applied sciences

MDPI

Article

DE-CapsNet: A Diverse Enhanced Capsule Network with Disperse Dynamic Routing

Bohan Jia and Qiyu Huang *

School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; bohan_ieee@163.com

* Correspondence: iyu@sjtu.edu.cn

Received: 24 December 2019; Accepted: 22 January 2020; Published: 29 January 2020

Abstract: Capsule Network (CapsNet) is a methodology with good prospects in visual tasks, since it can keep a stronger relationship of spatial information than Convolutional Neural Networks (CNNs). However, the current Capsule Network do not provide performance as expected on several benchmark data sets with complex data and backgrounds. Inspired by the multiple capsules of Diverse Capsule Network (DCNet++) and the Spatial Group-wise Enhance (SGE) mechanism, we propose the Diverse Enhanced Capsule Network (DE-CapsNet), a hierarchical architecture which uses residual convolutional layers and the position-wise dot product to build diverse enhanced primary capsules with various scales of images for complex data. The architecture adopts the Sigmoid function in a dynamic routing algorithm to get a more uniform distribution of routing coefficients which obviously distinguishes the assignment probabilities between capsules. DE-CapsNet achieved state-of-the-art accuracy on Canadian Institute For Advanced Research (CIFAR-10) in the Capsule Network field and provided better performance than the ensemble of seven CapsNets on Fashion-Modified National Institue of Standards and Technology database (F-MNIST) while achieving a 50.3% reduction in the number of parameters.

Keywords: Capsule Network; Diverse Enhanced Capsule Network; Convolutional Neural Networks; deep learning; disperse dynamic routing; artificial intelligence

1. Introduction

Deep networks have been successful in the tasks of image classification and object recognition. Increasing the depth of a Convolutional Neural Network (CNN) provides a substantial improvement in the performance [1]. However, if the CNN goes too deep, it can also lead to the challenges of vanishing gradient and saturated accuracy. The degradation problem can be countered by adopting Residual Networks (ResNets) [2], adding connections from the initial layers to the later layers, and by adopting Densely Connected Convolutional Networks (DenseNets) [3], adding dense connections between every other layer. However, CNNs are not robust enough to affine transformations and cannot reserve spatial relationships between features in an image. Diversion in the position of an object in the image may lead the CNN to an incorrect prediction. To overcome the abovementioned weakness, Sabor et al. [4] proposed the Capsule Network (CapsNet), which has shown huge potential compared to the conventional CNNs on multiple datasets. A capsule is a group of neurons whose activity vector can represent an object or a part of an object to extract structured features, while keeping the information of the spatial relationship at the same time. The architecture comprises one convolution layer and one fully connected capsule layer, using routing-by-agreement to achieve state-of-the-art accuracy on the Modified National Institue of Standards and Technology database (MNIST) benchmark data set and detecting overlapping digits by using reconstruction regularization. However, the performance of CapsNet on complex benchmark datasets, such as Canadian Institute For Advanced Research

(CIFAR-10), is still not on the same level as CNNs. One reason for this is that the implementation of CapsNet is susceptible to background information [5]. The shallow network structure of CapsNets lacks the ability of drawing features. It is not an ideal solution to simply stack the fully connected capsule layers, which may generate useless middle layers [6]. CapsNet passes the information of each capsule to the next layers at the full magnitude of its activation value yet lacks a suitable mechanism of selecting discriminative information from the outputs of each layer [7]. The initial logit b_{ij} of "routing Softmax" is the log prior probability that depends on the location and type of the two capsules, which determines how tightly capsule i should be coupled to capsule j. However, the Softmax function converts the logits of the coupling coefficients into a set of concentrative values, which may mistakenly send the background information to the next capsule layers with too large a coefficient, resulting in the wrong summation of the prediction vectors with larger values. This may affect the final result of the classification. In addition, the number of computational parameters for dynamic routing used in stacking capsules is huge, which leads to higher cost of training time.

In order to strengthen the anti-interference ability of the CapsNet [4] and discernibility of the output vector of each class and to improve the classification accuracy of images with complicated data and background, this paper presents the following contributions:

1. Drawing from Diverse Capsule Network (DCNet++) [8], we propose a novel architecture called Diverse Enhanced Capsule Network (DE-CapsNet). Multiple-layer residual blocks, instead of one convolutional layer, are used in a residual convolutional subnetwork to extract features from complicated data such as CIFAR-10. The features are input into different levels of primary capsules. DE-CapsNet utilizes a two-level primary capsules hierarchical model to represent different scales of images. Furthermore, the output from the primary capsule is assigned to digit capsules (DigitCaps) by a routing algorithm, and DE-CapsNet fuses the features of the two-level primary capsules together to identify the instantiation. Besides this, the Spatial Group-wise Enhance (SGE) [9] mechanism is introduced into our architecture as the enhancement method for the original capsule-based method. The enhancement is both between the neighboring residual blocks and inside the quasi-primary capsule layers, for the sake of helping the network to build dedicated capsules to improve the representation power of capsules. These dedicated capsules are focused on the true features and restrain susceptibility to the background information. It can tell the network which object or part of an object is truly important to learn.
2. Disperse dynamic routing is proposed that improves the performance of the dynamic routing algorithm. We found that the coupling coefficients using the Softmax function were mainly distributed around the interregion from 0.09 to 0.109, which is not as well distributed as can be obtained using the Sigmoid function. The Sigmoid function can assign larger coupling coefficients to real features, which transfer the true features actually related to the class to the next capsule layers, while assigning relatively smaller coupling coefficients to the fallacious ones. The true ones can be decisive in preventing the predicted sums of false classes from getting larger values.
3. Dynamic agreement routing is time-consuming due to the relatively higher complexity of its constituting elements. Our architecture is designed as two-level primary capsule layers with smaller kernel size in each primary capsule layer in order to reduce the training time compared with the seven ensembles of CapsNets.

