An Experimental Study on Capsule Networks

Nitin Kumar and Shruti Jadon

June 12, 2019

An Experimental Study on Capsule Networks

Nitin Kumar
University of Massachusetts
Amherst
nitinkumar@umass.edu

Shruti Jadon
University of Massachusetts
Amherst
sjadon@umass.edu

Abstract

In this work we perform experiments with the recently published work on Capsule Networks. Capsule Networks have been shown to deliver state of the art performance for MNIST and claim to have greater discriminative power than Convolutional Neural Networks for special tasks, such as recognizing overlapping digits. The authors of Capsule Networks have evaluated datasets with low number of categories, viz. MNIST, CIFAR-10, SVHN among others. We evaluate capsule networks on two datasets viz. Traffic Signals, Food101, and CIFAR10 with less number of iterations, making changes to the architecture to account for RGB images. Traditional techniques like dropout, batch normalization were applied to capsule networks for performance evaluation.

Keywords: Convolutional Neural Networks; Capsule Networks; Routing-by-agreement; Dropout

1. Introduction

The invention of Convolutional Neural Networks led to a drastic change in the vision community. CNN's were able to learn transition invariant features with far less parameters as compared to Fully Connected Networks. CNN's however lack the ability to learn relative spatial relationships between components in an image and hence do not generalize well to different viewpoints for images. This serves as one of the motivations behind the recent research in [1].

The main component of a CNN is a convolutional layer.It detects important features in the image pixels. Layers that are deeper (closer to the input) will learn to detect simple features such as edges and color gradients, whereas higher layers will combine simple features into more complex features. Finally, dense layers at the top of the network will combine very high level features and produce classification predictions. Nowhere in this setup there is pose (translational and rotational) relationship between simpler features that make up a higher level feature. CNNs approach to solve this issue is to use max pooling or successive convolutional layers that reduce spatial size of the data flowing through the network. This increases the field of view of higher layers neurons, thus allowing them to detect higher order features in a larger region of the input image. Max pooling help convolutional networks work surprisingly well, but leads to loss of positional information among features.

Capsule networks aim to overcome the shortcomings of CNNs by explicitly modelling spatial relationships among different components of an image. Capsule networks encode various properties (pose, velocity, hue, texture etc.) of a particular entity in an image by means of a "capsule". A layer in a Capsule network is a collection of such capsules. A novel routing-by-agreement algorithm is employed using which capsules in a lower layer propagates its output selectively to capsules in higher layers. This iterative routing process enable higher level capsules to perform the job of assembling lower level features of entities into higher level features of entities while preserving entity spatial information.

2. Capsule Networks

The authors of [1] subscribe to the notion that the human visual system assimilates visual stimuli by means of performing the inverse of image rendering; visual information received by the eyes is deconstructed into a hierarchical representation of the world (like a parse tree). Visual recognition is then a process of matching this parse-tree structure with learned representations stored in the brain. In the human brain, learned representations are viewpoint invariant. Hence humans are very good are recognizing objects presented from different viewpoints.

It is difficult to perform viewpoint invariant object detection with CNN's. This is because CNNs do not inherently model relative spatial relationships between components in an image. For CNN's to perform viewpoint invariant object detection, an large number of spatial features (very deep networks) and large number of labelled examples are required. Capsule Networks explicitly model these spatial relationships by means of capsules and routing-by-agreement.

Capsules are groupings of neurons within a traditional neural network layer. Contrasting with a typical convolutional layer, a capsule network layer is essentially several convolutional layers nested in a single layer. Each nested convolutional layer is equivalent to a capsule. Capsules thus replace the scalar-output feature detectors of CNNs by vector-outputs. This enables capsules to encode various properties of particular entities in an image such as pose (translation and rotation), velocity, deformation, hue, texture etc. In [1], the authors implement capsules so that the magnitude of each capsule's output calculates the probability of existence of the entity that the capsule represents. This is achieved by a novel "squashing" non-linearity that scales down low magnitude vectors closer to 0 and high magnitude vectors closer to 1.

