CapsNet, CNN, FCN: Comparative Performance Evaluation for Image Classification

Xuefeng Jiang, Yikun Wang, Wenbo Liu, Shuying Li, and Junrui Liu

***Abstract*—Image classification is one of the predominant tasks in computer vision. So far, there are many approaches in image classification, and the most typical methods are Convolutional Neural Networks (CNN), BOF-based algorithms, etc. Most of these methods have a good performance, but there are still some limitations. Capsule Network (CapsNet) is the most advanced algorithm, which realizes the operation based on active vector and dynamic routing, and can overcome limitations of the original algorithm. This paper attempts to apply CapsNet into image classification as well as another two efficient classification methods, which are CNN and Fully Convolutional Network (FCN). We use two datasets: MNIST and CIFAR-10 to train our model and tested the networks. Finally, compare and evaluate their performances in aspects of time cost, loss, accuracy and the number of parameters.**

***Index Terms*—Capsule network, convolutional neural networks, full convolutional neural network, image classification.**

I. Introduction

Image Classification (IC) is one of the most basic tasks in the dominant Computer Vision (CV) and those IC-based algorithms have been widely used in our life such as Hyperspectral Image classification, Optical Character Recognition, etc. Nowadays, Machine Learning (ML) based algorithms showed their great ability in the dominant performance of IC [1]. With the continuous development of machine learning, image classification steps into a brand-new era, arising many novel ML-based IC algorithms, such as k-Nearest Neighbor, Support Vector Machine, Convolutional Neural Network (CNN) [2] and so on. It's important to compare and evaluate these IC algorithms to find out their advantages and disadvantages.

In recent years, the standard solution to image classification has always been CNN which its performance has been performed well and continuously optimized. Convolutional Neural Network is one of deep learning [3]. It works by extracting features from images by convolutional neural networks and recognizing objects through feature learning. As the number of layers of a neural network increases, the features that can be extracted are more complex, and eventually, the convolutional neural network uses it to make judgments about the features learned. Nonetheless, if a standard multi-layer perceptron is used, which means all layers are fully connected, CNN will soon become having difficulty in calculating because the dimension of the image is too high. Moreover, CNN doesn't contain available space information. Besides, Traditional convolutional neural networks cannot be well identified for highly complex field-of-view images that contain a lot of overlap, mutual masking, and different backgrounds. Because of CNN's limitations, the classification results are not accurate enough.

Thus, Jonathan Longden [4] proposed the FCN neural network. FCN is a new method which only contains convolution layers and can classify each pixel of the image. The network is efficient, both asymptotically and absolutely, and precludes the need for the complications in other works. However, although FCN has improved accuracy, it does not take global context information into account, its segmentation is not instance-level, and is not efficient enough.

To address these issues, S. Sabour, N. Frosst and G. E. Hinton proposed the Capsule network in 2017 [5]. Geoffrey Hinton proposed "routing-by-agreement" to reduce the loss of interlayer transfer by transferring the underlying features to a matching high level. Their experiments showed that a discriminatively trained, multi-layer capsule system achieves state-of-the-art performance on MNIST and is considerably better than a convolutional net at recognizing highly overlapping digits.

So far, CapsNet shows excellent performance in various types of tasks, such as target recognition, analysis and so on. "Capsule network" has been recognized as the cornerstone of the next generation of deep learning, and the "successor" of CNN as well.

In this paper, we selected another two typical ML-based IC algorithms along with Capsule Network (CapsNet) [6]: CNN and FCN. And we compared them by analyzing their accuracy, time cost, etc.

In the following paper, we will first briefly review the three algorithms in Section II, describe the experiment in Section III and analyze the results in Section IV.

II. Related Work

A. Convolutional Neural Networks

CNN automatically extracts features on the image by building multiple layers of convolution layers [7], [8], resulting in a hierarchy of features. The front shallower convolution layer uses a smaller perceptive domain, which allows learning some local features of the image, and the

Manuscript received August 25, 2019; revised October 12, 2019. This work was supported by the National Natural Science Foundation of China under Grant 61701387.

Xuefeng Jiang, Yikun Wang, Wenbo Liu, and Junrui Liu are with the School of Computing, Northwestern Polytechnical University, Xi'an 710072, Shaanxi, P.R. China (e-mail: jxf@nwpu.edu.cn, yikunwang6@163.com, osmium@mail.nwpu.edu.cn, liu.junrui@nwpu.edu.cn).

Shuying Li is with the School of Automation, Xi'an University of Posts & Telecommunications, Xi'an 710121, Shaanxi, P.R. China (e-mail: angle_lisy@163.com).

doi: 10.18178/ijmlc.2019.9.6.881

back deeper convolution layer uses a larger perceptive domain and can learn more abstract features (such as object size, location, and direction information).

Trained pixels end-to-end, convolution network has achieved good performance: not only the overall image classification has been improved, but also the local task with structured output has made progress. Thus, it has been widely used in the field of image classification and image detection. But CNN still has the following limitations:

1) If a standard multi-layer perceptron is used, which means all layers are fully connected, CNNs will soon become having difficulty in calculating because the dimensions of the image are too high.
2) CNN doesn't contain available space information. If we use the pooling layer, the final efficiency may be very low. Since only the most active neurons [9] can be transmitted between each layer during the transfer of neurons, CNNs have a big loss of information in the pooling layer, reducing the spatial resolution. Therefore, CNNs will not be able to distinguish differences in postures and other facets. One way to solve this problem is to over-train all possible angles, but it usually requires more time and computing resources.
3) Traditional convolutional neural networks cannot be well identified for highly complex field-of-view images that contain a lot of overlap, mutual masking, and different backgrounds.

