Analysis and Learning of Capsule Networks Robust for Small Image Deformation

Nozomu Ohta*, Shin Kawai* and Hajime Nobuhara*
**Department of Intelligent Interaction Technologies,*
University of Tsukuba,
Tsukuba-City, Ibaraki, Japan
{ohta, kawai, nobuhara}@cmu.iit.tsukuba.ac.jp

***Abstract*—The Capsule Network (CapsNet) is a deep learning model proposed for image classification that is robust to pose of change of objects in images. A capsule is a vector representing the position, size and presence of an object. However, with CapsNet, the number of capsules increases, depending on the number of classification classes, and learning is computationally expensive. Thus, we propose a method for reducing computational costs by enabling a single capsule to represent multiple object classes. To learn the distance between classes, we incorporate the ArcFace distance learning method in the error function. In a preliminary experiment, the distribution of capsules was visualised by principal component analysis to demonstrate the validity of the proposed method. Using the MNIST and CIFAR-10 datasets, as well as an the affine transformed dataset, we compare the accuracy and learning time of the original CapsNet and proposed method. The results demonstrate that accuracy is improved by 2.74% on the CIFAR-10 dataset, and the learning time is reduced by more than 19% in both datasets.**

***Index Terms*—Deep learning, Convolutional neural networks, Capsule Network, Image recognition, ArcFace**

I. Introduction

Convolutional neural networks (CNNs) are commonly applied in the image recognition field; however, CNNs have some drawbacks. The general structure of a CNNs is presented in Fig. 1. CNNs recognise high-level features, e.g. eyes and mouth, from low-level features, e.g. line inclination, thickness and colour, by repeating a convolution layer for feature detection and a pooling layer for shift invariance. However, with CNNs, large-scale data expansion and very deep networks are required if the pose of the same object differs. This problem occurs because the pooling layer does not preserve the positional relationship between extracted features; thus, different poses of the same object result in entirely different internal representations (Fig. 2). To address this issue, the Google

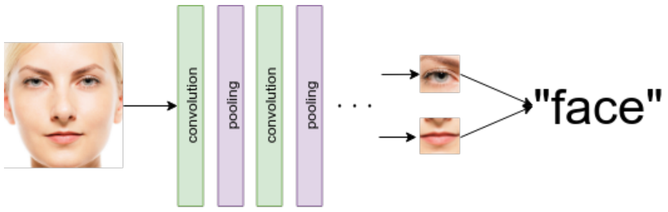

Fig. 1. Overview of CNN image recognition method

Brain team proposed the capsule network (CapsNet) [1], [2].

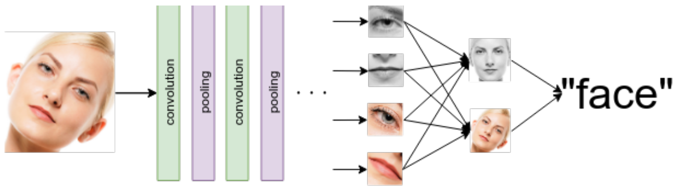

Fig. 2. Schematic diagram of CNN recognizing objects in different postures

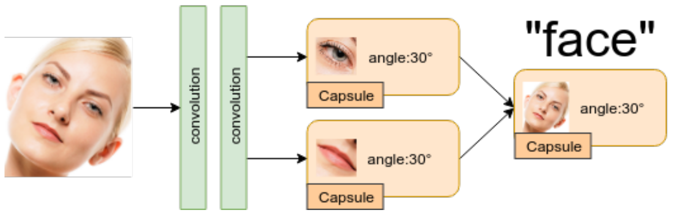

Fig. 3. Difference between CNN and CapsNet

A pixel of a feature map, which is an intermediate layer of conventional CNNs, determines whether a feature exists at a corresponding location. A capsule, which is a middle layer of CapsNet, outputs information about a part of an object as a vector. Fig. 3 shows that there are two types of capsules. The sub-capsules hold some features of the object as vectors, and the super-capsules are computed from multiple sub-capsules to represent the features and presence of the entire object.

CapsNet requires a long time for training and estimation. It is because in CapsNet, the number of parameters to be learned increases as the number of classes increases. This increase in the number of parameters is since the capsule network requires a super-capsule for each class to be classified. Therefore, it can be said that the problem with CapsNet is that it takes a long time to train and estimate.

