IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-2, Feb 2019 E-ISSN: 2347-2693

Capsule-Networks: Towards Object-Detection Capsule Object-Detector (COD)

Amit Baghel[1*], Swati Dwivedi[2]

[1]Department of Computer Science & Engg., School of Studies in Engg. & Tech.,GGV,Bilaspur, Chhattisgarh, India
[2] Department of Computer Science, Kalinga University,Raipur, Chhattisgarh,India

[*]*Corresponding Author: amit_kumar_baghel@rediffmail.com*

DOI: https://doi.org/10.26438/ijcse/v7i2.230236 | Available online at: www.ijcseonline.org

Accepted: 12/Feb/2019, Published: 28/Feb/2019

Abstract—Although Convolutional Neural Networks performed better in object detection, CNNs does not care about spatial relationships existing in an image. In this paper, we try describe "capsule network based object detection" model COD based on the VGG16 model (as a base network), which presents a substantial result in many sections of object detection over Convolution Neural Network based model by achieving the problem of spatial relationships. We used matrix capsules and dynamic EM routing to classify object from different viewpoints. The whole model is grounded on "dynamic routing between capsules", which is suggested by Geoffrey E Hinton. Both proposed theories use capsules that maps feature properties of an object as information for detecting that object which is extracted by capsules and Dynamic routing groups the capsules of lower level into parent level capsules by an iterative dynamic routing process. We train and test our model on Pascal VOC 2007 and dataset. We implement this in python using Keras (Tensorflow as backend) and train our model in Google cloud compute engine. COD achieves an accuracy of 67.3 mAP on Pascal VOC-2007 dataset and performing a comparable performance with Fast R-CNN.

***Keywords*—**Object Detection, CNN, Capsule Networks, VGG16

I. INTRODUCTION

To survive in the environment, the human glance at an image and instantly understand about environment. The human knows about object what they see and how they can interact with that object. The human vision system is very fast and accurate. Detecting and classifying objects are one of the most important uses of the human vision system. Human can easily detect more than 30,000 object categories in milliseconds.

Current object detection systems are based on CNN, which is not true visualization of the human vision system. CNN based object detectors (e.g., [1, 4, 5, 6, 7]) are accurate and fast, but they have many drawbacks:

1.CNNs are translation Invariant means CNN model needs images of every pose and translation of an object for training to perfectly detect that object because they are unable to identify the position of an object and they are also unable to identify the position of object one relative to another.

2. CNNs needs to learn object with different filters and different viewpoints so they need lots of data for generalization.

3.In CNN max pooling losses important information about pose and location of an object.

Capsule networks[9] are a true representation of the human vision system. Like human vision system, in capsule networks we use inverse graphics means network need not to learn for every pose and translation of an object.

In this paper, we try to build a capsule networks based object detection model COD, which removes drawbacks of CNN based object detector. In this approach, there is no loss of important features like pose and location of an object in the image. Our network architecture looks like complex, but it is easy to implement. We try different types of methodology to increase the accuracy of the model.

Recently, object detection system achieves a state-of-art performance in object detection. This paper presents the first object detector model that does not use any max-pooling layer. Instead of max-pooling we use a dynamic routing algorithm to train our model. Our model looks like Fast R-CNN, but it is faster. In future we try to build our model like YOLO [6] and SSD [5], which are fastest object detector mode till now. The fig 1 shows the operating model of COD.

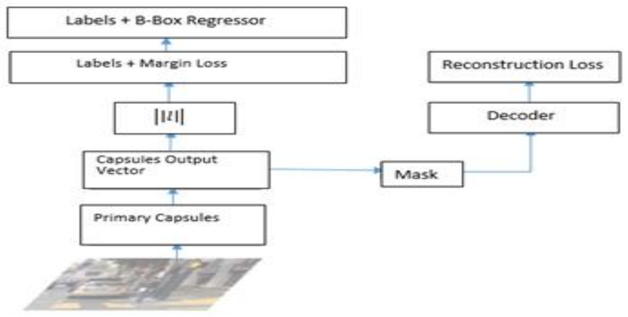


Fig 1: operating model of COD

We sum up our contributions as follows:
1. We introduce a basic network architecture COD using capsule networks [9] that fixes some disadvantages over CNN. It is more accurate than Fast R-CNN. But in future we try to make it as accurate as YOLO and SSD.