Due to these limitations, its classification results are not accurate enough.

B. Fully Convolutional Neural Network

Ning *et al*. [10] defined a cone for the coarse multi-class segmentation of C.elegans with full convolution reasoning. Sliding Window Detection by Sermanet *et al*. [11], the Semantic Split by Pinello and Collobert [12], as well as image recovery by Eigen *et al*. [13], all made full convolution. Tompson *et al*. [14] are effectively used to learn the pose estimation of end-to-end part detectors and spatial models [15]. He *et al*. [16] abandoned the non-convolutional parts of the classification network to make feature extractors. They combine recommendations with spatial pyramid pools [17] to generate localized fixed-length features for classification. While this hybrid model is fast and effective, it cannot be learned from end-to-end.

To address the challenges and verify the improvement of CNN's accuracy, Jonathan Longden has proposed the FCN model. Fully Convolutional Networks (FCN) recovers the category of each pixel from the abstract features. Namely, the problem of semantic segmentation is solved by extending the classification from image-level to pixel level. FCN adopts the convolutional layer in place of the full connection layer of the traditional convolutional network. Different from CNN, which uses a full connection layer to generate feature vectors of fixed length for classification after the convolution layer, FCN allows input images of any size. Meanwhile, to reduce the influence from the convolution and pooling process the size of the original image, a deconvolution layer is adopted in upsampling to recover its original size and spatial information as well as generating a prediction for each pixel. Finally, process classification on the characteristics of the sampling feature map.

C. Capsule Network

A new algorithm based on CapsNet has recently been proposed, which provides feasible ideas for further refinement of the results. We believe that CapsNet has the potential to perform better by making some modifications to the hyper-parameters.

Data sets are the metrics for evaluating the performance of the capsule network and its ability to model object features from different perspectives. For CapsNet, several important data sets recently proposed are the Yale face database B, CIFAR-100 Data set (includes 60,000 images of 100 everyday items (e.g. vehicles and animals)), The Belgium TS data set, etc. In these data sets, changes between images of the same class (lighting, posture, size, etc.) are the biggest challenges facing image classification tasks.

The capsule network works by storing all the extracted features [18] in the capsule in the form of vector. CapsNet replaces the scalar output feature detector from CNN with vector output and replaces the largest pooling with a protocol-by-protocol route, enabling replication of what has been learned across space. For example, when processing gesture information, the capsule network calculates the probability of an object's existence while encoding the spatial information and represents it in the form of a vector. The module length of the vector represents the probability, and its direction represents the attitude information.

The core of the capsule network [19] is inverse-rendering. The capsule network learns to predict and estimate parameters by comparing represented objects with the labels on the training data set. In the real-time traffic-condition monitoring pedestrians and vehicles have a great probability of appearance dynamically, which means the characteristics that we extract are mobile. However, this won't change the modal length of the vector in the capsule network, indicating the probability of the appearance won't be affected. This is also one of the outstanding advantages of capsule networks used in traffic images. Capsule networks provide the opportunity to take full advantage of inherent spatial relationships and simulate the ability to understand image changes.

For low-level capsules, location information [20] is "position-coded" by the active capsule. As the hierarchy is raised, more and more positional information is "rate-coded" [21], [22] in the actual value component of the capsule output vector. All of these imply that as we move up the hierarchy, the dimensions of the capsule sits must increase. Anyway, Capsule networks allow us to take full advantage of inherent spatial relationships [23] and simulate the ability to understand image changes to better summarize what we perceive.

III. Proposed Approach

In this section, we described models based on CNN, FCN, and CapsNet separately in the experiment.

A. The Algorithm Based on Convolutional Neural Network

CNN is a kind of machine learning algorithm that is more applied in the field of image recognition, which is consisted of three layers: convolution layer, pooling layer, full

connection layer.

The activation function we choose was the ReLU function [24] and the Softmax function [25]. The ReLU function is nonlinear. When the input x is positive, its output (as shown in Fig. 1 is x itself. When x is negative or zero, its output is zero. That is,

$$relu(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (1)$$

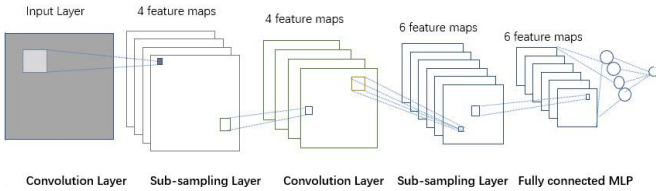

Fig. 1. Output via ReLU function.

The reason we choose it as the activation function is the sparsity of the activation. Generally, we hope some neurons aren't activated to make the activation sparse and efficient. Considering the ReLU function's character (The output is zero when the input is negative.), there should be 50% neurons not activated, meaningless computation consumption. However, the gradient of the horizontal part of the ReLU function tends to be zero. Meaning the region activated by the ReLU function may not be adjusted.

