4, 2, 32)
4.94M	(16, 8, 32)	(8, 2, 4, 2, 32)
9.75M	(32, 8, 32)	(4, 4, 4, 4, 32)

References

1. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. In Proceedings of the 31st Conferenceon Neural Information Processing Systems (NIPS2017), Long Beach, CA, USA, 4–9 December 2017; pp. 3856–3866.
2. Engstrom, L.; Tsipras, D.; Schmidt, L.; Madry, A. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv* **2017**, arXiv:1712.02779.
3. Alcorn, M.A.; Li, Q.; Gong, Z.; Wang, C.; Mai, L.; Ku, W.S.; Nguyen, A. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 4845–4854.
4. Hinton, G.E.; Sabour, S.; Frosst, N. Matrix capsules with EM routing. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
5. Hinton, G.E.; Krizhevsky, A.; Wang, S.D. Transforming auto-encoders. In Proceedings of the International Conference on Artificial Neural Networks, Espoo, Finland, 14–17 June 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 44–51.
6. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
7. Zhao, Z.; Cheng, S. Capsule networks with non-iterative cluster routing. *Neural Netw.* **2021**, *143*, 690–697. [CrossRef] [PubMed]
8. Yang, S.; Lee, F.; Miao, R.; Cai, J.; Chen, L.; Yao, W.; Kotani, K.; Chen, Q. RS-CapsNet: An advanced capsule network. *IEEE Access* **2020**, *8*, 85007–85018. [CrossRef]
9. Wu, Y.; He, K. Group normalization. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
10. Ribeiro, F.D.S.; Duarte, K.; Everett, M.; Leontidis, G.; Shah, M. Learning with capsules: A survey. *arXiv* **2022**, arXiv:2206.02664.
11. Zade, A.A.T.; Aziz, M.J.; Masoudnia, S.; Mirbagheri, A.; Ahmadian, A. An improved capsule network for glioma segmentation on MRI images: A curriculum learning approach. *Comput. Biol. Med.* **2022**, *148*, 105917.
12. Wang, L.; Tang, M.; Hu, X. Evaluation of grouped capsule network for intracranial hemorrhage segmentation in CT scans. *Sci. Rep.* **2023**, *13*, 3440. [CrossRef]
13. Zhang, Z.; Ye, S.; Liao, P.; Liu, Y.; Su, G.; Sun, Y. Enhanced capsule network for medical image classification. In Proceedings of the 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Montreal, BC, Canada, 20–24 June 2020; IEEE: New York, NY, USA, 2020; pp. 1544–1547.
14. Akinyelu, A.A.; Zaccagna, F.; Grist, J.T.; Castelli, M.; Rundo, L. Brain tumor diagnosis using machine learning, convolutional neural networks, capsule neural networks and vision transformers, applied to MRI: A survey. *J. Imaging* **2022**, *8*, 205. [CrossRef]
15. Tran, M.; Vo-Ho, V.K.; Quinn, K.; Nguyen, H.; Luu, K.; Le, N. CapsNet for medical image segmentation. In *Deep Learning for Medical Image Analysis*; Elsevier: Amsterdam, The Netherlands, 2024; pp. 75–97.
16. Avesta, A.; Hui, Y.; Aboian, M.; Duncan, J.; Krumholz, H.; Aneja, S. 3D capsule networks for brain image segmentation. *Am. J. Neuroradiol.* **2023**, *44*, 562–568. [CrossRef]
17. Bhagtani, K.; Yadav, A.K.S.; Bartusiak, E.R.; Xiang, Z.; Shao, R.; Baireddy, S.; Delp, E.J. An overview of recent work in media forensics: Methods and threats. *arXiv* **2022**, arXiv:2204.12067.
18. Nguyen, H.; Yamagishi, J.; Echizen, I. Use of a capsule network to detect fake images and videos. *arXiv* **2019**, arXiv:1910.12467.
19. Mehra, A.; Spreeuwers, L.J.; Strisciuglio, N. Deepfake Detection using Capsule Networks and Long Short-Term Memory Networks. In Proceedings of the VISIGRAPP, Virtual Event, 8–10 February 2021.
20. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
21. Choi, J.; Seo, H.; Im, S.; Kang, M. Attention routing between capsules. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Seoul, Republic of Korea, 27–28 October 2019.
22. Huang, W.; Zhou, F. DA-CapsNet: Dual attention mechanism capsule network. *Sci. Rep.* **2020**, *10*, 11383. [CrossRef] [PubMed]
23. Mazzia, V.; Salvetti, F.; Chiaberge, M. Efficient-capsnet: Capsule network with self-attention routing. *Sci. Rep.* **2021**, *11*, 14634. [CrossRef]
24. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in pytorch. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS2017), Long Beach, CA, USA, 4–9 December 2017.
25. Xie, N.; Wan, X. Residual Vector Capsule: Improving Capsule by Pose Attention. *IEEE Access* **2021**, *9*, 129626–129634. [CrossRef]