2. In order to achieve state of art accuracy, we use different types of methodology like stacking more primary capsules, scaling of reconstruction loss and using convolutional layers before primary capsules.

3. We perform experiments of Pascal VOC 2007 and CIFR 100 dataset and compare results with several object-detector.

The rest of the paper is organized as follows: Section II deals with the Related Work. Section III describes the model architecture, methodology and other implementation details of the model. Section IV deals with the results after implementation, performance metrics followed by conclusion and future scope in Section V.

II. RELATED WORK

All object detection models (e.g., [1, 4, 5, 6, 7, 8]) till now are based on the Convolution Neural Network (CNN). We can classify all these models into three categories (1) based on Sliding Window Technique (e.g., [8]), (2) based on the Region Proposal Network (e.g., [1, 4, 7]), (3) based on Single Shot Detection (e.g., [5, 6]).

All these models have comparable and continued increasing performance and accuracy. The models of first class were the beginning of object detection. R-CNN approach improves the quality and speed of classification of images, but it takes a number of convolution neural layers for extracting features of images, which is expensive and time-consuming.

The second class models improve the quality of proposals. Fast R-CNN [7] model uses selective search region proposal [15] but Faster R-CNN and R-FCN [4] use different types of network layer called RPN (Region Proposal Network) [1] instead of selective search [15] for proposal generation. These models are not complex because it uses a single layer of convolution for feature extraction.

Another set of models is based on single shot detection (SSD, YOLO), these models skip the proposal step used in earlier models. YOLO architecture is based on fully convolutional neural network (F-CNN [16]) and passes the image once through one F-CNN [16] layer and predict the output. It reframes object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

SSD (Single Shot Multi-Box Detector) [5] model also falls in this category, it also uses the default bounding boxes instead of proposal step, but it is more flexible and more accurate for tiny and crowded objects as compared to YOLO. It uses VGG-16 [17] network architecture as a base network.

Our model uses a different type of networks called Capsule Networks (an overview of capsule network is given below). As like SSD we used VGG-16 [17] as a base network for feature extraction and instead of max-pooling in convolution layer we used caps-pooling layer. Our model does not yield that much accuracy like Faster R-CNN, YOLO and SSD but it is a beginning of capsule networks in the field of object detection.

It delivers tons of advantages over any convolution neural network based object detection models (given below). We test our model on Pascal VOC-07 [2] and CIFR 100 dataset. Our model uses VGG-16 [17] network architecture for feature extraction in the form of an array, then reshape an array to set of vectors for each location and one different type of function Squash function is used to ensure that length of vectors should range between 0 to 1. Capsule networks preserve detailed information about the pose and location of data so our model needs less training data over other models.

Table 1: comparison of all object detection methods over our model on Pascal VOC-07 dataset.

Object Detector Model	**Accuracy in mAP (%)**
RCNN	62
Fast R-CNN	66
Faster R-CNN (RPN + VGG)	70.4
YOLO V1	78.6
SSD 300	74.3
SSD 500	76.8
COD (VGG16 + Capsnet)	67.3

III. Proposed Method

1. Model Architecture

Fig 2 describes the architecture of our model. For training, it takes the entire image as input with bounding boxes. The model first takes whole image and reshape it into 300x300 shape and process further the reshaped image with convolutional layers of VGG16 model and produce feature maps. Next, we reshape the output to 8D vectors. Then, we squash the output vector using a squash function based on the equation given in Hinton's paper. This gives the outputs of primary capsules. Next, we create a layer called caps pooling layer which implements the dynamic routing between capsules and preserve translational invariance of the network. This layer allows to train our model. Caps pooling layer extract object features from feature maps and gives feature vectors as output. Then we fine tune VGG 16 model by adding two fully connected capsule layers with Relu activation function. Feature vectors fed into two branches: one gives the sigmoid probability of object classes and another gives four points which define the position of bounding boxes. Our network architecture looks like Fast R-CNN, but it gives more accuracy on Pascal VOC 2007 dataset.