2. Related Work

Increasing the depth of layers in networks promotes the performance of deep networks and stimulates the innovation of architectures. Highway Networks [10] is a deep feedforward network that provides an effective way to train networks with more than 100 layers by using bypassing paths [10]. ResNets further explores the effect of pure identity mapping by using it as the bypassing path, with deep layers which can achieve excellent performance in many challenging benchmark datasets [2]. Increasing the width of a network can help to train deeper networks. Feature maps operated by kernels with different sizes are concentrated using the "inception module" in GoogLeNet [1]. Huang et al. [3]

proposed a novel architecture called DenseNets that provides dense connections between all layers. It allows better gradient flow across deeper networks.

The current CapsNet [4] consists of one convolution layer, one primary capsule layer, and one digit capsule layer [4]. The input image is operated by a convolution layer with 256 9×9 kernels using a stride of 1 to extract features and then activated by the Rectified Linear Unit (ReLU) function. The output of the ReLU function is a feature map tensor. The primary capsule (PrimaryCaps) layer adopts a second convolutional layer with 9×9 kernels using a stride of 2 to deal with the feature map tensor. The output of the PrimaryCaps is also activated by the ReLU function. Every group of 8 scalars in the feature map tensor constitutes the primary capsule i. Capsules use feature vectors to represent the properties of entities which can capture position, size, texture, and other information. u_i is the output of primary capsule i. $\hat{u}_i$ is the prediction vector which is the input of final digit capsule j. W_{ij} is the weight matrix. $\hat{u}_i$ is calculated by Equation (1) [4].

$$\hat{u}_{j|i} = W_{ij} u_i \tag{1}$$

Routing-by-agreement will send the output of the primary capsules to the final capsules by increasing or decreasing the connection strength between the primary capsules and the digit capsules (DigitCaps) instead of pooling operation and keeping the spatial relations between object parts. It can be seen as a prediction which sends the output of the primary capsule i to the final digit capsule j. The coupling coefficient between the two capsules will increase when the output matches. b_{ij} is a logit of the Softmax function, which defines the coupling coefficient c_{ij} between capsule i in the layer above and capsule j in the layer below, as given by Equation (2) [4].

$$c_{ij} = \frac{\exp\left(\mathrm{b_{ij}}\right)}{\sum_{\mathrm{k}} \exp(\mathrm{b_{ik}})}, \; s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{2}$$

The length of these vectors s_j is the input of the digit capsule layer and is restricted to 1 by the squash function [4] to get the output v_j. The inner product of v_j and $\hat{u}_{j|i}$ updates the log probabilities b_{ij}. The more similar the two vectors, the longer the vector v_j will be. After several iterations, the one with the largest vector length in the final capsule layer corresponds to the true class.

Based on this, Hinton et al. [11] proposed matrix capsules using logistic units to represent the presence of entities and a pose matrix to represent the poses of the entities with the Expectation-Maximization (EM) routing algorithm. HitNet [12] uses centripetal function loss to train the Hit-or-Miss layers of capsules. The capsule corresponding to the true class makes a hit in its target space, while the others make misses. Zhao et al. [13] showed that the scale-invariant Max-Min function can promote the performance of CapsNet [4]. An optimization of the routing strategy and a new routing approach proposed in reference [14] outperformed the dynamic routing method in reference [4]. The Multi-Lane Capsule Network [15] divide the original Capsule Network [4] into multiple lanes to learn different dimensions of vectors that represent distinct features. The Diverse Capsule Network [8] uses three-level capsule layers to learn diverse features and concentrates the features into multi-dimensional vectors. Similarly, the Complex-valued Diverse Capsule Network [16] also utilizes a three-level hierarchical model but encodes complex-valued features for complicated datasets. DeepCaps [17] uses three-dimensional (3D) convolution and surpassed the state-of-the-art results in the field of Capsule Network. Capsule Network have been widely applied in many fields. Furkan et al. [18] investigated the performance of in-shop clothing retrieval using densely connected Capsule Network. Parnian et al. [19] studied the application of Capsule Network for the classification of Magnetic Resonance Imaging (MRI) images.

Attention mechanisms have achieved encouraging progress in the field of computer vision. These help the model to focus on the correlations between regions of images, including long-range dependence across image regions. The Squeeze-and-Excitation Network (SENet) [20] uses channel-wise importance to help attract attention for the model which puts higher weights on the channels with true

significance. Non-local Neural Networks [21] (NLNets) create blocks to compute the spatial weight of each point from the weighted sum of all positions. The Global Context Network (GCNet) [22] unifies the advantages of both Non-local and Squeeze-and-Excitation (SE) blocks together to get a more effective global context block based on the analysis of both blocks. Sanghyun et al. [23] proposed the Convolutional Block Attention Module (CBAM) focusing on both the spatial positions and channels via resizing. The results on object detection tasks are attractive.

3. Diverse Enhanced Capsule Network

3.1. Enhanced Capsules

We drew inspiration from the Spatial Group-wise Enhance module [9] in our architecture to enhance the representation power of capsules. Figure 1 shows the procedure of capsule enhancement.