In the analysis of images studied, the steps to apply CNN are as follows:

1) Multiply the input traffic scene image with the weight via the convolution layer to obtain the preliminary product. Here we choose the filter matrix. Let the step length equals to 1, and do padding on the original image, then we get the processed input vector x.
2) Figure out the result of the activation function ReLU function. The output is:

$$\max(0, w^{\top} x + b) \quad (2)$$

3) Enter the pooling layer for abstract dimensionality reduction and prevent over-fitting.
4) Logical Regression via MLP, then output the probability of getting the classification of the input image.

The CNN model we used is shown in Fig. 2.

Specifically, the network contains one input layer and a pair of convolution and max-pooling layers. Then follows a flatten layer and two fully connected layers. We use ReLU function as the activation function for all but the last layers. The Softmax function is adopted for the last layer to normalize the prediction result.

Softmax is a common activate function which can compress a K - dimension vector with any real number into another K - dimension real vector. The form of this function is usually given as follows:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad \text{for } j = 1, \ldots, K. \quad (3)$$

Suppose each picture of the dataset is a patch of $x \times x \times k$ ($28 \times 28 \times 1$ for MNIST and $32 \times 32 \times 3$ for CIFAR-10), here's how the input is processed in the network. The first convolution layer filters the input with 128 filters with a kernel whose size is 5×5. So, the extracted feature map has the size of $x1 \times x1 \times 128$, where $x1 = x - 4$. Then a max-pooling layer pools the feature map using a 3×3 kernel thus the pooled feature map is $x2 \times x2 \times 128$, where $x2 = x1 / 2$. The next convolution and pooling layers perform the same operation on the feature map but with different kernel sizes. Then a flatten layer flat the feature map and passes the output to two fully connected layers, where the last layer has ten neuro and each of the neuro's output represents the probability that the input image belongs to the corresponding class.

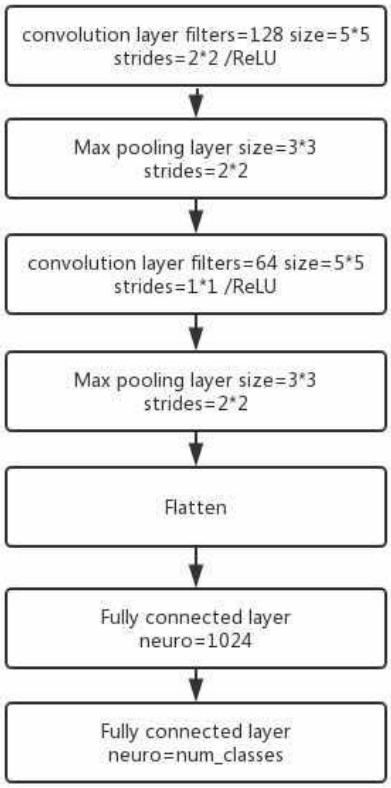

Fig. 2. CNN model.

B. The Algorithm Based on Fully Convolutional Networks

Different from CNN, FCN classifies images at the pixel level. The FCN can accept input images of any scale at the convolutional connection layer, and use the deconvolutional layer to sample the feature map of the last convolutional layer and restore the size of the image.

The steps applied to our research questions are as follows:

1) Input the original image, full convolution to extract feature (as shown in Fig. 3).
2) Change the filter size of the last fully connected layer to 1 and change it to the convolution layer.
3) Via the deconvolution layer, sample the characteristic image (feature map) of the last convolution layer and restore the size of the image to preserve the spatial information in the original image (as shown in the Fig. 4).
4) Predict feature image pixel by pixel. That is, apply a fully connected layer to each pixel. To classify each pixel, classification is obtained as a probability by performing a maximum numerical description of the position in each image one by one.
5) Deconvolution to enlarge and achieve ascending

sampling.

Fig. 3. Fully convolutional models –convolution.

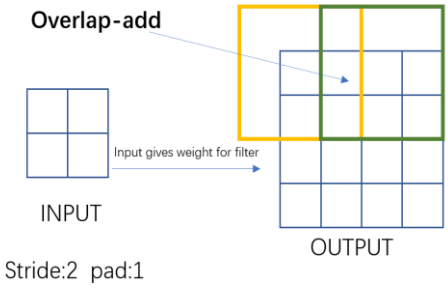

Fig. 4. Fully Convolutional Models –Deconvolution

We notice that since the FCN can preserve the spatial information of the original image, it can better analyze the information in the traffic scene than CNN.

The shape of the FCN model is shown in Fig 5.

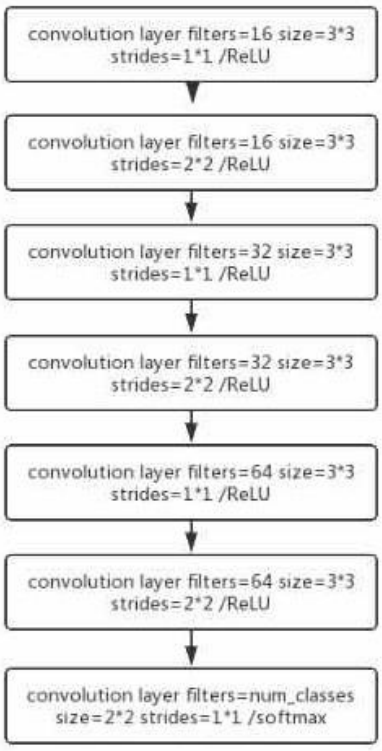

Fig. 5. FCN model.