The purpose of this study is to reduce the training and estimation time by reducing the number of parameters. For this purpose, we propose a learning method that can classify using a single super-capsule regardless of the number of classes to be classified. The proposed method is devised based on the analysis of the visualisation of CapsNet's internal representation. The results demonstrate that accuracy is improved on the CIFAR-10 dataset, and the learning time is reduced by greater than 19% in both datasets.

The remainder of this paper is organised as follows. Section II describes the structure of CapsNet, as well as the related studies and applications. Section III visualises super-capsules and examines the validity of the proposed method. Section IV discuss the proposed method in detail. Section V descirbes the experiments conducted to demonstrate the effectiveness of the proposed method. These experiments investigated the classification accuracy of the proposed method, the proposed method's robustness against small deformation, the time required to train the proposed method, and a subjective evaluation of images reconstructed by the proposed method. Section VI presents and discusses the results of these experiments. Section VII discusses the results of these experiments and further work.

II. Related Work

A. CapsNet Overview

This section briefly describes CapsNet and its related and applied studies. An overview of CapsNet is presented in Fig. 4, and the CapsNet calculation procedure is described in reference to this figure.

The calculation procedure of CapsNet is described as follows:

1) The two layers of the convolutional layer convert the input image to a feature map of 256 channels and 6×6. This conversion corresponds to a transformation between the input image in Fig. 4 and Conv1 and between Conv1 and PrimaryCaps.
2) Split the feature map pixel by pixel and 256 channels by eight channels. In ohter words, there are $6 \times 6 \times 32 = 1152$ eight-dimensional vectors. Here, the i-th vector is defined as the sub-capsule $\boldsymbol{u}_i \in \mathbb{R}^8, i = \{1, 2, \dots, 1152\}$. This corresponds to PrimaryCaps in Fig. 4.
3) $\hat{\boldsymbol{u}}_{j|i} \in \mathbb{R}^{16}$ is defined as the product of sub-capsule $\boldsymbol{u}_i$ and weight matrix $W_{ij} \in \mathbb{R}^{16\times 8}$:
$$\hat{\boldsymbol{u}}_{j|i} = W_{ij}\boldsymbol{u}_i, \tag{1}$$
where j is the ordinal of the super-capsule. In the proposed CapsNet, $j \in \{1, 2, \dots, 10\}$. Note that weight matrix W_{ij} is trained by an error backpropagation method.
4) $\boldsymbol{s}_j \in \mathbb{R}^{16}$ is given by the weighted sum of the product of $\hat{\boldsymbol{u}}_{j|i}$ and $c_{ij} \in \mathbb{R}$:
$$\boldsymbol{s}_j = \sum_i c_{ij}\hat{\boldsymbol{u}}_{j|i}, \tag{2}$$
where c_{ij} is calculated by dynamic routing, which is described later.
5) The following squash function,
$$\mathrm{squash}(\boldsymbol{s}) = \frac{\|\boldsymbol{s}\|^2}{1+\|\boldsymbol{s}\|^2}\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}, \tag{3}$$
calculates the super-capsule $\boldsymbol{v}_j = \mathrm{squash}(\boldsymbol{s}_j) \in \mathbb{R}^{16}$. This squash function is an activation function that converts the norm to the $[0, 1)$ range, thereby preserving the directions of vectors. The super-capsule $\boldsymbol{v}_j$ corresponds to DigitCaps in Fig. 4.
6) Repeat steps in 3–5 to construct 10 super-capsules $\boldsymbol{v}_j$.

The dynamic routing algorithm is presented in Algorithm 1. dynamic routing extracts sub-capsules $\boldsymbol{u}_i$ which are useful for classification and reconstruction in CapsNet. Essentially, dynamic routing algorithm extracts information, similar to the pooling layer in a CNN. Here, r is the number of routing times, and l is the number of layers. Previous studies [2] fixed $r = 3$ and $l = 1$. The dynamic routing process is summarised as follows.

1) Super-capsule $\boldsymbol{v}_j$ is calculated from sub-capsule $\boldsymbol{u}_i$ using the provisional weight c_{ij}.
2) The dot product of the calculated super-capsule $\boldsymbol{v}_j$ and each sub-capsule $\boldsymbol{u}_i$ is calculated. The calculation result can be considered as the similarity between super-capsule $\boldsymbol{v}_j$ and sub-capsule $\boldsymbol{u}_i$.
3) A new super-capsule $\boldsymbol{v}_j$ is calculated from sub-capsule $\boldsymbol{u}_i$ with the new weight c_{ij} for similarity.
4) Repeat steps 2 and 3 r times.