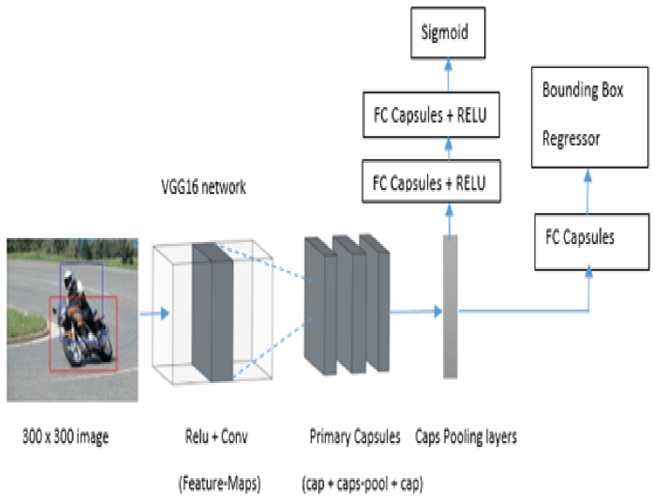


Figure 2: COD model architecture

1.1 Caps-Pooling

We used Caps-Pooling layers in our model like in CNN we use max-pooling layers. This caps-pooling layer is a naive inspiration taken from Hinton's paper on capsule networks. In our model we use some convolutional layer with caps-pooling layer instead of max-pooling layer. Our assumption is to make a layer which can implement dynamic routing between capsules. In our model we use an optimized dynamic routing algorithm [11] which gives us better result during training time. Caps-pooling layer groups capsules to form a parent capsules and calculates the output of capsules. Caps-pooling layer preserve translational invariance of the network. It works like a max-pooling layer, but it chooses which feature column pooled within pooling region. So we can say that each region can be a separate capsule problem. In caps-pooling we perform an optimized iterative dynamic routing algorithm. This algorithm allows to train our model.

1.1.1 How Many Routing Iteration uses

Hinton's paper examined a range of values for both MNIST and CIFAR data sets. From Hinton's paper, we can remark that more routing iterations increases the network capacity and over-fit the training data set. Hence, in our model we use three routing iterations. The below image depicts the experimental verification of average change in routing logits of our theoretical account at each routing iteration due to convergence of number of routing iterations.

1.2 Methodology

We start with the VGG 16 model as a baseline model on images with 3 color channels. The baseline model is for low dimensionality dataset like MNIST. To apply for the more complex features of Pascal VOC 2017 or CIFR 100 dataset, we stack more capsule layers for more complicated and detailed features of objects improves the representational ability of the model. As the dimensionality of data is increases, we try to increase the number of primary capsule layers. It yields a better accuracy of learning richer features till 1024 layers, but after 1024 primary capsules it does not bear on the accuracy. Hence, in our model we take 1024 capsules. We choose a VGG 16 model as a base network for making a more complex image encoding before feeding it into primary capsule layers. Producing a more complex encoding of images, improves the accuracy of model [12]. Hinton's capsule network has property to explain every object in images. So, we add a none of the above category. It also improves the accuracy, but not so much.

1.3 Loss Function

Loss function used in our model looks like very complicated, but it is not. For understanding about how our loss function works see Hinton's capsule networks paper. In our model there is two types of loss function, first one is margin loss and the second one is reconstruction loss.

We need to detect multiple target of same class that's why during Training of the network, for each training lesson, one margin loss is computed for each object class according to Hinton's margin loss formula (given below) and then all margin loss for each object class added together for calculating final margin loss.

In this model, there is an additional reconstruction loss as a regularization method to avoid overfitting. During training of capsule network, instead of sending all capsule networks' outputs, we need to send only output vector that belongs to target object class, so we must mask-out all other output vector.

1.3.1 Margin Loss

Our model uses a special margin loss to detect objects in crowded images in which two or more object of the same type. For each object capsule their is margin loss, which is find by Hinton's proposed formula (given below) in his paper.

$$L_C = T_C \max(0, m^{\pm ||v_c||})^2 + \lambda(1 - T_C)\max(0, ||v_C|| - m^-)^2$$

Where :

L_C = loss term for one capsule

Tc = 1 if an object of class c is present

$T_C \max(0, m^{\pm ||v_c||})^2$ is calculated for correct object capsules and we use L2 normalization.

