Assessing Four Neural Networks on Handwritten Digit Recognition Dataset (MNIST)

Feiyang Chen, Nan Chen, Hanyang Mao, Hanlin Hu

Novosibirisk Sate university

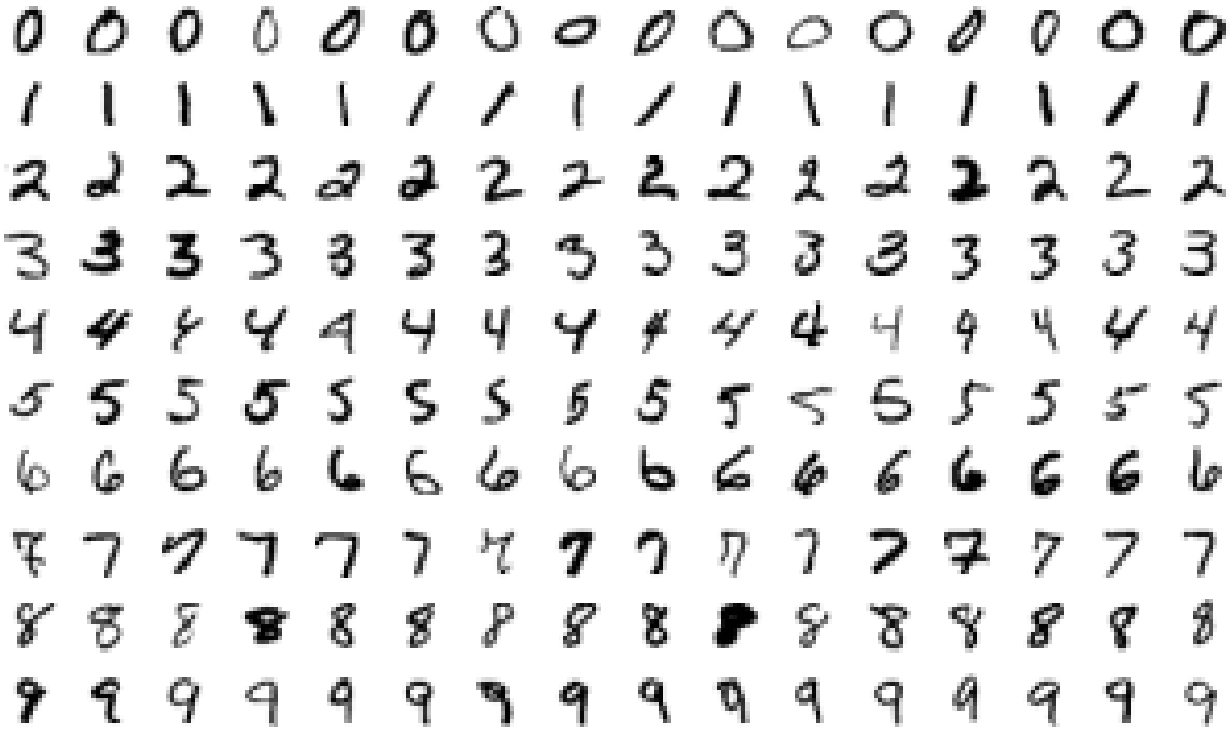
23-04-2020

Intoduction

- In this paper, our main goal is to compare four mainstream image recognition models on MNIST dataset with different division.
- These are CNN, ResNet and DenseNet respectively. These three models are already proved to have good performance in image recognition.
- They use CNN as a baseline model, and improve it through applying CapsNet to optimize our this model.
- They use the MNIST dataset for the test because the recognition of handwritten digits is a topic of practical importance.
- In order to make the model more generalizable in the field of image recognition, they randomly divided the MNIST dataset into different intervals.

Dataset

- The MNIST dataset is from the National Institute of Standards and Technology (MNIST).
- MNIST dataset totally contains 60,000 images in the training set and 10,000 patterns in the testing set, each of size 2828 pixels with 256 gray levels.