Thus, dynamic routing simultaneously estimates super-capsule $\boldsymbol{v}_j$ and selects sub-capsule $\boldsymbol{u}_i$. Here, super-capsule $\boldsymbol{v}_j$ corresponds to each class to be classified. Super-capsule $\boldsymbol{v}_j$ is used to classify and reconstruct images. The total loss function L is the weighted sum of the classification error L_{class} and reconstruction error L_{recon}:
$$L = L_{class} + \alpha L_{recon}, \tag{4}$$
where α is a factor used to balance classification and reconstruction errors ($\alpha = 0.0005$ in a previous study [2]).

B. Classification

The norm of each super-capsule $\boldsymbol{v}_i$ is defined as the existence probability of the corresponding object. The following margin loss function is used for the classification loss function:
$$L_{class} = \sum_k \Big(T_k \max(0, m^+ - \|\boldsymbol{v}_k\|)^2 + \lambda(1-T_k)\max(0, \|\boldsymbol{v}_k\| - m^-)^2 \Big), \tag{5}$$
where k is the number of classification classes, and T_k is 1 for a true class and 0 otherwise. m^+ is the threshold for errors in the true class, and m^- is the threshold for errors in the incorrect class. λ is a factor used to balance the errors of true and incorrect classes. In a previous study [2], $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$.

C. Reconstruction

CapsNet reconstructs an input image by combining three coupling layers to each super-capsule $\boldsymbol{v}_j$. The loss function L_{recon} of the reconstruction is the mean square error between the input and reconstructed images. A previous study [2] revealed that capsule elements, which are the elements of vectors represented capsules, describe how the corresponding object

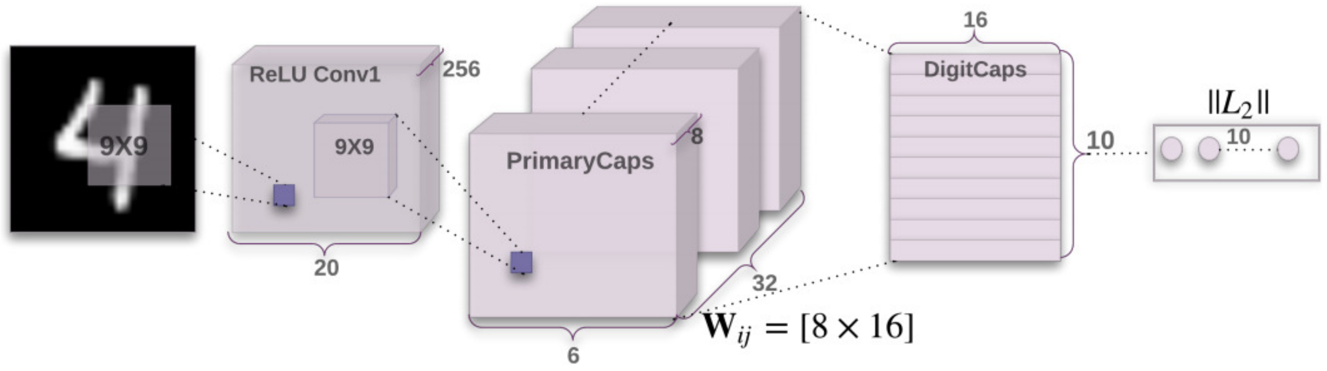

Fig. 4. Overview of CapsNet

Algorithm 1 Routing algorithm

```
procedure ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
    for all capsule $i$ in layer $l$ and capsule $j$ in layer$(l+1)$: $b_{ij} \leftarrow 0$
    for $r$ iterations do
        for all capsule $i$ in layer $l$: $\boldsymbol{c_i} \leftarrow$ softmax$(\boldsymbol{b_i})$
        for all capsule $j$ in layer $(l+1)$: $\boldsymbol{s_j} \leftarrow c_{ij}\hat{\boldsymbol{u}}_{j|i}$
        for all capsule $i$ in layer $(l+1)$: $\boldsymbol{v_j} \leftarrow$ squash$(\boldsymbol{s_j})$
        for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\boldsymbol{u}}_{j|i} \cdot \boldsymbol{v}_j$
    end for
    return $\boldsymbol{v_j}$
end procedure
```