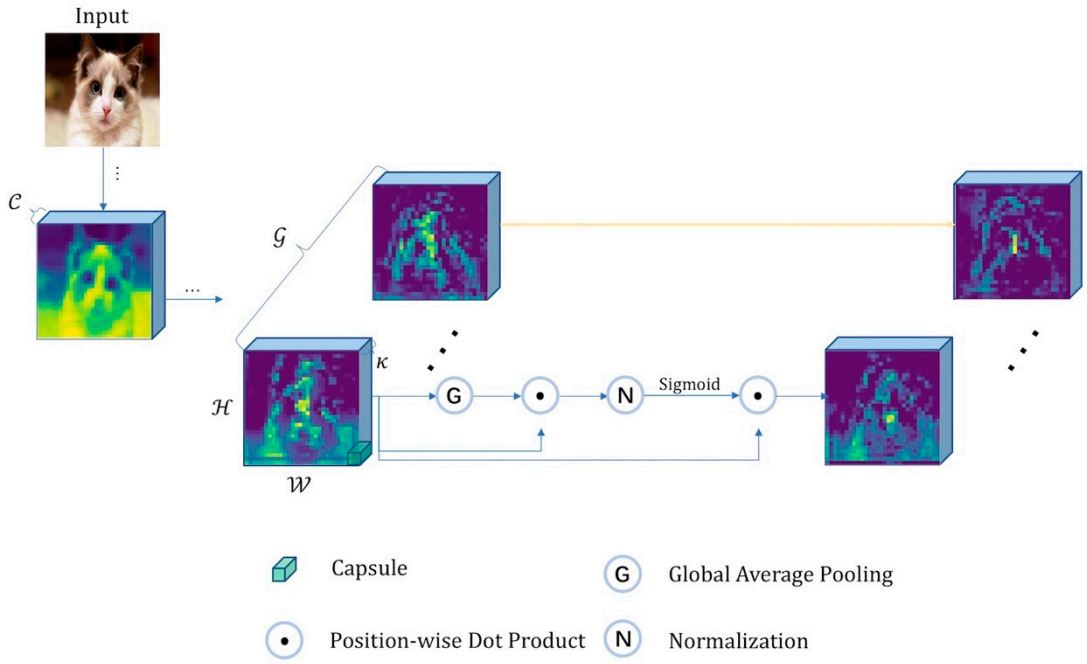

Figure 1. Illustration of capsule enhancement by dot product for group feature learning from Spatial Group-wise Enhance.

A capsule is a group of neurons. In the Spatial Group-wise Enhance module [9], the operations are based on each group of neurons, which can learn diversifying entity representations and learn the group-wise similarity. In view of this, we arranged C channels and $f = \mathrm{H} \times \mathrm{W}$ feature maps in groups. The quantity of feature maps in each set was equal to the dimension of each capsule. Therefore, $\mathcal{G}$ groups of feature maps are able to be viewed as $\mathcal{G}$ channels of $\kappa = \frac{C}{\mathcal{G}}$ dimension capsules. p_i is a vector that represents the capsule, $p_i \in \mathbb{R}^{\kappa}$. Different sets of feature maps constitute a space containing several capsules. The space is named $\Gamma = \left\{\mathrm{p}_1, \ldots \mathrm{p}_\mathrm{f}\right\}$.

First, we obtained the global feature $\mathbf{g}$ of each set of grouped capsules by computing the spatial average, as in Equation (3) [9].

$$\mathbf{g} = \frac{1}{f}\sum_{i=1}^{f} p_i \tag{3}$$

After that, the simple dot product was used to compare the resemblance between the global $\mathbf{g}$ feature and the capsule p_i. This can be simply seen as the projection of capsule p_i onto the global feature vector $\mathbf{g}$. θ_i is the angle between the two vectors, as in Equation (4) [9].

$$r_i = \mathbf{g} \cdot p_i = \left\| \mathbf{g} \right\| \left\| p_i \right\| \cos(\theta_i) \tag{4}$$

Normalizing r_i, which is shown in Equation (5), can offset the bias size between different samples [24].

$$\hat{r}_i = \frac{r_i - \mu_r}{\sigma_r + \epsilon} \tag{5}$$

Here, ϵ is a constant for numerical stability [24], μ_r is the expectation of $\mathcal{R} = \{r_1, \ldots r_i\}$, and σ_r is the variance of $\mathcal{R}$, with [9,25,26]

$$\mu_r = \frac{1}{f}\sum_j^f r_j, \sigma_r^2 = \frac{1}{f}\sum_j^f \left(r_j - \mu_r\right)^2.$$

Parameters γ and β corresponding to each coefficient of $\hat{r}_i$ scale and divert the normalized value to represent the characteristic transform, as indicated in Equation (6) [9].

$$a_i = \gamma\, \hat{r}_i + \beta \tag{6}$$

Finally, we adopt the Sigmoid function σ to scale the transforming space, and $\hat{p}_i$ is the enhanced capsule, as shown in Equation (7) [9].

$$\hat{p}_i = p_i \cdot \sigma(a_i) \tag{7}$$

The group of enhanced capsules is named $\hat{\Gamma} = \left\{\hat{p}_1, \ldots \hat{p}_f\right\}$. The enhanced capsule blocks are inserted in between the residual blocks in enhanced capsule residual convolutional subnetworks. Furthermore, the enhancement is introduced after convolution on the quasi-primary capsule layers.