CNN

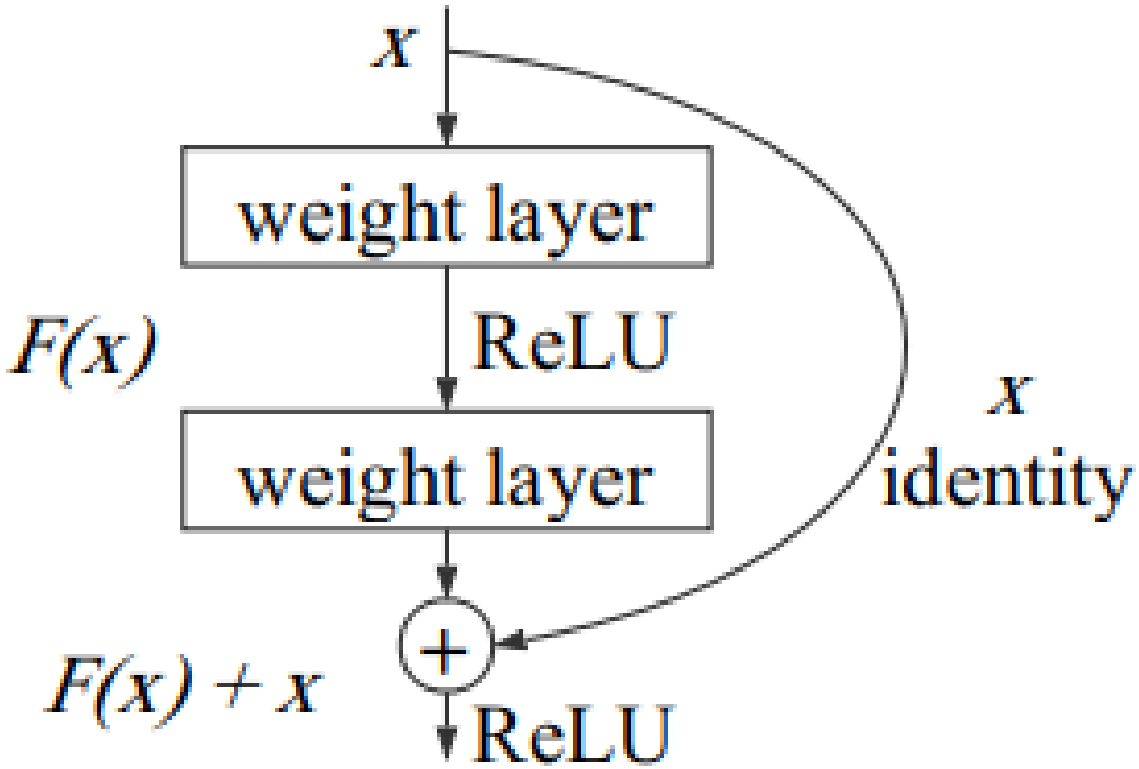
- CNN is a feed-forward artificial neural networks, most commonly applied to analyzing visual imagery.
- It consists of one or more convolutional layers and the top fully connected layer, and it also includes associated weights and a pooling layer.
- This structure allows the convolutional neural network to take advantage of the two dimensional structure of the input data, so it can give very good results in image recognition.

RESNET

- Deep convolutional neural networks have led to a series of breakthroughs for image classification.
- As the network depth increased, accuracy gets saturated and then degrades rapidly.
- ResNet is presented in 2017. It can reduce the train error while deepening the depth of the network, and solve the problem of gradient dispersion.
- ResNet can not only be very deep, but also has a very simple structure. It is a very small single module piled up, its unit module block

Continued

- $x_l = H_l * (x_l - 1) + (x_l - 1)$
- where l represents layer, xl represents the output of the l layer, Hl represents a nonlinear transformation. For ResNet, the output of the l layer is the output of the $l - 1$ layer plus the nonlinear transformation of the output of the $l - 1$ layer



DenseNet

- The basic idea of the DenseNet model is the same as that of ResNet, but it establishes a dense connection between all previous and subsequent layers.
- Its other major feature is feature reuse through the connection of features on the channel.

Experimental Setup

- They compare four models, three of which are mentioned before.
- The other is our retrofitting and improvement based on CNN model.
- They tested all the models using a workstation built from commodity hardware: dual GeForce GTX 1080 graphics cards, an i7-6800K CPU, and 64 GB of RAM.
- They use the Adam optimizer with TensorFlow default parameters, including the exponentially decaying learning rate, to minimize the sum of the margin losses.

Baseline

- Our model is based on a standard CNN with three convolutional layers, which is demonstrated to achieve a low test error rate on MNIST.
- The channels of three layers are 256, 256, 128 respectively. Each layer has 5×5 kernels and stride of one.
- The last convolutional layers are two fully connected layers of size 328, 192 after that is a 10 class softmax with cross entropy loss.
- The baseline model has two shortcomings. First, training a powerful CNN model requires a large number of training data. Second, in the pooling layer, CNN loses some of the information, which leads to the ignorance of interrelationships between different component.
- That is small changes in input, the output of CNN will be almost constant

Retrofitting

- In order to overcome these shortcomings of CNN, they try to introduce CapsNet to optimize the baseline.
- It uses capsules instead of neurons.
- The input and output of the capsule are highdimensional vectors, where the module length represents the probability of occurrence of an object, and the direction represents the position, color, size and other information.
- The output of the low-level capsules is used to generate a prediction through transformation matrices, which are then linearly integrated and passed into high-level capsules according to certain weights.
- The method of updating the weights is a dynamic routing algorithm, which compares the output of high-level capsules with the prediction of lowlevel capsules, and increases the input weights of low-level capsules with higher similarity until convergence.