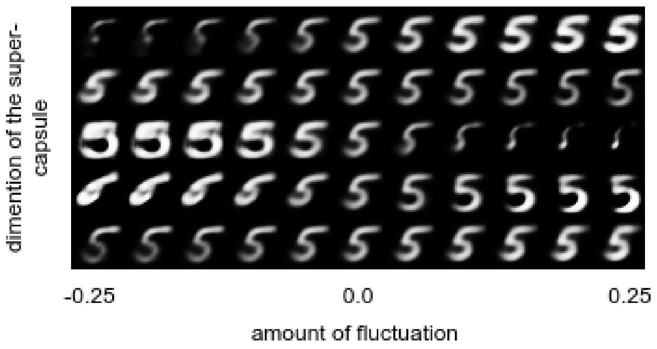

Fig. 5. MNIST Reconstructed Images with CapsNet

is positioned and scaled. They demonstrated that changes in a reconstructed object differ from element to element when each capsule element is increased or decreased slightly.

D. CapsNet Applied Research

Many studies have applied the characteristic of learning the whole part relation of CapsNet hierarchically to solve practical problems. In the imaging field, various studies have used CapsNet, to simultaneously solve the problem of classifying the motion of a person from a moving image and object detection [5] and to detect oddness in a falsified part of a forged image [3]. CapsNet has also been applied in the natural language processing field. For example, in a previous study [6], the subject, objects, and their relationships in a sentence were expressed as capsules and words and embedded by converting them to super-capsules. In the medical field, a previous study [4] attempted to classify brain tumours from MRI images using less training data than a CNN by utilising CapsNet's robust affine transformation.

III. CapsNet Experiment

The problem with CapsNet is that it takes a long time to train and estimate because it has many parameters. This section demonstrates that each super-capsule $\boldsymbol{v}_j$ is concentrated in a small part of each feature space, and discusses the scope of parameter reduction.

To examine the distribution of super-capsules $\boldsymbol{v}_j$ by class, super-capsules $\boldsymbol{v}_j$ are projected and visualised in a two-dimensional space based on two main components with high contribution rates, which are selected from super-capsules $\boldsymbol{v}_j$ by principal component analysis (PCA) [8]. PCA is a technique to compress multidimensional data into low-dimensional space. Figs. 7, 8 present the visualisations of super-capsules $\boldsymbol{v}_j$ classified using the MNIST and CIFAR-10 datasets, respectively. MNIST is a dataset of monochrome images of handwritten digits (0 to 9 28×28 pixels), which are used to evaluate image recognition. The CIFAR-10 dataset contains RGB images (32×32 pixels), with 50,000 learning data and 10,000 testing images, respectively. Note that there is no deviation in the number of images between classes in the CIFAR-10 dataset. In Figs. 7, 8, the super-capsules $\boldsymbol{v}_j$ of each

Fig. 6. Sample MNIST and CIFAR-10 datasets

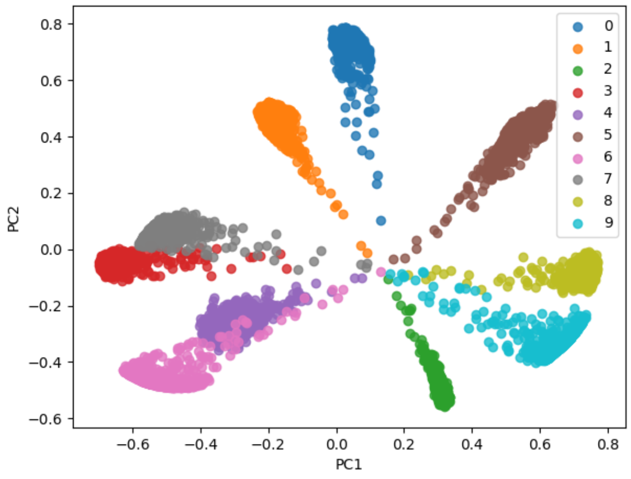

Fig. 7. Visualization of super-capsules of MNIST by PCA

class are distributed to extend in one direction from the origin.

Here, the following observations should be considered:

1) The super-capsules $\boldsymbol{v}_j$ of each class are not distributed throughout the feature space; therefore, there is no need to use a separate feature space for each class.
2) The conventional CapsNet uses the L^2 norm of the super-capsule $\boldsymbol{v}_j$, i.e. the distance from the origin, as a loss function. However, it is more appropriate to use the angle with the class centre rather than the distance from the origin for the loss function.