3.2. Disperse Dynamic Routing

The inputs to the digit capsules (DigitCaps) are the "prediction vectors" $\hat{u}_{j|i}$ produced by learned transformation weight matrices and the outputs of the primary capsule layer [4]. The routing algorithm calculates the "digit capsules" v_j from $\hat{u}_{j|i}$, which is kept fixed throughout the procedure. The dynamic routing procedure from reference [4] is given as follows (Algorithm 1).

Algorithm 1. Softmax Routing Procedure

1: Input to Routing Procedure: $(\hat{u}_{j|i}, r, l)$
2: for capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$
3: **for** r iterations:
4: for capsule i in layer l :
$c_{ij} \leftarrow Softmax\left(b_{ij}\right)$
5: for capsule j in layer $(l+1)$:
$s_j \leftarrow \sum_i c_{ij}\hat{u}_{j|i}$
6: for capsule j in layer $(l+1)$: $v_j \leftarrow Squash\left(s_j\right)$
7: for capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} \cdot v_j$
8: **Return** v_j

$\hat{u}_{j|i}$ is the prediction vector that means feature i belongs to final digit capsule j. Digit capsule j matches one class of images. The coupling coefficient c_{ij} represents the relative strength between primary capsule i and digit capsule j, which are refined as Equation (2) is iterated. s_j is the input of digit capsule j and is calculated via Equation (2). In the digit capsule layer, we restrict the vector s_j to 1 by the squashing function as shown in Equation (8) [4].

$$v_j = \frac{\left\|s_j\right\|^2}{1 + \left\|s_j\right\|^2}\frac{s_j}{\left\|s_j\right\|} \tag{8}$$

The function shrinks short vectors to almost zero and long vectors to a length below 1. In reference [4], there are 10 classes represented by 10 capsules in the digit capsule layer. However, the distribution of coefficients is concentrated in the interval from 0.09 to 1.09, as demonstrated in Figure 2, which makes the "sum" s_j of prediction vectors poorly distinguishing. In other words, the probabilities of the features sent to digit capsules are nearly equal. As a result, the lengths of each vector v_j in the final digit capsule layer are close to each other, which may produce a wrong class.

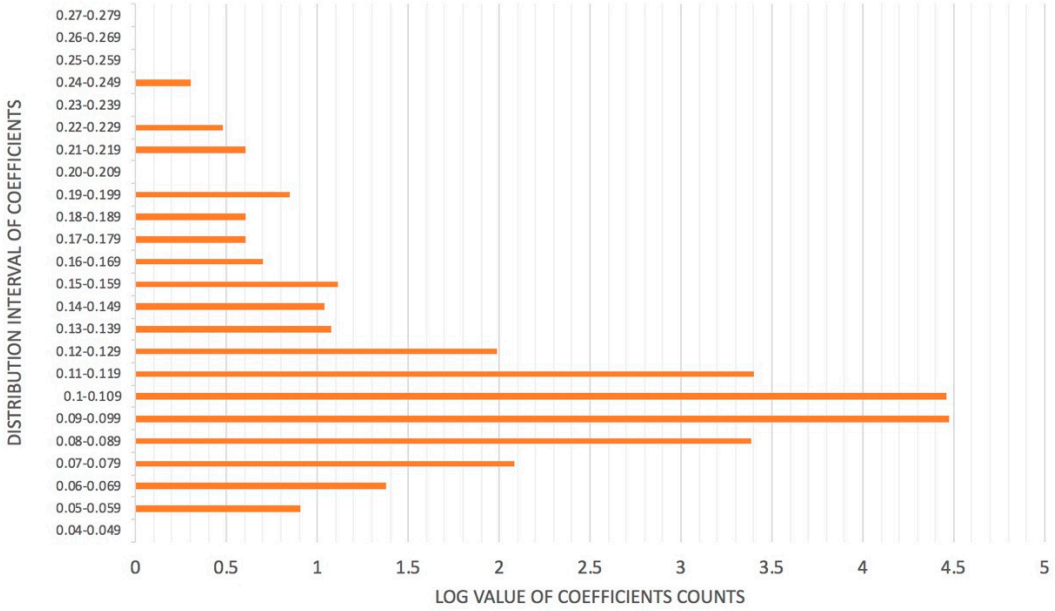

Figure 2. Distribution of routing coefficients for the same training image from the CIFAR-10 dataset for network training using Softmax. The horizontal axis represents the logarithm of the number of coefficients to base 10; the vertical axis represents the interval of coefficients.

Therefore, we calculated the coefficients by using the Sigmoid function instead of the Softmax function. Our Sigmoid routing procedure is almost the same as the Softmax routing procedure in reference [4], but we replace the Softmax function with the Sigmoid function as shown in Equation (9). c_{ij} no longer stands for the allocation probabilities toward the final capsules, but the correlation strength between the primary capsules and the final capsules. As can be seen from Figure 3, the distribution interval of the logarithm value of c_{ij} to base 10, as indicated in Equation (9), is much better distributed. The difference between the minimum and maximum coefficients is even bigger. The important prediction vectors are multiplied with larger coupling coefficients to make the significant features more decisive, while unrelated features get smaller ones. Besides this, it increases the difference between the lengths of the vectors in the final capsule layer. The correct digit capsule then exceeds all the other digit capsules in length. We adopted the Sigmoid routing in our model and the performance was better than that of the model using Softmax routing, which is shown as follows (Algorithm 2).