The FCN model contains seven layers, the same as the CNN model but only contains convolution layers. The FCN model uses two convolution layers whose stride is two to replace the pooling layer in the CNN model. And the last layer is a convolution layer with ten kernels instead of the fully connected layer in the CNN model. The first six-layer used the ReLU function to non-linearize the output and the final layer uses Softmax function to normalize the output.

Here's how the FCN model processes the input data. All the layers use filters with different size to extract features and pass the feature map to the next layer. The final layer uses ten filters to extract the prior feature and pass it to the Softmax activation function to output the final probability.

C. The Algorithm Based on CapsNet

In artificial neural networks, Scalar neuron （SN） first receives the input scalar × from other neurons and multiply it by the corresponding weight to figure out the scalar a. Finally, use a non-linear activation function to generate an output scale as the input scalar for the next neuron.

CapsNet replaces the scalar output feature detector from CNN with vector output. It also replaces the largest pool sam-flat line technology with a protocol-by-protocol route to achieve the cross-space learning of the reproduction. In this new CapsNet architecture, only the first layer of cap capsules, also known as primary capsules, include CNN layer groups. Traditional CNN ideas, higher-level capsules cover images of a wider area. However, information about the exact location of the entity in the area is the opposite of the standard CNN.

And in the capsule network, we use vector neurons (VN). Similar to the above process, first, process the original input vector u with the attitude matrix W to generate the final input vector U. Then Multiply U with the corresponding weight c and sum them up to get the vector s. Finally, s is converted into the final vector v by a nonlinear function. This enables the transition between the underlying and high-level features. If the position of the high-level feature being pushed out by different underlying features points roughly the same place, it is possible to determine that the object has a great probability of existence.

The concrete steps (as shown in the Fig. 6) are as follows:

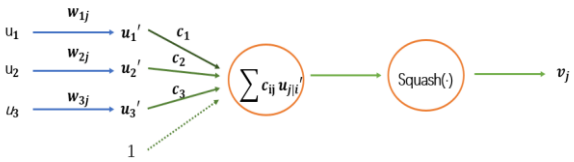

Fig. 6. CapsNet algorithm.

1) Initialize all probability and parameters, let all b equals to zero to make c distributed evenly. Via Softmax function to normalize.
2) Convert the input data via the attitude matrix.
3) Use dynamic routing to input weights. According to the formula $c_{ij} \times U_{j|i}$, adjust these two parameters to make decisions about the mapping between levels constantly, where c_{ij} represents the weight and the sum is equal to 1;
4) Weighted summation, similar to normal neurons corresponding steps to get the average position of the features of the object in the image. Softmax function's output is non-negative and its sum is 1, where *c* is a set of probability variables, s_j is the approximate center of the underlying feature output.
5) Using the nonlinear activation function Squash function, *s* is united and converted into the final vector *v*, the modular length of *v* is the final output.

The definition of the Squash function is as follows:

$$v_j = \frac{\|s_j\|}{1+\|s_j\|^2} \cdot \frac{s_j}{\|s_j\|} \quad . \tag{4}$$

This function compresses the length to range from 0 to 1 and keep the direction unchanged. That is, the squash function ensures that the direction of the vector remains the same and the length is compressed. This is also in line with the fact that the corresponding vector module length of the capsule network mentioned at the beginning of this paper represents the probability of the existence of the feature.

6) Use $u_{j|i} \cdot v_j$ to update parameter b_{ij} according to:

$$b_{ij} \leftarrow b_{ij} + u_{j|i}^{'} \cdot v_j \tag{5}$$

7) The loss function of CapsNet can be figured out by Margin Loss which is usually used in SVM [26]:

$$L_c = T_c \max\left(0, m^+ - \|v_c\|\right)^2 + \lambda\left(1 - T_c\right)\max\left(0, \|v_c\| - m^-\right)^2 \tag{6}$$

Inspired by the CNN architecture of Le-Net-5 References [27], [28], since the capsule is suitable for characterization of advanced instances, but the extraction of the underlying features is not ideal. Hence, we chose CNN to extract lower-level features. The advantage of having CNN is that it requires a much lower number of trained parameters than a multi-layer feed neural network that supports shared weights and partially connected layers to build the first layer of a neural network from image input to low-level features. This step uses 256 9×9 filters with steps of 1 to get the output of 20×20×256 and gets a local feature detection of the original image pixels. The second layer is then built from the lower-level feature to the primary capsule used to store the low-level feature vector by using 8 convolution operations with 9×9×256 filters with the steps of 2; from the construction of a network from the primary capsule to the corresponding network used to store higher levels of feature vectors, which are fully connected (FC) in the form of vectors and vectors; finally we output the probability of object appearance.

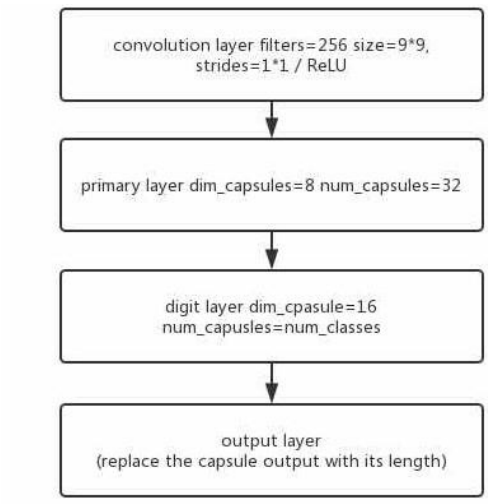

Fig. 7. CapsNet model.