From the above considerations, we propose a classification method based on the angle with the representative point of each class that reduces the number of parameters W_{ij} by using a single super-capsule $\boldsymbol{v}$.

IV. Proposed Method

A. Outline

Based on the discussion in the previous section, this section describes the proposed method. In the conventional method, super-capsules are output for each class, however, in the proposed method, the parameters are reduced by outputting a single super-capsule. With this change, the margin loss is not available; thus, we also propose a new loss function based on ArcFace, a type of metric learning. ArcFace is a classification method based on the similarity between the super-capsule and representative vector of each class held as a parameter.

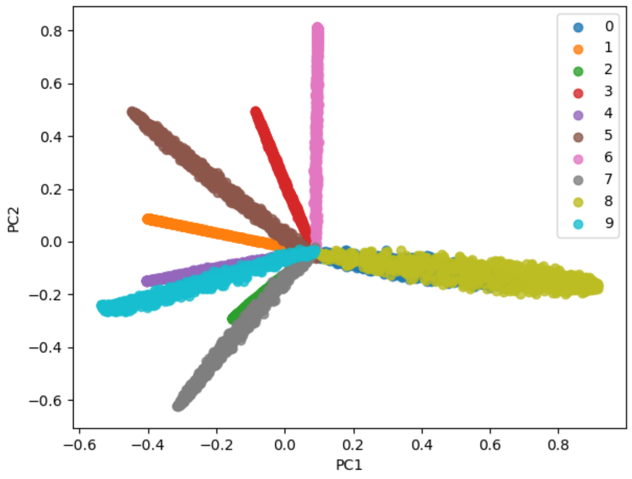

Fig. 8. Visualization of super-capsules of CIFAR-10 by PCA

B. Metric Learning

The prposed method is inspired by a technique called metric learning, which is training method such that the outputs of images of the same class are close to each other, and the outputs of images of different classes are distant from each other. Metric learning is primarily used to determine whether input face images are the same person and to detect abnormalities. In the proposed method, we employ the ArcFace [7] metric learning loss function to calculate the similarity between a class representative vector and a single super-capsule. Fig. 9 presents an overview of ArcFace. A feature of ArcFace is that it is trained such that a margin can be made with other representative vectors by imposing a penalty that the angle with the representative vector corresponding to the correct answer label must be less than the angle with other representative vectors by m or more.

C. Proposed Method

The procedure of the proposed method is described as follows:

1) A single super-capsule $\boldsymbol{v}$ is calculated using the traditional method.
2) Super-capsule $\boldsymbol{v}$ is then L^2 normalised to $\hat{\boldsymbol{v}}$.

$$\hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \tag{6}$$

3) Let $\hat{\boldsymbol{a}}_j$ be the L^2 normalisation of the representative vector $\boldsymbol{a}_j$ of class j. The representative vector is held in the network as a parameter. Here, j is the class number ($j \in \{1, 2, \ldots, 10\}$).
4) The angle θ_j [rad] between the normalised super-capsule $\hat{\boldsymbol{v}}$ and class vector $\hat{\boldsymbol{a}}_j$ is calculated.

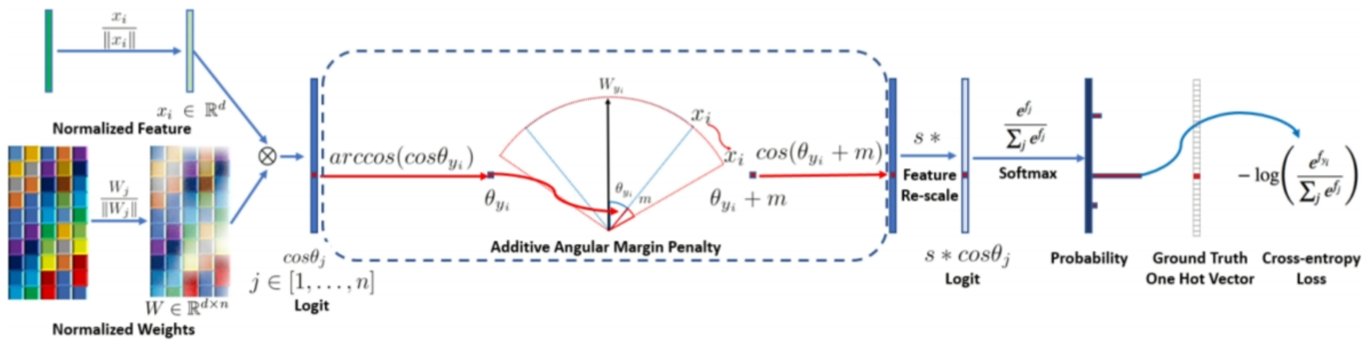

Fig. 9. Overview of ArcFace system [7].

