CapsNet Implementation for Overlapping Objects Segmentation

Zhiao Zhou
zz1749@nyu.edu

Jianghao Zhu
jz2575@nyu.edu

Chenyu Xu
cx463@nyu.edu

New York University Tandon School of Engineering
6 MetroTech Center, Brooklyn, NY 11201

Abstract

CapsNet is becoming a buzzword as it is said to be proficient at handling different types of visual stimulus and encoding things like pose (position, size, orientation), deformation, velocity, albedo, hue, texture and so on and even might replace Convolutional Neural Network in the future, which is the most widely used image classification learning algorithm. In this paper, we compared a CapsNet and a Convolutional Neural Network with very similar structure except for the capsules layers. And it turns out that CapsNet outperform CNN with an accuracy of 90% to 65%. However, their F1-scores are both around 0.6. We think CapsNet will be applied to more computer vision fields in the future but it still needs improving.

1. Introduction

Convolutional Neural Network has gained huge amount of attentions and successes in computer vision field. Deep Convolutional Neural Network has achieved significant improvement in accuracy which outperforms human recognition capability. Regardless that max pooling in Convolutional Neural Network improves performance quite well, there are still questions about information lost during the process. In addition, Convolutional Neural Networks do not quite address relationships between features, for example, relative spatial characteristics. A face with displaced eyes, nose, and mouths could still be recognized as a face by Convolutional Neural Network, but not by Capsule Network. In Hinton[7], he proposed Capsules Network which implements dynamic routing by agreement algorithm that achieved state of art performance with only three layers of network. According to[7], Capsules would be able to represent various properties of objects like pose, deformation, velocity etc. And Capsules Network is able to learn and detect variant representations of features without utilizing many layers for storing and representing these variances. In this sense, Capsules Network acts more like human brains that learning with less examples and comprehensive understanding. This is a new perspective in the computer vision field which brings more insights out from images using different methods instead of propagation for data fitting and prediction. Furthermore, Capsules Network works exceptionally well on highly overlapping objects segmentation comparing to other projects for objects that have low overlapping areas. Objects segmentation is now one of the most significant areas within the image analysis domain, which is necessary in a majority of scientific fields including face and object recognition, medical imaging, industrial imaging, engineering and technology.

2. Related work

Chang and Chen[3] built batch-normalized network in addition to convolutional networks leads to improved accuracy of image classification. Abadi et al[1] implemented large scale machine learning system on TensorFlow gives some guidance on building such network. Ba et al[2] have worked on easily 4% overlapping objects images and get a accuracy of 5%. Prochazka et al[6] addressed the most challenging obstacle due to noise in segmentation problems. Greff et al[4], 2016 have also tried to train a deep convolutional neural network to segment overlapping digits based on Google Map View. We can see that there are not so many prior researches on this domain and from those done we find that these learning algorithms cannot deal with badly overlapping objects or they require a mighty complex structure. However, Hinton et al[7], 2017 trained a simple 3 layer CapsNet to get the same or even better result as these state-of-the-art but complex learning algorithms which can be a breakthrough to the similar domain. Accordingly, we are going to train a CapsNet to test its performance on overlapping digits. And we will try to improve the model by adding batch normalization, regularization, optimizers and so on so that we can get a better result than Hinton[7]. The implementation of this study adopted from [5].

3. Problem Formulation

In this project, our aim is to distinguish 2 overlapped handwriting digits from MNIST database. To solve this problem, we should divide the problem into several sub-problems:

1. We need to distinguish a single handwriting digit from MNIST database. In this step, we need to acquire a higher accuracy;

2. We need to consider the similarity and difference between single-digit recognition and 2-digits recognition. To make codes more readable and concise, we should try to use the same structure to implement 2-digits recognition. That means we should try to use the same functions in single digit recognition while set parameters to distinguish two processes;

3. After adjusting functions in single-digit recognition, we need to complete the design of 2-digits recognition. In this step we should also test the accuracy, and try to acquire a satisfying result.

4. If the accuracy of 2-digits recognition is lower, we need to analyze the reasons, and try to use various solutions to improve the accuracy. After the optimization, we should acquire a satisfying result.