Continued

- Through the capsule, we retain the information on the details of the picture.
- on the basis of accurately identifying the image, small changes in the image input will cause small changes in the output.

Continued

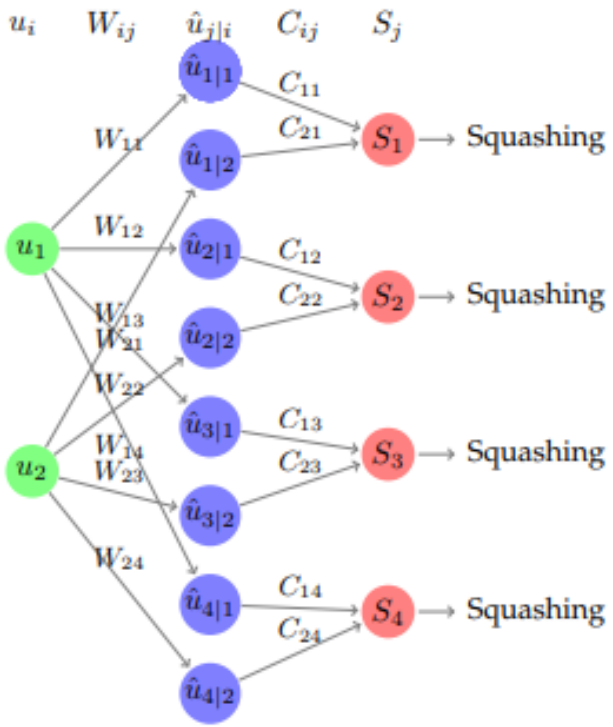


Fig. 3. Structure of CapsNet, where u_i is the input layer, W_{ij} is the weight matrix, $\hat{u}_{j|i}$ is the u_i's prediction to S_j, C_{ij} is the weight and S_j is the output layer. Squashing is the activate function as shown in Eq. 2.

$$v_j = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||}$$

CapsNet Architecture

- It consists of one convolutional layer and two capsule layers.
- The convolutional layer 1 has 256, 9×9 convolution kernels with a stride of 1 and ReLU activation.
- This layer extracts the basic features of the image, and then uses them as the inputs of the primary capsules layer (PrimaryCaps).
- The PrimaryCaps contains 32 capsules, which receives the basic features detected by the convolution layer, creating a combination of features.
- The output of PrimaryCaps is 6632 eight-dimensional vector.
- The last layer is digital capsules layer (DigitCaps), it has 10 digital capsules and each of which represents the prediction of number.

CapsNet Architecture

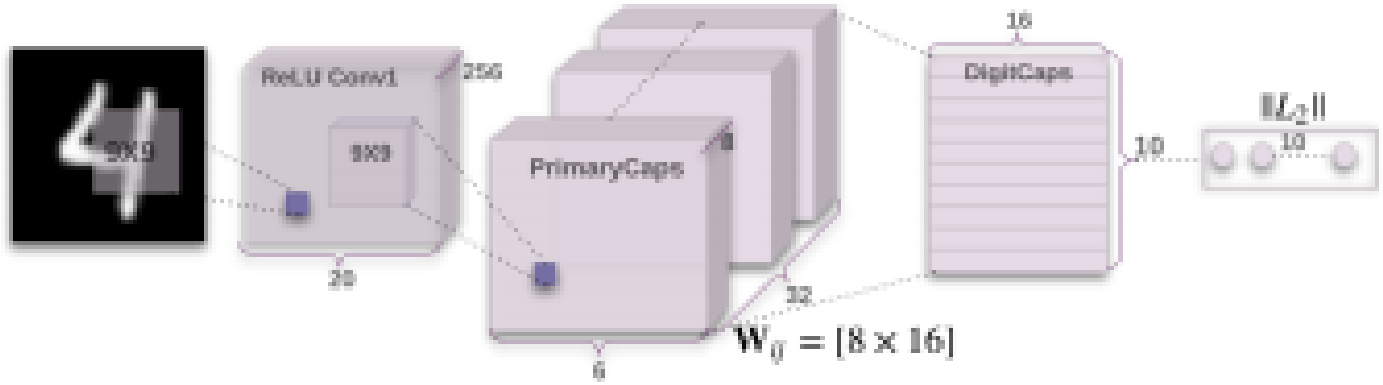


Fig. 4. A simple CapsNet with three layers. The convolutional layer extracts image features, PrimaryCaps integration the features, and DigitCaps output the prediction.