5) Margin angles m [rad] are added to correct labels only. Here, let $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \dots\ \theta_{10}]^{\mathrm{T}} \in \mathbb{R}^{10}$ be the vector of the result θ_j of each class.
6) $\boldsymbol{\theta}$ is multiplied by s and input to the softmax function.
7) The squared error is learned from the one hot class label.

The overall error function L is given as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{k \neq y_i} e^{s\cos(\theta_k)}}, \quad (7)$$

where N is the number of batch samples and y_i is the correct label of the i-th data. The inner product of the representative vector and the super-capsule is a simple cosine similarity because each is L^2 normalised.

$$\cos\theta_j = \hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{a}}_{\boldsymbol{j}}. \quad (8)$$

ArcFace has two hyperparameters; a scaling parameter s and a margin parameter m. A larger s value results in the input difference being better reflected in the output and a larger m value means that the margin to be taken by the model will be larger, which makes it more difficult to train. Note that parameter m is not required in the test; thus, $m = 0$.

As in previous studies, the reconstruction of input images from super-capsules is performed simultaneously with classification. The error function of the reconstruction component is the mean square error, and the total loss function is the weighted sum of the classification and reconstruction errors.

V. Experiment

This section describes the experiments to validate the proposed method. Four experiments were conducted; comparison experiment with various hyperparameters was conducted to observe the change in accuracy rate caused by varying the ArcFace hyperparameters s and m; comparison experiment with conventional CapsNet confirmed the proposed method's validity; visualisation experiment of the super-capsule demonstrated that the super-capsule of the proposed method distributed in whole feature space; image reconstruction experiment presented that CpasNet is not suitable for image reconstruction;

The experiments were conducted using the MNIST, CIFAR-10, and their deformed datasets. In reference to a previous study [2], we make deformed datasets by the following procedure.

1) Rotates images uniform randomly in the range of $[-20, 20]$ degrees around the image centre.
2) Deforms images shears uniform randomly in the range of $[-0.2, 0.2]$ along the x axis and y axis.
3) Enlarge the image uniform randomly by a factor in the range of $[0.8, 1.2]$.

15% of each learning data is used as validation data.

Python 3 and pyTorch were used for the implementation, and a Geforce GTX 1080 Ti was used for computation. In addition, optimisation algorithm was performed using the Adam.

TABLE I
Test Accuracy with Varied Hyperparameter s

s	MNIST	CIFAR-10
1	99.24	59.28
5	**99.59**	**73.80**
30	99.33	70.96
100	99.57	10.00

VI. Result and Discussion

A. Comparison Experiment with Various Hyperparameters

TABLE I, TABLE II present the change in accuracy rate caused by varying the ArcFace hyperparameters s and m. As can be seen, no significant changes were observed in the MNIST dataset. In constant, with the CIFAR-10 dataset, setting s too small or too large decreased the accuracy rate. Even when m was 0, the accuracy rate did not decrease significantly. This indicates that the feature space is sufficiently wide to perform classification, and there is no need to set a margin m. Therefore the hyperparameter s and m are set to $s = 5, m = 0.1$ in the following experiments.

B. Comparison Experiment with Conventional CapsNet

The experimental results are presented in TABLE III when the ArcFace hyperparameters are $s = 5$ and $m = 0.1$. Here, the deformed and original columns represent cases where the test data were deformed and not, respectively. With the MNIST dataset, there was not much change in the accuracy rate. With

TABLE II
TEST ACCURACY WITH VARIED HYPERPARAMETER m

m	MNIST	CIFAR-10
0	99.55	73.36
0.01	99.43	73.12
0.1	**99.59**	**73.80**
1	99.33	64.78

the CIFAR-10 dataset, the accuracy of the proposed method improved by more than 2.74% regardless of the presence or absence of small deformations. TABLE IV presents the learning time for one epoch of the conventional and proposed methods. As can be seen, the learning time was reduced with both datasets by more than 19%. Obtained results imply that the reduction of the number of parameters by the proposed method is valid to reduce training and estimation time of CapsNet.