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

where v_j is the vector output of capsule j and s_j is its total input.

$$s_j = \sum c_{ij}\hat{u}_{j|i}$$

The total input s_j to a capsule is a weighted sum over all prediction vectors $\hat{u}_{j|i}$, from capsules in the lower layer and is produced by multiplying the output u_i of a lower level capsule by a weight matrix W_{ij}. The weight matrix W_{ij} encodes part-whole relationships among entities detected by lower level capsules. For example, the W_{ij} corresponding to a capsule that detects the digit 7, learns the length of the horizontal and vertical edges that compose a 7 and the angle between them.

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

The routing-by-agreement algorithm dictates the propagation of outputs from a child capsule to a parent capsule. This is done by computing the coupling coefficient c_{ij} between every child and parent capsule pair (i, j) in an iterative fashion. The coupling coefficients are updated by computing the scalar product b_{ij} of the prediction vector with the output of a possible parent v_j. The coupling coefficients are updated by simple adding its value to its corresponding b_{ij}

$$b_{ij} = v_j \cdot \hat{u}_{j|i}$$

$$c_{ij} = c_{ij} + b_{ij}$$

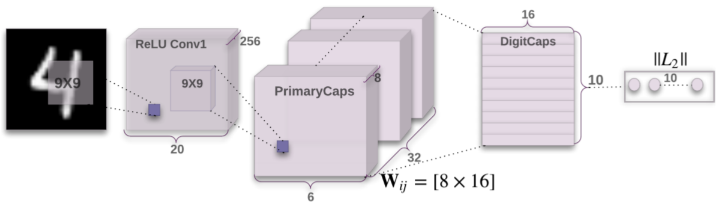

Figure 1. Architecture of Capsule Networks

For prediction vectors $\hat{u}_{j|i}$ that "strongly agree" with each other (vectors having similar orientation and high magnitudes), the corresponding s_j and v_j for parent capsule j will have large magnitudes. This in turn boosts the coupling coefficients c_{ij} of child and parent capsule pair (i, j). The routing algorithm thus establishes strong connections between children and parent capsules such that the orientation of u_i of children capsules help explain the orientation of v_j of a parent capsule.

The loss used by the authors in [1] is a combination of margin loss and reconstruction loss. The reconstruction loss is used as a regularizer. The margin loss (for MNIST) is defined by

$$L_k = T_k max(0, m^+ - ||v_k||)^2 + \lambda(1-T_k)max(0, ||v_k|| - m^-)^2$$

where L_k is the loss for each digit class k. T_k is 1 iff a digit of class k is present. $m^+ = 0.9$ and $m^- = 0.1$ enforce margins for correct class probabilities; $\lambda = 0.5$ is the penalty constant. The total loss is the sum of the loss of all digit capsules.

The authors of [1] employ a reconstruction loss as a regularization technique. During training, only the activity vector of the correct digit capsule is fed into a decoder consisting of 3 fully connected layers that model pixel intensities. The sum of squared differences between the output of the decoder and the input image pixel intensities is used as the regularization loss. This loss is scaled down by 0.0005 so that it does not dominate the margin loss during training.

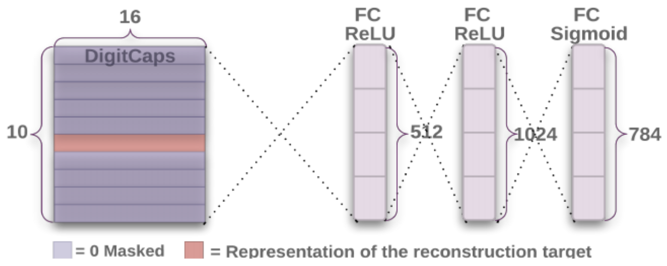

Figure 2. Decoder network for digit reconstruction

Procedure 1 Routing algorithm.

```
1: procedure ROUTING($\hat{\mathbf{u}}_{j|i}$, $r$, $l$)
2:     for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3:     for $r$ iterations do
4:         for all capsule $i$ in layer $l$: $\mathbf{c}_i \leftarrow \mathtt{softmax}(\mathbf{b}_i)$
5:         for all capsule $j$ in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
6:         for all capsule $j$ in layer $(l+1)$: $\mathbf{v}_j \leftarrow \mathtt{squash}(\mathbf{s}_j)$
7:         for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
       return $\mathbf{v}_j$
```

Figure 3. Routing Algorithm

3. Datasets

Capsule networks were evaluated on the following datasets.