Result

TABLE 1
Results of experiment on divided datasets.

Accuracy(%) MNIST / Models	25%	50%	75%	100%
CNN	80.73	86.73	91.23	98.32
ResNet	79.46	90.55	93.78	99.16
DenseNet	82.57	89.24	94.20	99.37
CapsNet	87.68	97.12	98.79	99.75

Result

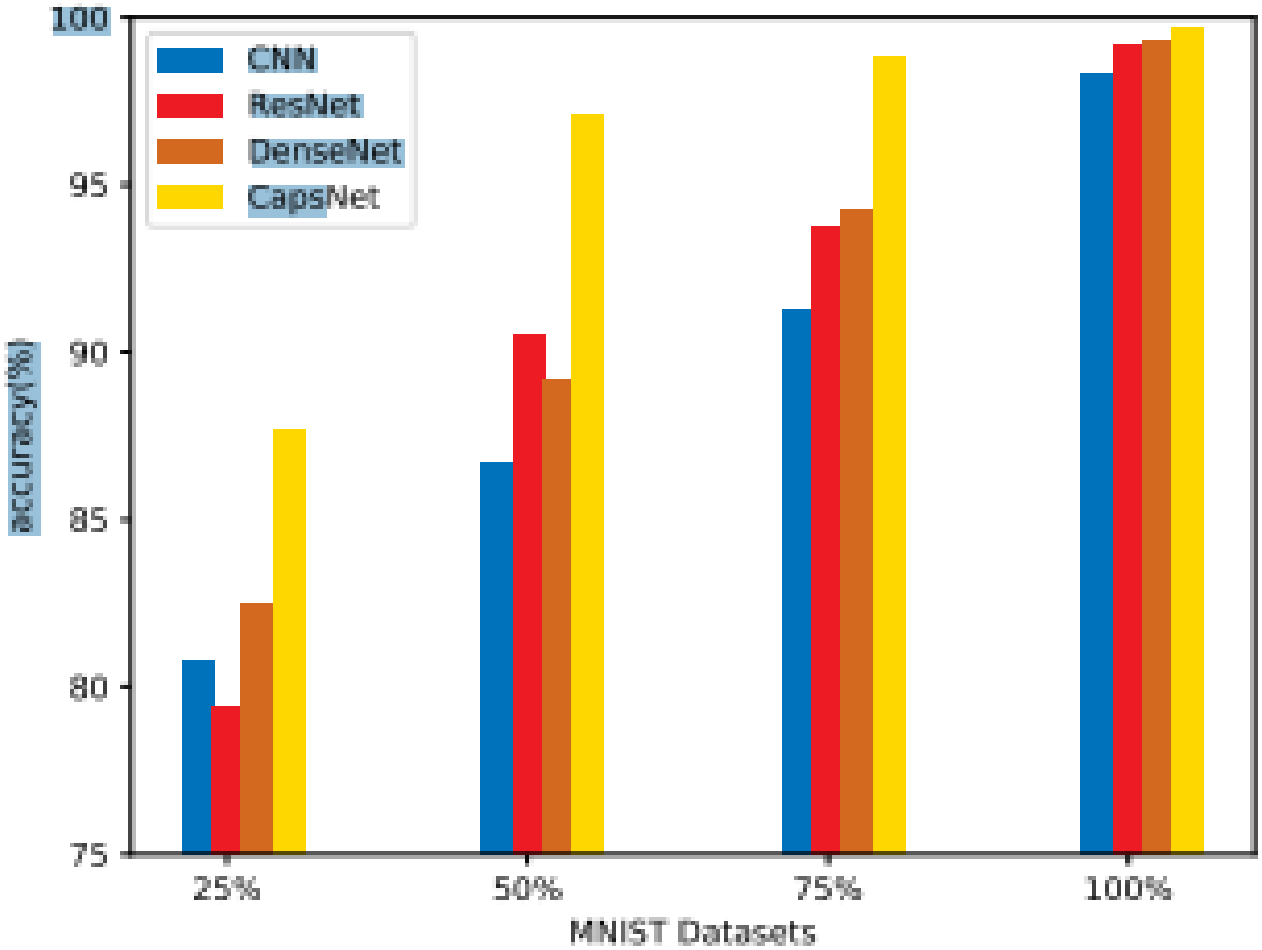


Fig. 5. The results for the four models across all divided datasets.

Conclusion

- The goal of this paper has been to discover which models perform better across divided MNIST datasets. We compared four models on MNIST dataset with different division, and showed that CapsNet perform best across datasets.
  Additionally, we also observe surprisingly that CapsNet requires only a small amount of data to achieve excellent performance.

THANK YOU