$$c_{ij} = \frac{1}{1 + \exp(b_{ij})} \tag{9}$$

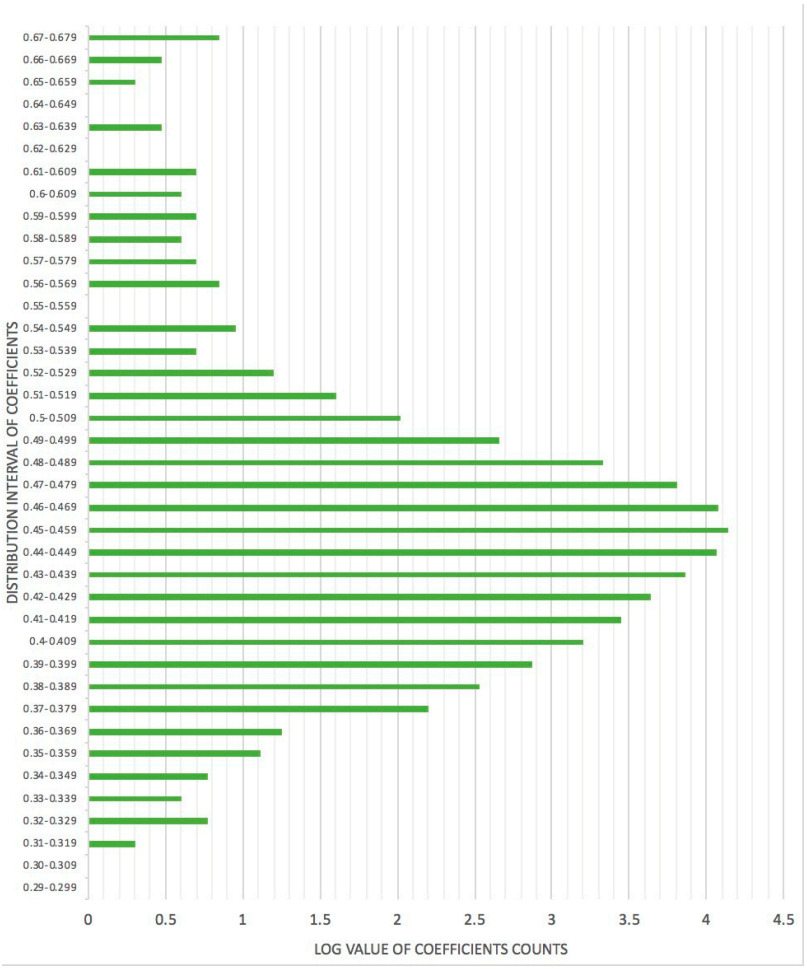

Figure 3. Distribution of routing coefficients for the same training image from the CIFAR-10 dataset for network training using Sigmoid. The horizontal axis represents the logarithm of the number of coefficients to base 10; the vertical axis represents the interval of coefficients.

Algorithm 2. Sigmoid Routing Procedure

1: Input to Routing Procedure: $(\hat{u}_{j|i}, r, l)$
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$
3: **for** r iterations:
4: for all capsule i in layer l :
$c_{ij} \leftarrow Sigmoid\left(b_{ij}\right)$
5: for all capsule j in layer $(l+1)$:
$s_j \leftarrow \sum_i c_{ij}\hat{u}_{j|i}$
6: for all capsule j in layer $(l+1)$:
$v_j \leftarrow Squash\left(s_j\right)$
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} \cdot v_j$
8: **Return** v_j

3.3. DE-CapsNet Architecture

Figure 4 demonstrates the training process of the CIFAR-10 dataset using our model. An enhanced capsule residual convolutional subnetwork was used to build the capsules based on the residual basic block [2] for complex datasets. These layers can copy other layers from the learned shallower model to reduce gradient loss [2] based on the connections from the initial layers to the later layers.

Going deeper improves the learning for capturing diversified features. The architecture contains two levels of primary capsule layers. Each primary capsule represents a small area of input, and different levels of capsules carry different scales of the image. The first-level primary capsules were established with two residual convolutional subnetworks and two quasi-primary enhanced capsule convolutional layers. The second-level primary capsules were established with one residual convolutional subnetwork and one quasi-primary enhanced-capsule convolutional layer. One example of the residual convolutional subnetwork and the quasi-primary enhanced capsules is shown in Figure 5. We used a 1×1 convolutional layer before every subnetwork to implement cross-channel information combination and to add nonlinear features. The feature maps of subnetworks act as grouped capsules called quasi-primary capsules. Compared to the model in reference [4], our model adopts one convolutional layer with 5×5 kernels using a stride of 2 and 3×3 kernels using a stride of 1 to build primary capsules, rather than 9×9 kernels. This significantly reduces the number of parameters that need to be calculated. The first quasi-primary capsule layer consists of 32 groups of capsules and the second quasi-primary capsule layer consists of 8 groups of capsules. The features from the quasi-primary capsules are reshaped by the squash function to form the primary capsule layer. Carrying various scales of images at different levels of primary capsules provides macro and local entities of features as shown in Figures 6 and 7. The output from the two-level primary capsule layers is passed into the squash activation layer by disperse dynamic routing to generate two digit capsule (DigitCaps) layers. Besides this, one more DigitCaps output layer is created by routing the concatenation of the two-level primary capsule layers to learn features from the various scales of images. The DE-CapsNet performs better than simple stacking of DigitCaps and joint back-propagation. Finally, the DigitCaps layers are concatenated and squashed to create a 32-dimension (32D) capsule for each of the 10 classes.