- Traffic Signals: It consists of 50,000 images and 43 categories. The training data was divided into training, test and validation sets having the following proportions 50%, 33%, and 17% respectively.
- Food 101 Dataset: This data set consists of 101 food categories, with 10,099 images. For each class, 250 manually reviewed test images are provided as well as 750 training images.
- CIFAR10 Dataset : This data set consists of consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Figure 4. Sample images from Traffic Signals and Food101

4. Experiments

The following architecture was used for evaluating capsule networks

4.1. Main network

- Convolutional Layer: 9x9 kernel, stride 1, filters 256, activation RELU
- Optional dropout layer
- Primary Capsule Layer: 5x5 kernel, stride 2, capsules 16, capsule dimensions 16D, activation RELU
- Final Capsule Layer: capsules 43, capsule dimensions 32D
- Loss: Margin Loss

4.2. Decoder network

- Final Capsule Layer: capsules 43, capsule dimensions 32D
- FC layer: input neurons 43, output neurons 400, activation RELU, resize output to 5x5x16, upsample to 8x8 image
- Convolutional Layer: 4x4 kernel, stride 3, filters 4, activation RELU, upsample to 16x16 image
- Convolutional Layer: 8x8 kernel, stride 3, filters 8, activation RELU, upsample to 32x32 image
- Convolutional Layer: 3x3 kernel, stride 3, filters 3
- Loss: Reconstruction loss

The learning rate was set to 0.0001 and dropout rate to 0.7. We performed 1 iteration of routing-by-agreement. For the Food101 dataset, the Final Capsule Layer had 101 capsules instead of 43.

The loss convergence and accuracy graph for the Traffic Signal dataset are listed below.

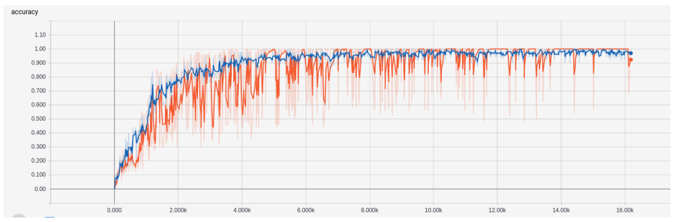

Figure 5. Accuracy (Orange-Training, Blue-Validation)

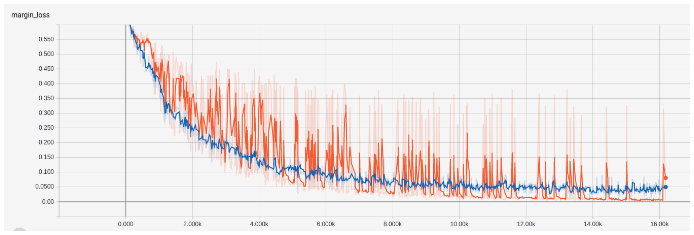

Figure 6. Margin Loss (Orange-Training, Blue-Validation)

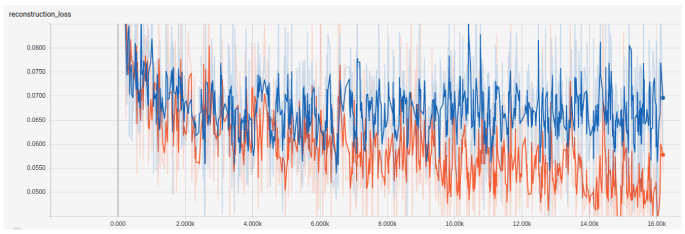

Figure 7. Reconstruction Loss (Orange-Training, Blue-Validation)

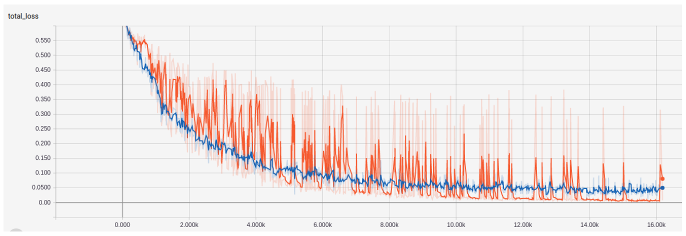

Figure 8. Total Loss (Orange-Training, Blue-Validation)

Later, we trained the same architecture for CIFAR-10, with only 60 epochs to check the convergence and accuracy for less number of epochs. The loss convergence and accuracy graph looks for CIFAR 10 dataset is listed below.