C. Visualisation Experiment of the Super-Capsule

Visualisation of super-capsule given by poposed method is presented in Figs. 10, 11. Here, the spherical distribution of the data points relative to Figs. 7, 8 is due to the normalisation of the super-capsules. CIFAR-10 has a denser distribution of super-capsules than Fig. 10 does Fig. 11.

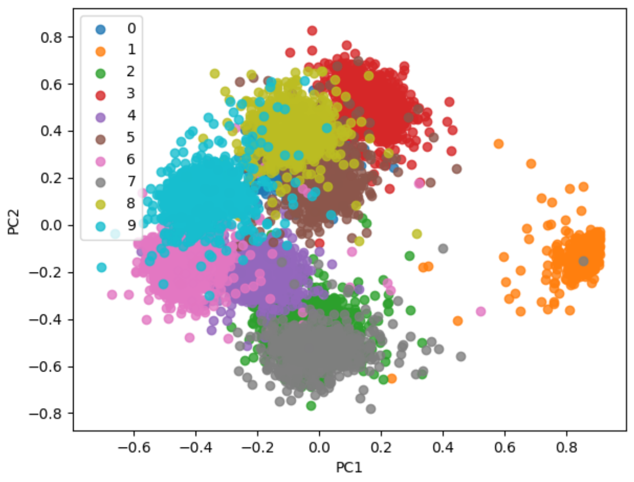

Fig. 10. Visualisation of the internal representation of MNIST datasets by PCA with the proposed method

D. Image Reconstruction Experiment

Figs. 12, 13 present the input images of each class without deformation (default input) and with the affine transformation (affine input), respectively, and their corresponding reconstructed images (reconstruction). With the MNIST dataset, the reconstructed image changed according to the rotation. In addition, with the CIFAR-10 dataset, the reconstructed image is not clear, and only the entire colour tone was learned.

Figs. 14, 15 present the success rate learning curve. With the CIFAR-10 dataset, learning was unstable; however, the accuracy rate of validation slowly increased, which indicates that learning was successful.

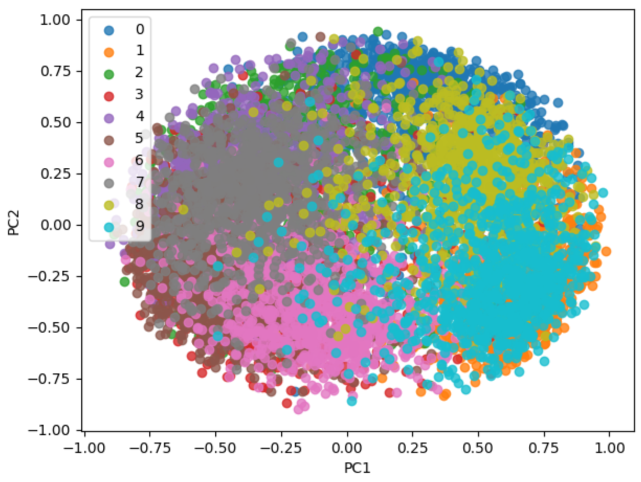

Fig. 11. Visualization by PCA of internal representation of CIFAR-10 dataset by proposed method

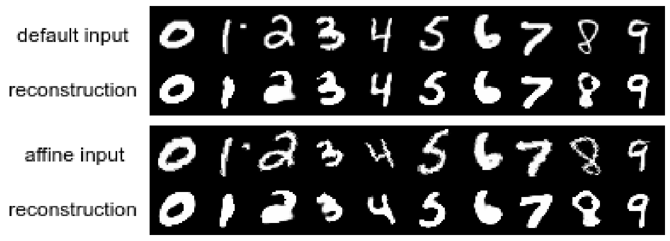

Fig. 12. Comparison of MNIST and reconstructed images

VII. CONCLUSION

From the experimental results, we conclude that the proposed method improves accuracy and reduces the learning time. The ArcFace hyperparameter s significantly affects the classification accuracy, whereas m does not contribute to the classification accuracy.

This indicates that the feature space is sufficiently wide to perform classification, and there is no need to set a margin. There are two limitations to this study. One is that the classification accuracy of CIFAR-10 is still low and the reconstructed image is blurred. This problem is thought to be due to the fact that the number and dimensions of the super-capsules are not enough to represent the features of the image. Second, in the proposed method, as the image size increases, the number of parameters increases and learning becomes difficult.

In the future, we plan to apply CapsNet to super-resolution video. In addition, it will be improved so that CapsNet can be trained on large data.