Margin loss [4] was adopted as the loss function in the proposed architecture to enhance the probability of the true class and restrain the others. The loss function is defined as shown in Equation (10) [4]:

$$L_k = T_k \max\left(0, m^+ - \|v_k\|\right)^2 + \lambda(1 - T_k) \max(0, \|v_k\| - m^-)^2. \quad (10)$$

In Equation (10), $T_k = 1$ if class k is true and $T_k = 0$ otherwise. λ, m^+, and m^- are hyper parameters. We set $m^+ = 0.9$, $m^- = 0.1$, and $\lambda = 0.5$ before training. λ is used to control the effect of gradient backpropagation at the initial learning [4]. The losses of the two layers are backpropagated separately.

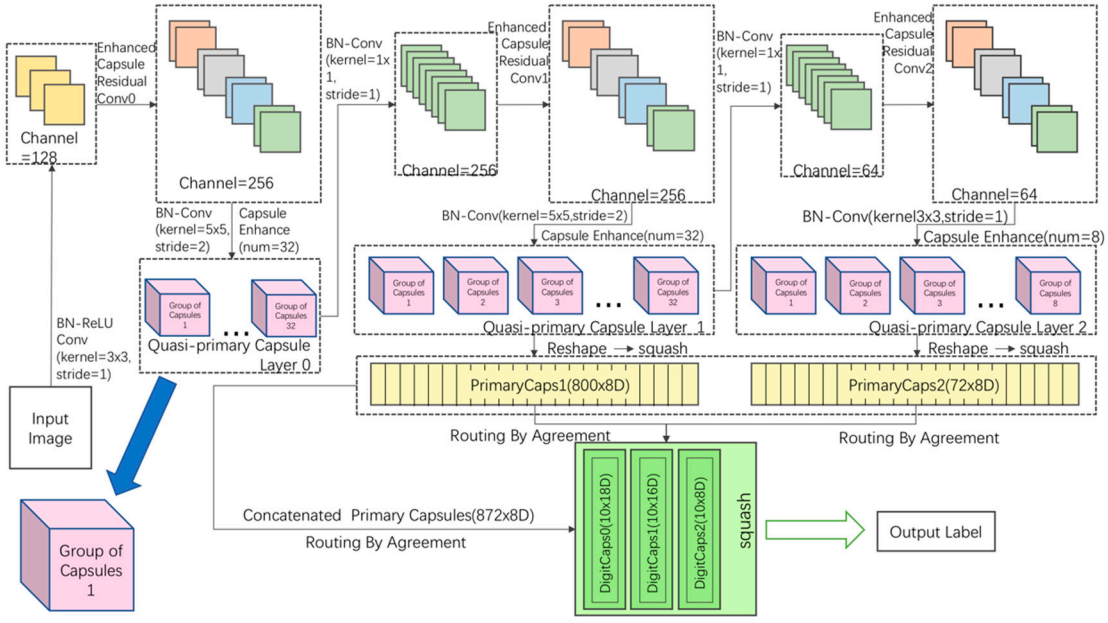

Figure 4. Pipeline of two-level DE-CapsNet for CIFAR-10.

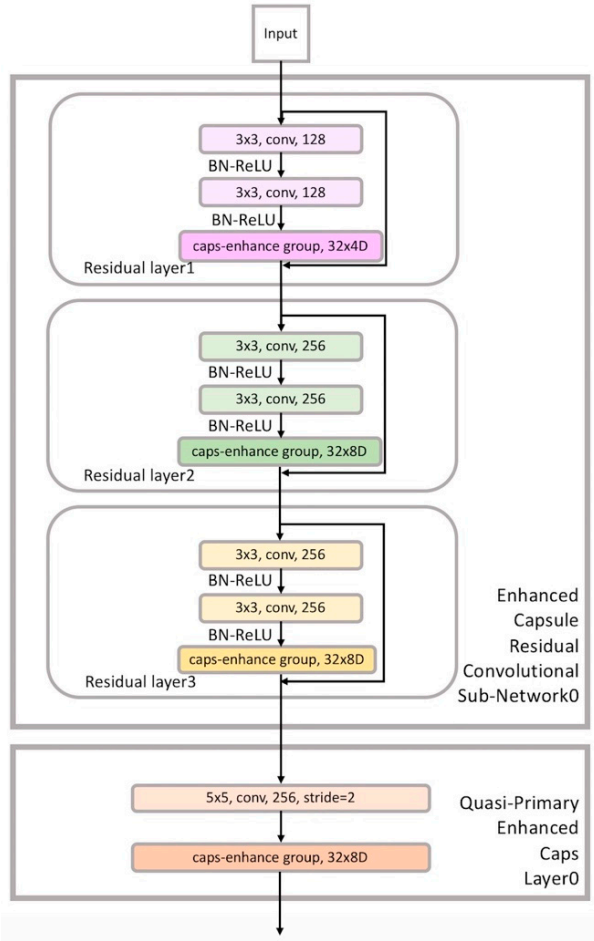

Figure 5. Enhanced Capsule Residual Convolutional Subnetwork 0 and Quasi-Primary Enhanced Capsule Layer 0.

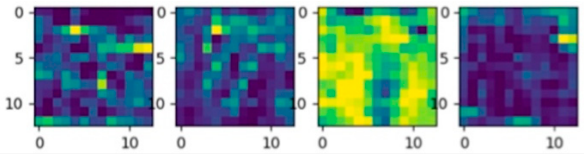

Figure 6. Output of the cat image from the first-level primary capsules.