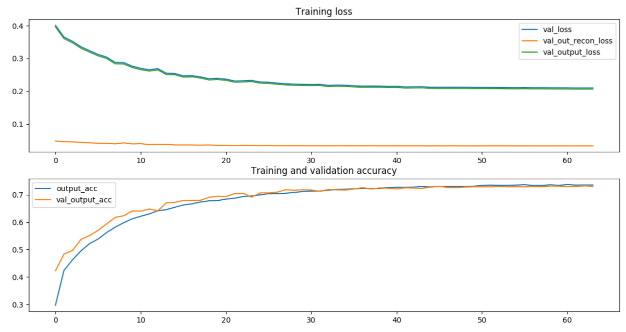

Figure 9. Loss and Accuracy Curve

4.3. Results

Results for both dataset are documented below

Table 1. Reported accuracy for different datasets

Dataset	Dropout	Train Acc	Test Acc	Iterations
Traffic Signal	-	100%	90.57%	16K
	0.7	100%	96.76%	16K
Food 101	-	41.7%	10.53%	10K
	0.7	33%	5.49%	10K
CIFAR10	-	94.58%	77.34%	600

The reconstructed images for the Traffic Signal dataset and CIFAR10 are displayed below. Although reconstructions are a little hazy, the network was able to learn features well enough to get good accuracy on the dataset. When we tried batch normalization, it was facing issue similar to vanishing gradient, as the architecture itself was normalizing outputs of capsules using squashing function. So, we can conclude that, unless we intend to use multiple CNN layers inside one capsule, it is not necessary to use Batch Normalization. We also tested the application of dropout, it boosted the performance of the model by almost 7.0%.

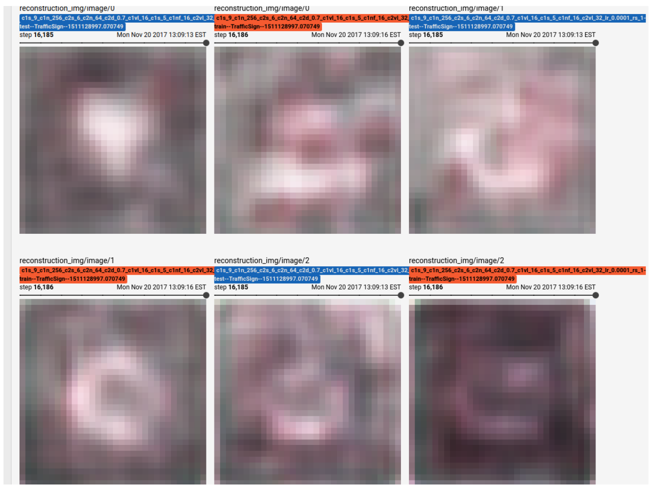
reconstruction_img/image/0
reconstruction_img/image/0
reconstruction_img/image/1
step 16,185
step 16,186
step 16,185
Mon Nov 20 2017 13:09:13 EST
Mon Nov 20 2017 13:09:16 EST
Mon Nov 20 2017 13:09:13 EST
reconstruction_img/image/1
reconstruction_img/image/2
reconstruction_img/image/2
step 16,186
step 16,185
step 16,186
Mon Nov 20 2017 13:09:16 EST
Mon Nov 20 2017 13:09:13 EST
Mon Nov 20 2017 13:09:16 EST