It is worth noting that because Capsule can identify multiple objects at the same time, the sum of the probabilities of the final output is not equal to 1.

Compare to the CNN and FCN model, the CapsNet contains a shallower network with only four layers. The first layer is a convolution layer and follows the primary layer. Then follows a digit layer and an auxiliary layer to replace each capsule with its length. The steps are shown in Fig. 7.

IV. Experiment and Analysis

We evaluated CNN comparing with CapsNet, FCN algorithms in aspects of training time, training loss and accuracy on MNIST (as shown in Fig. 8) and CIFAR (as shown in Fig. 9). The hardware we use an NVIDIA RTX 2080Ti with 11GiB memory and dual Intel Xeon E5-2678 v3 with 64GiB memory Compare to the CNN and FCN model, the CapsNet.

The classification performance metrics used in this paper are as follows:

1) Time: The time consumption of the training process. Less time consumption means more efficient to train the model. Less is better.
2) Loss: Representing how different the model's prediction is from the true label. In CNN and FCN model we use cross-entropy loss function and in CapsNet model we use margin loss function. Less is better.
3) Accuracy: The ratio of the number of category pixels that are correctly classified to the total number of categories. We use overall accuracy in the experiment. Higher is better.

A. Data Sets

This paper utilizes MNIST(Fig. 8) and CIFAR-10(Fig. 9) as data sets.

Fig. 8. Example images from MNIST.

Fig. 9. Example images from CIFAR-10.

MNIST is a database of the handwritten digit, which is composed of four parts: the training set images, training set labels, test set images, test set labels. To be specific, the training set has 60000 examples in total while the test set has 10,000 examples. Chris Burges and Corinna Cortes originally selected the digit images in the MNIST set by using bounding-box normalization and centering and exerted them on the experiment. Later, Yann LeCun updated the version. Via computing the center of mass of the pixels and translating the image, the images in MNIST were centered in a 28×28 image.

Another dataset is CIFAR-10 purposed by Alex Krizhevsky, Vinod Nair and Geoffrey Hinton. CIFAR-10 consists of ten different classes, which each class contains

6,000 pieces of 32×32 RGB images. The dataset is divided into five training batches and one test batch, each with 10,000 images. To be specific, the training set has 50,000 examples in total while the test set has 10,000 examples. The training batch contains the remaining images in a random order, but some training batches may contain more images from one category than the other. As for the test batch, it contains exactly 1,000 randomly selected images from each category. Generally speaking, the sum of the five training sets contains exactly 5,000 images from each class. These classes are completely exclusive. "Cars" are cars, SUVs, and things like that. "Truck" only includes big trucks. There is no overlap between cars and trucks.

B. Training

When training, we trained the CNN and FCN model with both MNIST and CIFAR-10 datasets and trained each for 50 epochs. For the 70000 pictures in the MNIST dataset, we used 60000 pictures to train the model and 10000 pictures to test the model. For the 60000 pictures in the CIFAR-10 datasets, we use 50000 pictures to train the model and 10000 pictures to test the model.

C. Results and Analysis

The training results are shown in Table I and Table II while the model's loss and accuracy are showed in Fig. 10.