REFERENCES

[1] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming auto-encoders,” International Conference on Artificial Neural Networks, pp. 44–51, Jun. 2011.

TABLE III
ACCURACY RATE BETWEEN CONVENTIONAL (CAPSNET) AND PROPOSED METHODS

	Original		Deformed	
	MNIST	CIFAR-10	MNIST	CIFAR-10
CapsNet	99.47	70.96	**97.90**	61.63
Proposed	**99.59**	**73.80**	97.37	**69.92**

TABLE IV
ONE EPOCH AVERAGE LEARNING TIME [S] BETWEEN CONVENTIONAL (CAPSNET) AND PROPOSED METHODS

	MNIST	CIFAR-10
CapsNet	176.69	210.82
Proposed	**142.58**	**158.99**

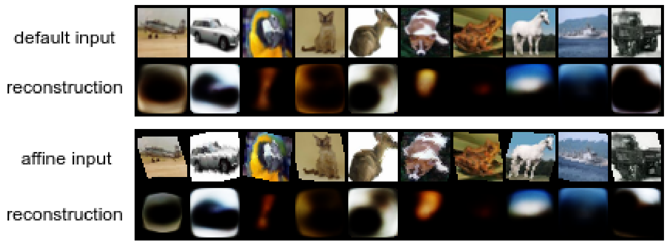

Fig. 13. Comparison of CIFAR-10 and reconstructed images

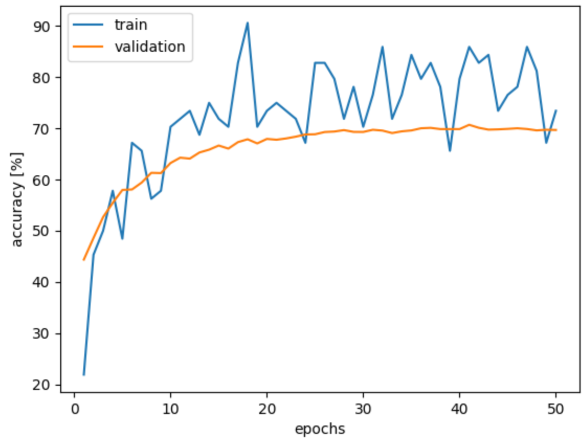

Fig. 15. Learning curve of accuracy rate on CIFAR-10 dataset

[2] S. Sabour, N. Frosst, G. E. Hinton, “Dynamic routing between capsules,” Advances in Neural Information Processing Systems, pp. 3856–3966, 2017.
[3] H. H. Nguyen, J. Yamagishi, I. Echizen, “Capsule-forensics: Using Capsule Networks to Detect Forged Images and Videos,” International Conference on Acoustics, Speech and Signal Processing, pp. 2307–2311, May 2019.
[4] P. Afshar, A. Mohammadi, and K. N. Plataniotis, “Brain Tumor Type Classification via Capsule Networks,” International Conference on Image Processing, pp. 3129–3133, Oct. 2018.
[5] K. Duarte, Y. Yogesh, and M. Shah, “VideoCapsuleNet: A Simplified Network for Action Detection,” Advances in Neural Information Processing Systems, pp. 7610–7619, 2018.
[6] H. Ren, H. Lu, “Compositional coding capsule network with k-means routing for text classification,” ArXiv, 2018.
[7] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” Advances in Neural Information Processing Systems, pp. 4690–4699, 2019.
[8] K. Pearson, “Principal components analysis,” The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 6, no. 2, p. 559, 1901.

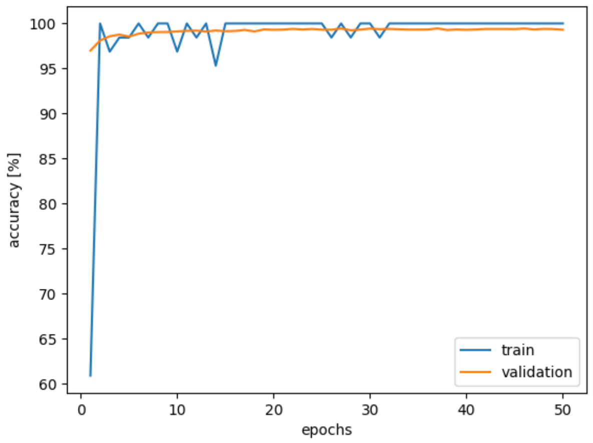

Fig. 14. Learning curve of accuracy rate on MNIST dataset