Figure 10. Reconstructed Images of Traffic Signal Dataset

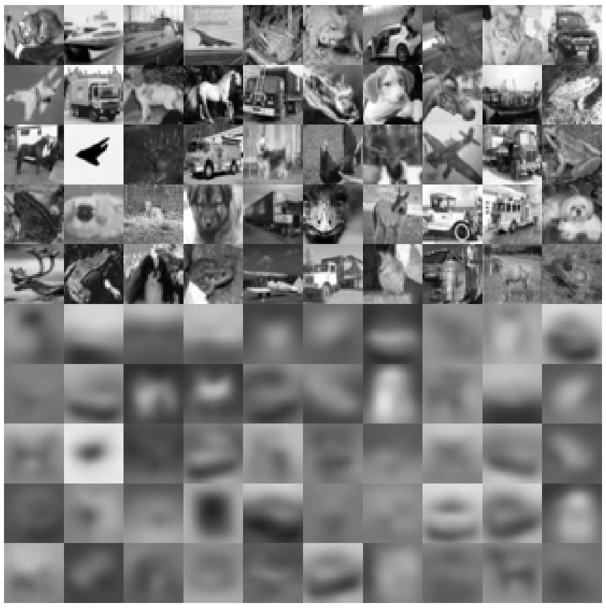

Figure 11. Reconstructed Images of CIFAR10

Above are the Reconstruction images obtained of CIFAR10. For CIFAR10, we tested the same model mentioned in paper, with two dimensional (black& white) images. When we tested Food 101 dataset with more deep capsule network, it failed with memory error. It is due to large number of categories, which made the matrix multiplication for backward pass more complex. Later, we tested for very simple network with less number of examples, but performance with it was extremely poor. The model underfit to the data with or without the application of dropout. This can be attributed to the low and varying number of examples per category.

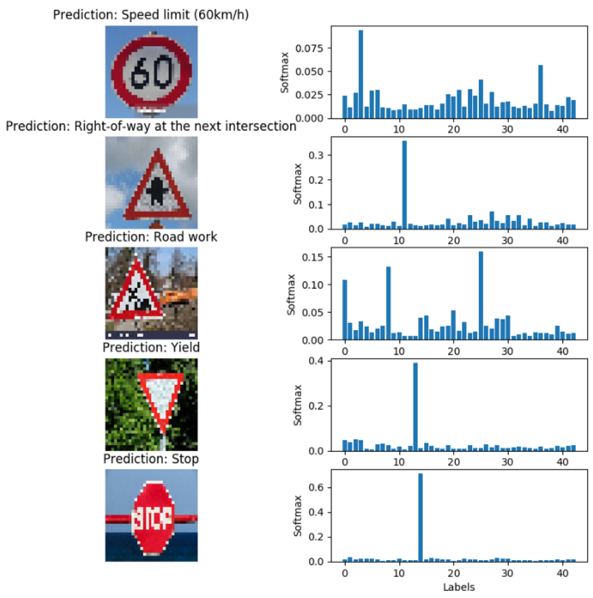

Figure 12. Sample Probability output for Traffic Signal Dataset

5. Discussion

Our study of capsule networks brought to light the some points of consideration. In order for higher level capsules to capture part-whole relationships from lower level capsules, [1] suggests increasing capsule dimensions as one move up the layers of the network. For our experiments we increased capsule dimensions from 16D to 32D (by a factor of 2, similar to [1]). This increase in dimensionality caused a choking of computational resources when performing classification on Food101 dataset. [1] does not discuss methods of estimating capsule dimensions and the rate at which their dimensionality should grow. Also, training time is contingent on the number of iterations used for the routing-by-agreement process. A large number of iterations would lead to slower training times. This was a great challenge given that our model with just 1 iteration of dynamic routing took close to 20 hours to converge. Capsule Networks can be really helpful in terms of image segmentation, as they are able to capture the exact spatial features, as they capture the exact spatial position of each element inside image, thus can be used in to various applications including Object Recognition in Image Segmentation, and Automatic Driving Cars.

6. Conclusion

Capsule networks performed very well for the Traffic Signal dataset with 43 categories. More variations of the architecture need to be explored in order to scale to datasets with large number of categories like ImageNet. What is interesting about capsule networks is, that it has directed neurons for each of the features to be considered. we were able to get better accuracy with drop outs, which says that ensembles of the capsule networks was giving the best result. Ensembles work because there might be a lot of values on which global optima is based on, but those doesn't play role in loss minimal value. In future research, we can look in to analysis of dropouts in capsule networks, so as to understand the loss graph minima's, as the structure of capsule network is comparatively small and simple, it will help us understand properly about loss convergance.

References

[1] Dynamic Routing Between Capsules, Sara Sabour, Nicholas Frost, Geoffery E. Hinton, arXiv:1710.09829

[2] Understanding Hintons Capsule Networks. Part I: Intuition: "https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b"