5. We could also implement the structure of regular CNN, and compare the performance of regular CNN and Capsule Module.

4. Technical Depth and Innovation

Convolutional Neural Network(CNN) is widely used to classify images which are very close to the original training set. However, if the test images have rotation, tilt or any other different orientation, CNN would perform poorly. However, usually this is solved by data augmentation in advance and by adding pooling layers. Whereas a CNN will still find it difficult to understand the rotation or proportion change and automatically adapt itself so that the position information inside the images will not be lost. So that is why the CapsNet is designed. The capsules within a CapsNet are mighty good at addressing various kinds of visual stimulus and encoding things such as position, orientation, deformation, texture and so on. The analysis of Capsule will be discussed in Part 5. G. Hinton[7] also compares their 3 layer CapsNet model with their baseline convolutional model which is the sequential attention model of Ba et al[2], 2014 achieves on a much easier task that has far less overlap (less than 4%). And it turns out that the CapsNet model achieves higher accuracy. Instead of training a same CapsNet as Hinton et al[7] do, we will try the state-of-the-art optimization techniques for tuning the hyper-parameters and improving the model such as using exponentially decayed learning rate, batch normalization, cyclical learning rate method and so on.

5. Methodology

In this part, we will mainly focus on the algorithm of the whole project, and discussed it in three aspects: the whole process from input to output; the module of Capsule and analysis of its structure; the optimization of the whole structure.

5.1. Whole Process

5.1.1 Mini-batch Generation

We use mini-batch algorithm to train and test. The initialization of batch size is 64. To reduce the overfitting, we could use data augmentation to increase the diversity of data set. Here we could just shift each pictures for a small random distance (0-2 pixels) in random direction. Since the train set is generated from MNIST, which only contains single digit in each picture, for overlapping case, we should overlap two shifted pictures to generate a new picture with two overlapped digits. Therefore, in each iteration, we need to retrieve two batches and overlap them after shifting operation.

5.1.2 Structure of Propagation

The process of propagation is much simpler here than regular CNN module. It has been described in [7] as Figure 1. For each mini-batch, the size is 64 * 28 * 28 * 1. The first layer is a convolution layer. The size of normalized weight matrix is 9 * 9 * 1 * 256 with stride as 1, and that of normalized bias matrix is 256. The output is a 64 * 20 * 20 * 256 matrix. In this layer, we could detect basic features like edge and angular point. ReLU activation function is introduced after convolution. The second layer is also a convolution layer. The size of normalized weight matrix is 9 * 9 * 256 * 256 with stride as 2, and that of normalized bias matrix is 256. The output is a 64 * 6 * 6 * 256 matrix. In this layer, we could detect some small features. Instead of using ReLU activation function as the non-linearity activation function, we use squash function. Before using squash function, we reshape the matrix as 64 * 6 * 6 * 32 * 8 * 1 * 1. That means for each pixel, we divide 256 channels into 32 groups, and each group contains 8 channel forming a vector. We will do squash for each vector. The squash function will be discussed in 5.1.3. After squash, the size of matrix is 64 * 6 * 6 * 32 * 8 * 1 * 1. And that is the input of capsule. The next step is the Dynamic Routing, which is the kernel of the Capsule. We will discuss it in 5.1.4. After that, we could acquire the prediction result for current mini-batch. The output is a 64 * 10 * 16 matrix.

5.1.3 Squash Function

The equation of squash function is shown as (1):

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1+||\mathbf{s}_j||^2}\frac{\mathbf{s}_j}{||\mathbf{s}_j||} \tag{1}$$

This function will keep the direction of original vectors, while it will squash the length of the vectors. The length will always be smaller than 1. When we use vectors to describe the features, spatial angle is used for describe the relationship between different features. Therefore, vectors could describe more complicated relationship than scalar. While length of vector works as the scalar in regular CNN module. The larger of a scalar, the higher probability of existing of that feature. In this way, we use squash function to squash the length, and make length to represent the existing probablity of that feature.