26. Tsai, Y.H.H.; Srivastava, N.; Goh, H.; Salakhutdinov, R. Capsules with Inverted Dot-Product Attention Routing. *arXiv* **2020**, arXiv:2002.04764.
27. Ribeiro, F.D.S.; Leontidis, G.; Kollias, S.D. Capsule Routing via Variational Bayes. In Proceedings of the PAAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 3749–3756.
28. Gugglberger, J.; Peer, D.; Rodríguez-Sánchez, A. Momentum Capsule Networks. *arXiv* **2022**, arXiv:2201.11091.
29. Hollósi, J.; Ballagi, Á.; Pozna, C.R. Simplified Routing Mechanism for Capsule Networks. *Algorithms* **2023**, *16*, 336. [CrossRef]
30. Alaoui-Elfels, E.; Gadi, T.; ER-Caps: ELU Residual Capsule Network for Complex Images Classification. *Int. J. Intell. Eng. Syst.* **2023**, *16*. [CrossRef]
31. Ohta, N.; Kawai, S.; Nobuhara, H. Capsule Network Extension Based on Metric Learning. *J. Adv. Comput. Intell. Intell. Inform.* **2023**, *27*, 173–181. [CrossRef]
32. Chen, J.; Liu, Z. Mask dynamic routing to combined model of deep capsule network and u-net. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 2653–2664. [CrossRef]
33. Noor, K.T.; Robles-Kelly, A.; Kusy, B. A Capsule Network for Hierarchical Multi-label Image Classification. In Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Montreal, BC, Canada, 26–27 August 2022; Springer: Cham, Switzerland, 2022; pp. 163–172.
34. Pan, C.; Velipasalar, S. PT-CapsNet: A novel prediction-tuning capsule network suitable for deeper architectures. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 11996–12005.
35. Marchisio, A.; Massa, A.; Mrazek, V.; Bussolino, B.; Martina, M.; Shafique, M. NASCaps: A framework for neural architecture search to optimize the accuracy and hardware efficiency of convolutional capsule networks. In Proceedings of the 39th International Conference on Computer-Aided Design, Virtual Event, 2–5 November 2020; pp. 1–9.
36. Sun, G.; Ding, S.; Sun, T.; Zhang, C.; Du, W. A novel dense capsule network based on dense capsule layers. *Appl. Intell.* **2022**, *52*, 3066–3076. [CrossRef]
37. LeCun, Y.; Huang, F.J.; Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 27 June–2 July 2004; IEEE: New York, NY, USA, 2004; Volume 2, pp. II–104.
38. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
39. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report TR-2009; University of Toronto: Toronto, ON, Canada, 2009.
40. Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A.Y. Reading digits in natural images with unsupervised feature learning. In Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain, 12–17 December 2011.
41. Zhang, H.; Li, Z.; Zhao, H.; Li, Z.; Zhang, Y. Attentive Octave Convolutional Capsule Network for Medical Image Classification. *Appl. Sci.* **2022**, *12*, 2634. [CrossRef]