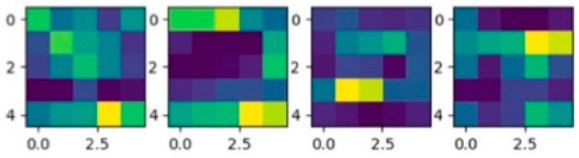

Figure 7. Output of the cat image from the second-level primary capsules.

4. Experiments

4.1. Datasets

The proposed model was evaluated on the Fashion-MNIST (F-MNIST) and CIFAR-10 datasets, with the results compared to those of the Capsule Network (CapsNet) [4] and the DeepCaps Network [18]. F-MNIST and CIFAR-10 were chosen as our datasets because of their complexity compared to MNIST. After scaling each pixel in the range of 0 to 1, each pixel value was divided by 255 before the model was trained on the image datasets.

CIFAR-10 is a subset of samples consisting of $32 \times 32 \times 3$ colored and labeled images in 10 classes, with 6K images per class. Five batches were used as the training data and one batch was used as the test data. F-MNIST includes 70K examples in the size of $28 \times 28 \times 1$, of which 60K and 10K labeled images were assigned as the training and test sets, respectively.

4.2. System Setup

Pytorch libraries were used to implement the DE-CapsNet. All the experiments were performed using GeForce GTX 1080 Ti with 16GB RAM. The initial learning rate was 0.0001 and the decay rate was 0.9 with Adam as the optimizer. Different hyperparameters were set for training CIFAR-10 and F-MNIST. The numbers of iterations were set to 80 and 60, respectively.

4.3. Results

The accuracy of prediction is defined as in Equation (11) [16].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

TP represents the number of true positive samples and TN represents the number of true negative samples. Similarly, FP represents the number of false positive samples and FN represents the number of false negative samples. Table 1 presents a comparison of the accuracy of our model with that of DeepCaps, CapsNet, and other variants of Capsule Network, which showed that we achieved state-of-the art results on CIFAR-10 in the Capsule Network field. Our results exceeded those of all other models in the Capsule Network field on CIFAR-10 and outperformed CapsNet which had seven ensembles on F-MNIST. There was a 3.56% improvement on CIFAR-10 and a 0.88% improvement on F-MNIST compared to the CapsNet results in [4]. Even though our results were slightly below those of the DeepCaps that had seven ensembles on F-MNIST, which does not have complex backgrounds, our model outperformed both DeepCaps with a single model [17] by 1.95% and DeepCaps with seven ensembles by 0.22% on CIFAR-10.

Table 1. Comparison of the accuracy of our model with that of DeepCaps, CapsNet, and other variants of CapsNets.

Model	CIFAR-10	F-MNIST
Sabour et al. [4]	89.40%	93.60%
HitNet [12]	73.30%	92.30%
Zhao et al. [13]	75.92%	92.07%
DeepCaps [17]	91.01%	94.46%
DeepCaps (7 ensembles) [17]	92.74%	94.73%
DE-CapsNet (ND)	92.33%	93.64%
DE-CapsNet	92.96%	94.25%

Our model achieved state-of-the-art results among capsule domain networks on CIFAR-10, being only slightly worse than seven ensembles of DeepCaps. ND—No Disperse Dynamic routing.

For our proposed model trained on CIFAR-10, DE-CapsNet had only 11.2 million parameters, while DeepCaps with seven ensembles [17] had 7×7.22 million parameters and CapsNet [4] had 22.48 million parameters. Our model achieved 92.96% accuracy on CIFAR-10, while DeepCaps with seven ensembles [17] achieved 92.74% and CapsNet [4] achieved 89.40%. Our architectures for CIFAR-10 are shown in Table 2.

Table 2. Our architectures for CIFAR-10.

Layer Name	Subwork-0	Subwork-1	Subwork-2
Conv1	3×3, 128, stride = 1	1×1, 256, stride = 1	1×1, 64, stride = 1
Conv2_x	$\begin{bmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{bmatrix}$	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix}$	$\begin{bmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{bmatrix}$
Caps-enhance	Groups = 32 Capsule dimension = 4	Groups = 32 Capsule dimension = 8	Groups = 8 Capsule dimension = 8
Conv3_x	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix}$	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix}$	$\begin{bmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{bmatrix}$
Caps-enhance	Groups = 32 Capsule dimension = 8	Groups = 32 Capsule dimension = 8	Groups = 8 Capsule dimension = 8
Conv4_x	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix}$	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix}$	$\begin{bmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{bmatrix}$
Caps-enhance	Groups = 32 Capsule dimension = 8	Groups = 32 Capsule dimension = 8	Groups = 8 Capsule dimension = 8
Quasi-conv	5×5, 256, stride = 2	5×5, 256, stride = 2	3×3, 64, stride=1
Quasi-primarycaps-enhance	Groups = 32 Capsule dimension = 8	Groups = 32 Capsule dimension = 8	Groups = 8 Capsule dimension = 8

5. Conclusions

In this paper, we proposed the Diverse Enhanced Capsule Network, or DE-CapsNet, with disperse dynamic routing. We drew inspiration from residual learning and Spatial Group-wise Enhance [9] to enhance the capsules in grouped channels representing the entities of images with complex data or backgrounds. Furthermore, on the basis of analyzing the distribution of coupling coefficients clustered around the value of 0.1, we proposed a disperse dynamic routing algorithm to increase the range of coefficients and strengthen the difference between the lengths of the true class and the others. We also adopted a smaller kernel size for primary capsules compared to Hinton's work [4] to reduce the number of computed parameters. Our model showed better performance and fewer trainable parameters than seven ensembles of CapsNets on CIFAR-10 and F-MNIST. This work has been proved applicable in the field of image classification. Compared to Convolutional Neural Networks (CNNs), our model has only few more parameters to calculate when achieving the same results because of the convolutional layers with larger kernel size in the primary capsule layers. Besides this, disperse dynamic routing agreement has to calculate the parameters cyclically for set iterations. The computational complexity could be further reduced for our model in the future. We plan to optimize the convolutional layers in the primary capsule layers and the disperse dynamic routing.