5.1.4 Reconstruction & Loss function

The aim of reconstruction is to distinguish the difference between real label and predicted ones. The process of reconstruction has been described in [7] as Figure 2. There are 3 layers: First, we adjust the prediction into 64 * 160 matrix; Second, we generate first fully connected layer with size 64 * 512; Third, generate the second fully connected layer with size 64 * 1024; Forth, generate the third fully connected layer with size 64 * 784; Last, reshape the matrix into 64 * 28 * 28 * 1. When we reconstruct the input pictures, there are two methods to calculate the difference. The first one is to calculate the Euclidean distance between each reconstructed picture and the original pictures derived from. Then we calculate the mean. Another way is to use:

$$L_c = T_c\max(0, m^+ - ||\mathrm{v}_c||))^2 + \lambda(1-T_c)\max(0, ||\mathrm{v}_c|| - m^-))^2 \tag{2}$$

to calculate the loss. The result for both loss equation has 64 as the size. Then we need to combine these two loss to form the loss function. Here we introduce a mask function. For non-overlapping case, the function generates 64 size ones matrix. For overlapping case, the function will calculate the number of digits distinguished for each picture, then minus 1. The result is a 64 size matrix. Now we could use:

$$\mathit{Loss1} = \frac{\sum \mathit{Loss} * \mathit{mask}}{\sum \mathit{mask}} \tag{3}$$

To calculate the new loss for each previous loss. Then use

$$\mathit{Loss} = \mathit{Loss2} + 0.0005 * \mathit{Loss1} \tag{4}$$

to acquire the final loss function. Now, we could use Adam Module to do the back propagation and minimize the loss we have acquired.

5.1.5 Accuracy

For each iteration, we fetch a mini-batch train set, train it. Then we use a mini-batch test set to test the accuracy. The process of the test is the same as propagation of training: First, we acquire the prediction result with 64 * 10 * 16 size. Second, we calculate the length for each higher-layer capsule, and acquire a 64 * 10 size matrix. Then, for non-overlapping case, we take the index of largest value for each picture, and compare it with the correct label of that picture; for overlapping case, we take the index of largest two value for each picture, and fetch the label of original two pictures, to check whether the predicted labels contains the correct ones. In this way, we could count the accuracy for the whole mini-batch, and that is the accuracy for current iteration.

5.2. Capsule & Dynamic Routing

It is usually difficult for regular CNN module to achieve a satisfying prediction result when we are dealing with rotated pictures. That is because regular CNN module never consider the relation between each features and max-pooling layer will lose the feature if rotated angle is too large. Therefore we cannot distinguish the same feature in a rotated picture. Dynamic Routing is the solution that improves the performance of regular CNN module. It could not only guarantee the rotational invariance, it could keep the affine invariance. In regular CNN module, the feature is a scalar, and we cannot describe the spatial relation between each feature. If we adjust each feature as a vector, it would be much easier for us to describe this relation. Here, each vector is a capsule. The input vector could also be called as lower-layer capsule. The output is higher-layer capsule. For each picture, there is 10 * 16 higher-level capsules. In fact, 10 is the number of classification, and each classification could also be recognized as a larger feature. 16 is the dimension for each output vector / higher-layer capsule. We could choose any dimension and 16 could acquire the best performance. There are 3 steps in this part and last 2 are Dynamic Routing. The first step is affine transform:

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathrm{u}_i \tag{5}$$

It considers the relation between lower-layer capsules and higher-level capsules. That is the reason that Dynamic Routing could keep affine invariance.Here matrix $\mathbf{W}_{ij}$ is those relations. The size of $\mathbf{W}_{ij}$ is 1 * 6 * 6 * 32 * 8 * 10 * 16, which means for each element of each vector, we need to consider the relations with each element for all output capsules. After calculating dot product with lower-layer capsule, we acquire a 64 * 6 *6 * 32 * 8 * 10 * 16 matrix. Then we sum the elements for each vector and acquire a 64 * 6 * 6 * 32 * 1 * 10 * 6 vector. It shows the prediction for output feature by each input vector. We could consider this step as another convolution layer. Not all predictions are useful. We need a scalar weight matrix to determine which predictions are more important and which are not. The size is 1 * 6 * 6 * 32 * 1 * 10 * 1. We initialize this matrix as constant 0, and it could adjust to suitable value within 3 iterations. Softmax function could be added on this weight matrix. Therefore, in second step, we do dot product and acquire the suitable prediction for each input vector. Then we need to sum all predictions in the same pictures.