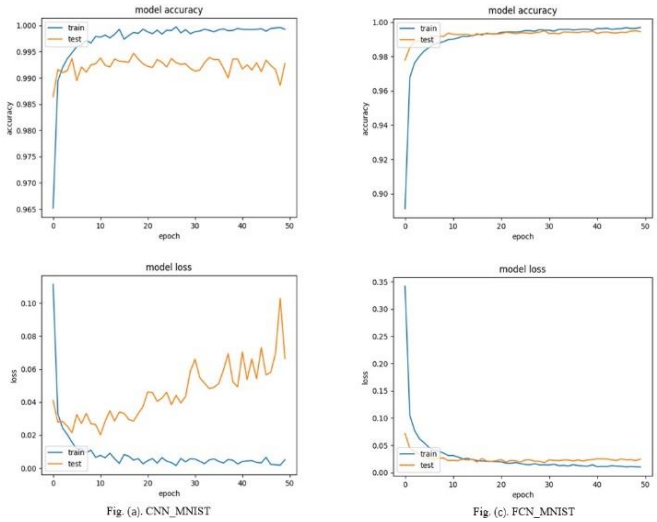

Fig. (a). CNN_MNIST Fig. (c). FCN_MNIST

Fig.(a), (c). CNN_MNIST, FCN_MNIST

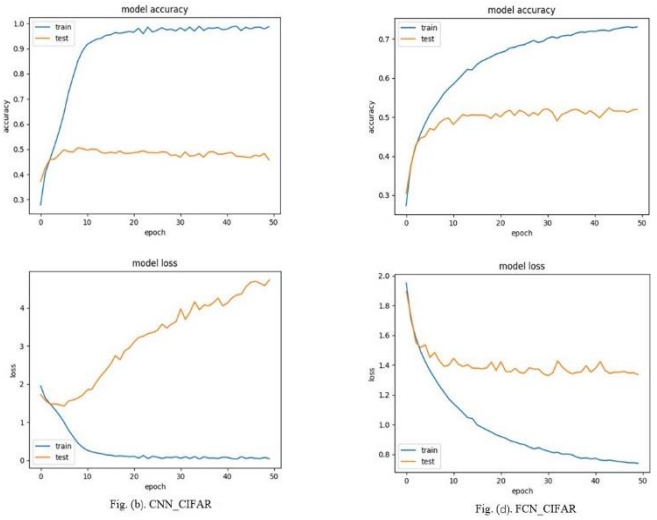

Fig. (b). CNN_CIFAR Fig. (d). FCN_CIFAR

Fig. (b), (d). CNN_CIFAR, FCN_CIFAR

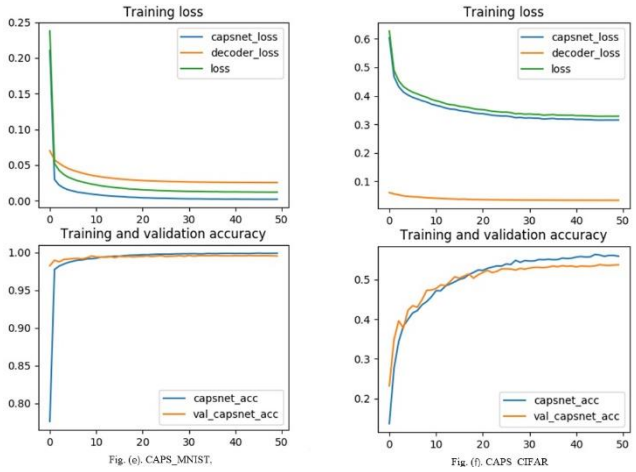

Fig. (e). CAPS_MNIST. Fig. (f). CAPS_CIFAR

Fig. (e), (f). CAPS_MNIST, CAPS_CIFAR

Fig. 10. Models' training results on MNIST and CIFAR-10.

First, let's focus on the MNIST data set.

As shown in Table I, the CNN model consumes the least time in the training process, then the FCN model and the CapsNet model consumes almost 30 times more time than the other two. In terms of accuracy, all three models are 99% more accurate but FCN is a little higher. As is shown in the parameter column, the FCN model uses the least parameters and the CapsNet model uses the most. The CapsNet model trained over 8.2 million parameters which are far more than the other two, and this explains why it takes so much time to train itself.

Intuitively view from the experimental results, these algorithms' training loss (CNN, FCN, CapsNet) reduced to less than 0.7% and achieved a more ideal training result after adequate training.

TABLE I: COMPARATIVE PERFORMANCES TESTED ON MNIST

item	time	loss	accuracy	parameter
CNN	277.30	0.0663279	0.9928	3430730
FCN	328.30	0.024775	0.9945	74362
Caps	8453.70	0.0231	0.9952	8215568

For the CIFAR-10 dataset, shown in Table II, it's obvious that none of the three models achieved very high accuracy. Both the losses of CNN and FCN are very high. However, although the CapsNet model's loss achieved 0.0346, which is almost the same as when it was trained on the MNIST dataset, its accuracy is still very low, only 53.70%.

TABLE II: COMPARATIVE PERFORMANCES TESTED ON CIFAR-10

item	time	loss	accuracy	parameter
CNN	310.37	4.730176	0.4577	4420170
FCN	332.18	1.336495	0.5198	77850
Caps	3246.82	0.0346	0.5370	11749120

We visualized the training process and showed them in Fig. 10 (a)-(f).

In Fig. 10 (a), (c) and (e), we can see that when trained on the MNIST dataset, all three models converged quickly when epoch < 4. After that point, the accuracy increased slowly and the loss almost stopped decreasing. For the CIFAR-10 data set, shown in fig b, d, and f, we can see they converge slower than training on the MNIST dataset and the accuracy of the test dataset almost stopped increasing when it reached 50%, indicating they might over-fitted. Thus, we did another

experiment in Section D whose result showed that our prediction is true.

This experimental data shows that the quality of the neural network is excellent and is an ideal method for processing image classification. The commonality of the three networks lies in the fact that they all emphasize the identification of the orientation, size and other attributes of their network structure and the generation of internal connections. Among them, CNN and FCN belong to the convolutional neural network, with characteristics of few parameters, fast training, high score, easy to migrate. It has great improvement compared with the traditional neural network method. The convolution layer is the key to achieve this ascension, which can be called the soul of the entire convolutional neural network.

As we mentioned above, CNN has lost a lot of information in the pooling layer. Thus, it is impossible to recognize the orientation of objects. To enhance the accuracy of image-feature-extraction, we constructed data sets which expanded the scale of our origin data set by downsizing and rotating. Many of these pictures were essentially the same, whereas CNN could not identify them. This is the predominant factor to the output.

Due to its full convolution network without a full connection layer (FC), FCN is based on CNN and is capable of dealing with different sizes of input. The deconvolutional layer permits the increase of the data's scale which contributes to a more accurate result. The skip structure of different depth layer results is combined to ensure robustness and accuracy of the output. Via the combination of three factors mentioned above, FCN compensates for the defects which exist in CNN's pooling layer. Meanwhile, since the images of the test set we selected and the images of the training set are approximately the same in scale, the performance of FCN won't be affected greatly. Hence, FCN performs better than CNN.