Author Contributions: B.J. designed and performed the experiments. B.J. and Q.H. wrote this paper. Q.H. supervised the whole work. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded in part by the National Natural Science Foundation of China (NSFC) through Grant U1732120.

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

References

1. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
2. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
3. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.

4. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic Routing Between Capsules. In Advances in Neural Information Processing Systems 30. In Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS, Long Beach, CA, USA, 4–9 December 2017; pp. 3856–3866.
5. Hinton, G.E.; Krizhevsky, A.; Wang, S.D. Transforming Auto-Encoders. In *Artificial Neural Networks and Machine Learning—ICANN 2011*; Honkela, T., Duch, W., Girolami, M., Kaski, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6791, pp. 44–51. ISBN 978-3-642-21734-0.
6. Xi, E.; Bing, S.; Jin, Y. Capsule Network Performance on Complex Data. *arXiv* **2017**, arXiv:1712.03480.
7. Yang, Z.; Wang, X. Reducing the dilution: An analysis of the information sensitiveness of capsule network with a practical solution. *arXiv* **2019**, arXiv:1903.10588.
8. Phaye, S.S.R.; Sikka, A.; Dhall, A.; Bathula, D. Dense and Diverse Capsule Networks: Making the Capsules Learn Better. *arXiv* **2018**, arXiv:1805.04001.
9. Li, X.; Hu, X.; Yang, J. Spatial Group-wise Enhance: Improving Semantic Feature Learning in Convolutional Networks. *arXiv* **2019**, arXiv:1905.09646.
10. Srivastava, R.K.; Greff, K.; Schmidhuber, J. Training Very Deep Networks. In Advances in Neural Information Processing Systems 28. In Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS 2015, Montreal, QC, Canada, 7–12 December 2015; pp. 2377–2385.
11. Hinton, G.; Sabour, S.; Frosst, N. Matrix Capsules With EM Routing. In Proceedings of the 6th International Conference on Learning Representations, ICLR, Vancouver, BC, Canada, 30 April–3 May 2018.
12. Deliège, A.; Cioppa, A.; Van Droogenbroeck, M. HitNet: A neural network with capsules embedded in a Hit-or-Miss layer, extended with hybrid data augmentation and ghost capsules. *arXiv* **2018**, arXiv:1806.06519.
13. Zhao, Z.; Kleinhans, A.; Sandhu, G.; Patel, I.; Unnikrishnan, K.P. Capsule Networks with Max-Min Normalization. *arXiv* **2019**, arXiv:1903.09662.
14. Wang, D.; Liu, Q. An Optimization View on Dynamic Routing Between Capsules. In Proceedings of the 6th International Conference on Learning Representations, ICLR, Vancouver, BC, Canada, 30 April–3 May 2018.
15. do Rosario, V.M.; Borin, E.; Breternitz, M., Jr. The Multi-Lane Capsule Network (MLCN). *IEEE Signal Process. Lett.* **2019**, *26*, 1006–1010. [CrossRef]
16. Cheng, X.; He, J.; He, J.; Xu, H. Cv-CapsNet: Complex-Valued Capsule Network. *IEEE Access* **2019**, *7*, 85492–85499. [CrossRef]
17. Rajasegaran, J.; Jayasundara, V.; Jayasekara, S.; Jayasekara, H.; Seneviratne, S.; Rodrigo, R. DeepCaps: Going Deeper with Capsule Networks. In Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 15–20 June 2019; pp. 10725–10733.
18. Kınlı, F.; Özcan, B.; Kıraç, F. Fashion Image Retrieval with Capsule Networks. In Proceedings of the 2019 IEEE International Conference on Computer Vision Workshops, ICCV Workshops, Seoul, Korea, 29 October–1 November 2019.
19. Afshar, P.; Mohammadi, A.; Plataniotis, K.N. Brain Tumor Type Classification via Capsule Networks. In Proceedings of the 25th IEEE International Conference on Image Processing, ICIP, Athens, Greece, 7–10 October 2018; pp. 3129–3133.
20. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Wu, E. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
21. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local Neural Networks. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7794–7803.
22. Cao, Y.; Xu, J.; Lin, S.; Wei, F.; Hu, H. GCNet: Non-local Networks Meet Squeeze-Excitation Networks and Beyond. *arXiv* **2019**, arXiv:1904.11492.
23. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. In Proceedings of the 15th European Conference on Computer Vision, ECCV, Part VII, Munich, Germany, 8–14 September 2018; pp. 3–19.
24. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Coference on Machine learning, ICML, Ithaca, NY, USA, 6–11 July 2015; pp. 448–456.

25. Qiao, S.; Wang, H.; Liu, C.; Shen, W.; Yuille, A. Weight Standardization. *arXiv* **2019**, arXiv:1903.10520.
26. Wu, Y.; He, K. Group Normalization. In Proceedings of the 15th European Conference on Computer Vision, ECCV, Part XIII, Munich, Germany, 8–14 September 2018; pp. 3–19.