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i} \tag{6}$$

Now we have acquired the prediction for each higher-layer capsule by each picture. The result is a 64 * 1 * 1* 1 * 1 * 10 * 16 matrix. The last step is to squash each output vector. The prediction result is a 64 * 10 * 16 matrix. We have mentioned that we should do iterations here to update the weight matrix in step 2. The process has been described in [7] as follow procedure 1. After 3 iterations, the module could acquire a satisfying prediction result.

5.3. Optimization

The original paper does not use some widely used optimization methods such as batch normalization, exponentially decayed learning rate. Accordingly, based on the architecture in [7], we tried batch normalization and exponentially decayed learning rate to see if there would be any improvement. The results will be shown in Result section. In addition, in order to prevent overfitting, we used an additional L2 regularization term to the loss function.

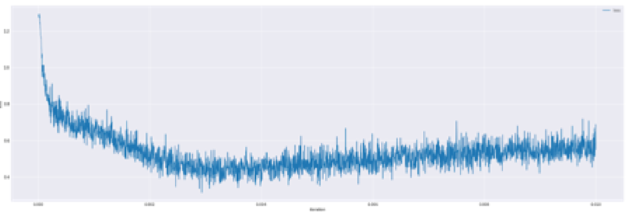
Figure 1. Prediction Loss

At last, we tested a new way called cyclical learning rate method to find our best initial learning rate. This method is proposed by Leslie N. Smith 2017. We first initiated our learning rate with 0.00001 and then let it increase gradually by every epoch and then we will get the image of loss function by epoch as shown in Figure 1. We can see that when learning rate goes up to 0.00075, the slope of the curve became the biggest so we chose this as our initial learning rate. And it turns out to perform well.

6. Architecture and Design

The structure of the whole process could be divided into two modules: mini-batch generation and Capsule. For Capsule module, there are several sub-modules:, two convolution layer, dynamic routing, picture reconstruction and loss function calculation, accuracy calculation.

6.1. Mini-batch Generation Module

We use mini batch algorithm to divide train set into several batches. Each batch is the input in each training iteration. For overlapped-digits recognition, we should also use an extra function to generate the overlapped images from MNIST database. In addition, the data augmentation is added here to reduce overfitting.

6.2. Capsule Module

Capsule module contains several sub-modules and we design it as a class. First, there are two convolution layer. Batch normalization is used after each layer to make the learning algorithm more stable and accurate. Second, it is the Dynamic Routing part. After that, we have acquired the prediction result. It is the final output when we input test image. The first two sub-modules have been set as a function. Third, we need to reconstruct the original picture and calculate the loss function. Here we add an L2 regularization term to loss function to reduce overfitting. When we determine the loss function, we could use Adam module to complete back propagation process and update the parameters. Besides, We try to find the best learning rate according to a paper published in 2015 on picking the best learning rate using cyclical learning rates[8]. Basically, we set a initial learning rate of 1e-5 and while training the learning algorithm we increase it by every iteration until the loss is not decreased anymore. We then get a figure of the by learning rate. At last, a learning rate that correspondingly has a large slope and a mighty low loss Forth, we need to calculate the accuracy of each iteration. It is useful for us to observe the performance of the whole structure, and after a large number of iterations, we should acquire a satisfying result.

6.3. Whole Structure

The architecture of this project, along with sub-structure of each steps could be organized as a flow chart shown in Figure 2:

7. Results

First, after training both of a CapsNet and a pure CNN learning algorithm, we found that CapsNet outperforms CNN from quite a few aspects. And to make it fair, in the CNN algorithm, we did not change anything except the last capsule layer which we changed to a convolutional layer. From Figure 3 and Figure 4 which is the curve of their loss function by epoch respectively, we can see that the CapsNet comes to the optima very shortly when CNN is still improving. In addition, from Table 1, we can be informed that a CapsNet is more powerful and more accurate than