Higher recognition accuracy was also obtained on CapsNet. The dataset we created was carefully designed to make it be a pure shape recognition task, which means an object can be recognized from different perspectives. From this perspective, CapsNet beat the state-of-the-art CNN on the dataset.

The most significant factor contributing to the improvement is the addition of dynamic-routing.

A lower-level capsule tends to send its output to higher-level capsules whose activity vectors have a big scalar product with the prediction coming from the lower-level capsule. It is because of dynamic-routing that CaspNet can reflect the characteristics of layer abstraction and layer classification in the operation. Thus, its status in CapsNet is self-evident and needs no elaboration.

By researching the differences between the results of the two datasets, we can conclude that the three models can easily achieve high accuracy on a single-class dataset but a multi-class dataset, they perform not as well as we expected.

D. Training with CIFAR-100 for 300 Epochs

CIFAR-100 is similar to CIFAR-10, which is also consisted of ten different classes. The only distinction is that the ten classes of CIFAR-100 are divided into 20 super-classes. Each image comes with a fine tag (the class it belongs to) and a rough tag (the superclass it belongs to).

As we mentioned in section C, all three models that trained with CIFAR-10 datasets aren't accurate enough to distinguish the ten classes of objects. Thus, we chose CIFAR-100 to test their capability of multi-type image classification and trained each model with CIFAR-100 dataset for 300 epochs expecting they will be more accurate, and the results are shown in Table III and the model's loss and accuracy are shown in Fig. 11 and Fig. 12.

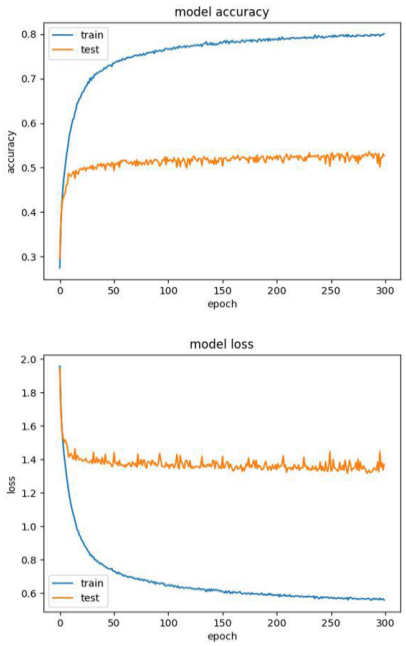

Fig. 11. CCN's performance on CIFAR-100 for 300 epochs.

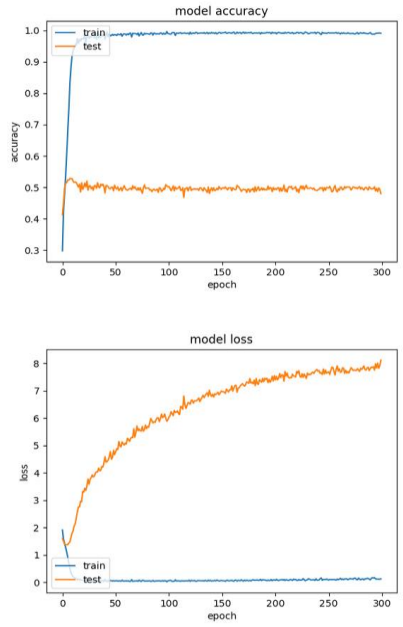

Fig. 12. FCN's performance on CIFAR-100 for 300 epochs.

As Table III showed, all three models didn't achieve high accuracy, indicating they over-fitted.

TABLE III: COMPARATIVE PERFORMANCES TESTED ON CIFAR-100 FOR 300 EPOCHS

item	time	loss	accuracy	parameter
CNN	1883.79	8.120001	0.48	4420170
FCN	2019.01	1.371521	0.527	77850
Caps	19432.09	0.0353	0.5512	8215568

V. Conclusion

In this paper, we evaluated the performances of the three efficient models on two datasets: MINST, CIFAR-10, which contain different types of objects. Via comparing the results, we found our models are good at single-class datasets but perform badly on multi-class datasets.

In the experiment, we found that even with a small number of hyper-parameters, CapsNet can still classify images efficiently in the experiment. Our model needs to be adjusted to achieve better performance on multi-class datasets.

We also found that the CNN and FCN models' performances are equivalent but the FCN model trains fewer parameters, which means it requires less memory consumption. CapsNet overcomes the limitations of losing large amounts of data information during the pooling phase of traditional CNNs, as well as realize higher resolution of image classification work. However, the model of CapsNet needs to train far more parameters and takes more time to train (around 10 times than the other two models). Generally speaking, the CNN and FCN model is better than the CapsNet model.

CapsNet is a brand-new method in image processing and its limitations are unavoidable in many practical applications. Although most of the complex scenario images are still done via CNN, it doesn't mean CapsNet will be useless in the future. This means, therefore, that more research, experiments, and tests are needed to unlock the full potential of this approach.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Xuefeng Jiang and Yikun Wang conducted the research and proposed the algorithm, Wenbo Liu programmed for the model. Shuying Li and Junrui Liu did the experiment; prepared the dataset. Yikun Wang and Wenbo Liu analyzed the data. All authors had approved the final version.