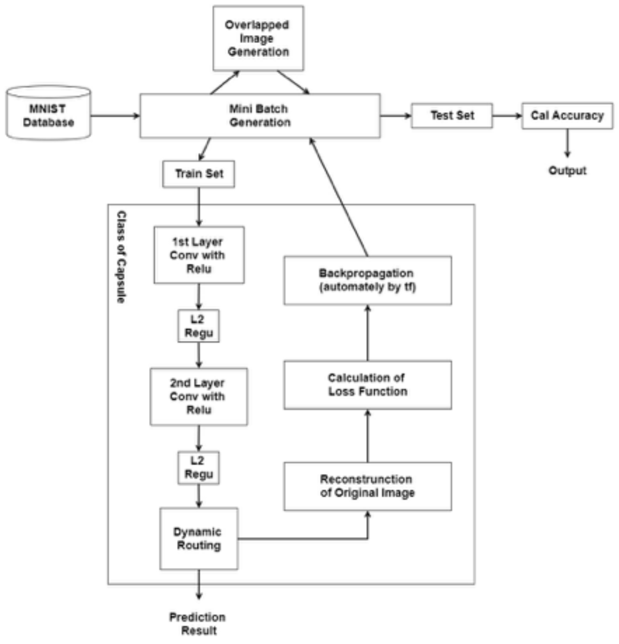

Figure 2. Project Architecture

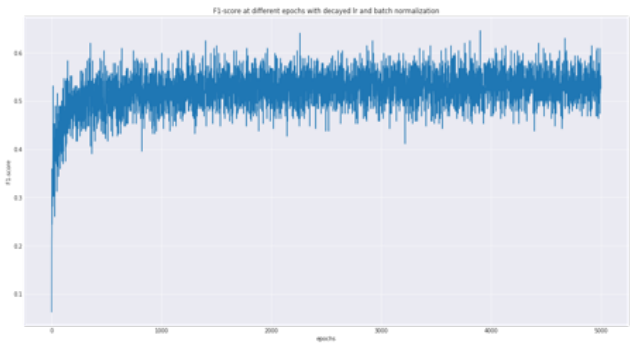

Figure 3. F1-score w/BN and Decay LR of CapsNet

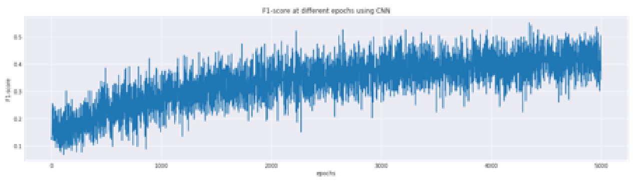

Figure 4. F1-score of CNN

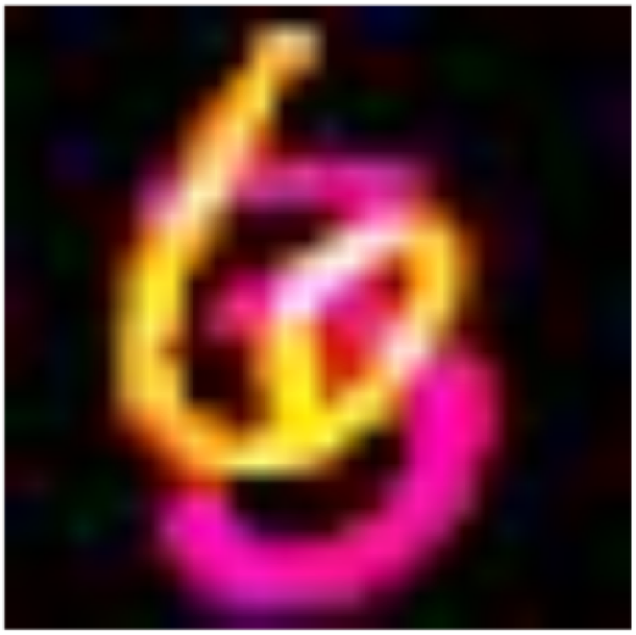

Figure 5. Prediction of Overlapping Digits

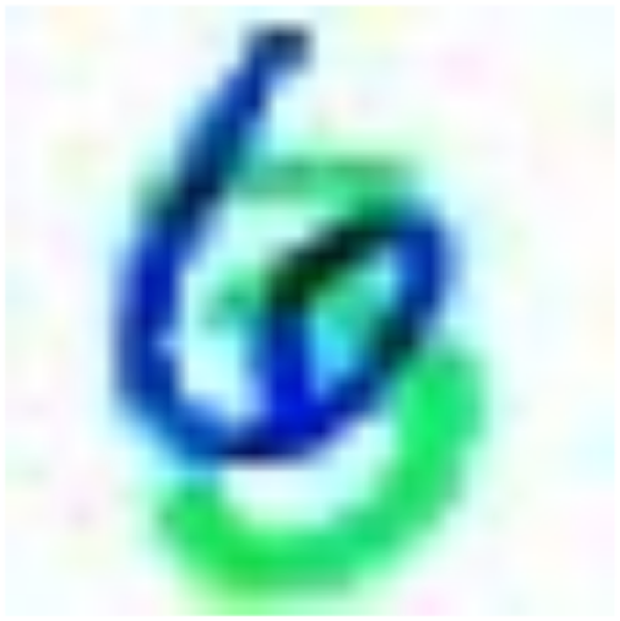

Figure 6. Prediction of Overlapping Digits

a CNN no matter of accuracy rate or the F1-score. This corresponds to [7] indicating that a CapsNet has more powerful ability at handling different types of visual stimulus and encoding things like pose (position, size, orientation), deformation, velocity, albedo, hue, texture and so on. And as shown in Figure 5, at most of time CapsNet can successfully detect two highly overlapping digits by analyzing the pixels of the digits against the background. And we also noticed that the original paper did not use some optimization such as batch normalization or exponentially decayed learning rate, so it only got an accuracy of 85% while we tested how the performance would be with or without those two optimization as shown in Table 1. It turns out that both of them are contributing to the model?s performance and batch normalization is providing a higher marginal utility than an exponentially decayed learning rate.

Learning Algorithm	Accuracy(5k ep.)	F1-score
4-hidden-layer CNN	65%	0.50
CapsNet	90%	0.60
CapsNet w/o BN	86%	0.55
CapsNet w/o Decayed LR	88%	0.58

Table 1. Performance Comparison

Black Background	White Background
6 (p:0.86183)	1 (p:0.19616)
3 (p: 0.63528)	2 (p:0.19542)

Table 2. Prediction Comparison with Black and White Background

8. Discussion

From the implementation of Capsules Network, the concept of capsules seems promising. The idea of using vectors to represent instantiation parameters of objects is evolutional. However, it is still an open exploration area with more research to be done. Dynamic routing algorithm makes it possible for a neural network to learn and predict without relying on back propagation for updating weights. Comparing to max pooling, dynamic routing transit best information from current layers to next layer instead of simply passing max value of certain area.

In addition, although CapsNet shows us its power and its possibility to replace CNN in the future, there is still some disadvantages in this project. First, we can see from Table 1 that the F1-score after 5k iteration is only 0.6 which is not very high which could result from the fact that the learning algorithm is having low precision rate. Additionally, when testing the demo application, we found that CapsNet is adapted to the background color of the input images. When we tried to input the same image in Figure 6 but only with white background, it turned out that CapsNet could not detect the two digits correctly and the probability it gets for all of the digits becomes very small. The prediction comparison of Black background and white background can be seen from Table 2. It evidently shows the weakness of the algorithm which should be developed in the future work.

9. Conclusion

CapsNet is truly a new and powerful learning algorithm for classification. Its performance over Convolutional Neural Network convince people of its prosperity in the future. And there are still a lot of unknown fields that CapsNet may apply to such as object localization/segmentation. And it?s still worth improving.

10. Github repository

`https://github.com/zhiaozhou/DL_Project_CapsNet`

References

[1] M. Abadi, A. Agarwal, and et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2016.

[2] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. 2014.

[3] J. Chang and Y. Chen. Batch-normalized maxout network in network. *arXiv*, 2015.

[4] J. Goodfellow, Y. Bulatov, and et al. Multi-digit number recognition from street view imagery using deep convolutional neural networks. 2013.

[5] laodar. a tensorflow implementation for capsnet.

[6] A. Prochazka and M. Yadollahi. Image segmentation for object detection. 2011.

[7] S. Sabour, N. Frosst, and G. Hinton. Dynamic routing between capsules. 2017.

[8] L. N. Smith. Cyclical learning rates for training neural networks. 2017.