References

[1] R. M. Haralick and K. Shanmugam, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, pp. 610-621, 1973.

[2] L. Yann and Y. Bengio, "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, 1995.

[3] D. Chao *et al.*, "Learning a deep convolutional network for image super-resolution," in *Proc. European Conference on Computer Vision*, Springer, Cham, 2014.

[4] L. Jonathan, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[5] S. Sara, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in Neural Information Processing Systems*, 2017.

[6] M. Rinat and J. Carrillo, "CapsNet comparative performance evaluation for image classification," arXiv preprint arXiv:1805.11195, 2018.

[7] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. European Conference on Computer Vision*, Springer, Cham, 2014.

[8] K. Alex, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*. 2012.

[9] S. Dominik, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proc. International Conference on Artificial Neural Networks*. Springer, Berlin, Heidelberg, 2010.

[10] N. Feng *et al.*, "Toward automatic phenotyping of developing embryos from videos," *IEEE Transactions on Image Processing*, vol 14, pp. 1360-1371, 2005.

[11] S. Pierre *et al.*, "Overfeat: Integrated recognition, localization and detection using convolutional networks," arXiv preprint arXiv:1312.6229, 2013.

[12] P. H. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *Proc. 31st International Conference on Machine Learning (ICML)*, 2014.

[13] E. David, D. Krishnan, and R. Fergus. "Restoring an image taken through a window covered with dirt or rain," in *Proc. the IEEE International Conference on Computer Vision*, 2013.

[14] J. J. Tompson *et al.*, "Joint training of a convolutional network and a graphical model for human pose estimation," *Advances in Neural Information Processing Systems*, 2014.

[15] H. Martin, "'Spatial' relationships? Towards a reconceptualization of embeddedness," *Progress in Human Geography*, vol. 28, no. 2, pp. 165-186, 2004.

[16] K. M. He *et al.*, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904-1916, 2015.

[17] G. Ross, "Fast r-cnn," in *Proc. the IEEE International Conference on Computer Vision*, 2015.

[18] S. Niko *et al.*, "On the performance of convnet features for place recognition." arXiv preprint arXiv:1501.04158, 2015.

[19] H. Song *et al.*, "Learning both weights and connections for efficient neural network," *Advances in Neural Information Processing Systems*, 2015.

[20] C. Q. Xiang *et al.*, "MS-CapsNet: A novel multi-scale capsule network," *IEEE Signal Processing Letters*, vol. 25, no. 12, pp. 1850-1854, 2018.

[21] T. Y. Whye and G. E. Hinton, "Rate-coded restricted Boltzmann machines for face recognition," *Advances in Neural Information Processing Systems*, 2001.

[22] K. J. Eric and G. P. Abousleman, "Adaptive-rate coded digital image transmission," U.S. Patent No. 6,154,489, Nov. 28, 2000.

[23] M. D. Ward and K. S. Gleditsch, *Spatial Regression Models*, vol. 155, Sage Publications, 2018.

[24] B. Xu *et al.*, "Empirical evaluation of rectified activations in convolutional network," arXiv preprint arXiv:1505.00853, 2015.

[25] C. Nicholas and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. 2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017.

[26] Lin, Yuanqing *et al.*, "Large-scale image classification: Fast feature extraction and svm training," in *Proc. CVPR 2011*, IEEE, 2011.

[27] E.-S. Ahmed, E.-B. Hazem, and M. Loey, "CNN for handwritten arabic digits recognition based on LeNet-5," in *Proc. International Conference on Advanced Intelligent Systems and Informatics*, Springer, Cham, 2016.

[28] H. Song *et al.*, "Learning both weights and connections for efficient neural network," *Advances in Neural Information Processing Systems*, 2015.

Xuefeng Jiang received the B.E. degree of electronic Engineering from Northwestern Polytechnical university and the master's degree of software engineering from Northwestern Polytechnical University. He is currently an associate professor with School of computing, Northwestern Polytechnical University, Xi'an, Shaanxi, China. His research interest includes computer vision, pattern recognition, remote sensing and multimedia analysis.

Yikun Wang was born in Shaanxi Provence of China on November 13, 2000. In 2018, she entered Northwest Polytechnical University in Xi'an, Shaanxi Provence of China. She majors in computer science and technology.

Wenbo Liu was born in Shandong, China on January 25, 1999. He is an undergraduate of computer science and technology in Northwest Polytechnical University in Xi'an, China. He majors is in computer science and technology. He is one of the bronze medalists of ICPC Jiaozuo station, 2018 and silver medalist of ICPC Yinchuan station, 2019.

Shuying Li received the B.E. degree of automation from University of Science and Technology of China in 2005, and the Ph.D degree of digital circuit and system from Chinese Academy of Sciences in 2010. She is currently a professor with School of Automation, Xi'an University of Posts & Telecommunications, Xi'an, Shaanxi, China. Her research interest includes remote sensing, computer vision and pattern recognition.

Junrui Liu received the B.E. degree of computer science from Xi'an Jiaotong University in 2000, and the Ph.D. degree of computer application from Northwestern Polytechnical University in 2011. She is currently an associate professor with School of Computing, Northwestern Polytechnical University, Xi'an, Shaanxi, China. Her research interest includes computer vision, pattern recognition, machine learning methods and their related applications particularly in video surveillance, intelligent.