λ = 0.6 (used for numerical stability

$+ \lambda(1 - T_C)\max(0, ||v_C|| - m^-)^2$ Is calculated for incorrect object capsule.

Finally, the total margin loss is simply the sum of the losses of all object capsules. For getting more details about above given formula follow Hinton's paper. [9]

1.3.2 Reconstruction Loss

In network architecture on top of capsule network there is a decoder. It is a band of three fully connected capsule layers which learns to reconstruct the image based on capsule network's output. This decoder network preserves the required information to reconstruct the objects. It avoids the risk of overfitting the training dataset and also helps to generalize new objects.

Reconstruction loss is just a squared difference between the input image and reconstructed image. We minimize reconstruction loss as much as possible up to 0.007. Final loss is the sum of margin loss and reconstruction loss.

1.4 Implementation Details

We train and test model using dynamic routing algorithm. We re-scale the images into 300 × 300 shape. Then, we apply convolutional layer of VGG 16 model which is pre-trained on ImageNet small dataset. In our model VGG16 network architecture is used for classification of objects. VGG 16 model helped to fuel our model using transfer learning and fine tuning where pre-trained model is used with some modification.

VGG model can be easily loaded with Keras deep learning library. We also look at the distribution of classes during training. There are lots of classes in the dataset. Fig 3 shows the distribution of some classes and fig 4 shows some preprocessed images of the dataset:

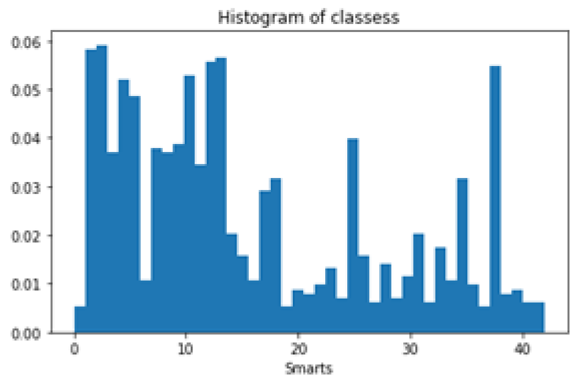


Figure 3: Histogram distribution of some class

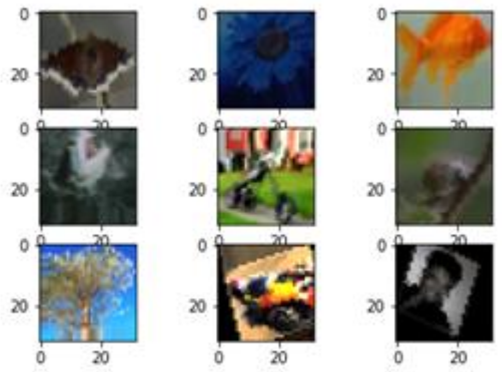


Figure 4: pre-processed images of PASCAL VOC7

There is lots of pre-trained model available in Keras deep learning library but VGG 16 is best performing model for classification of objects. With VGG 16 model we achieve an accuracy of 67.3 mAP. We trained three VGG model with capsule network architecture with varying the dimension of input images and ensemble with our capsule network architecture with bounding boxes during training. Our model is robust and scalable model which can classify and detect object in unseen images also. In our model there is no need to train the model with cross validation and data augmentation. In our model we use bounding box-regression like Fast R-CNN [7]. On higher learning rate, the model is failed to converge but when we use a small learning rate then, model is converged, but at a very slow rate. We also tried to train the model using normalized bounding boxes but it not makes so much difference.

Most of the time, our model is failed to detects object in images which has object of same type and it also not yields accuracy like Faster R-CNN, YOLO and SSD which are best suitable for Real-Time object detection systems. During testing, we see that bounding box regression is break down on some unseen images which has lots of object of same type.

IV. Results and Discussion

1 Experiments

Our all experiments are based on VGG16 [17] model. We use VGG16 model as a base network, which is pre-trained on Imagenet small dataset [3]. Like SSD model we remove all dropout layers and fc8 layer and fine tune our model using ADAM optimizer with 0.0007 weight decay and 100 batch size for 10 epochs. First, we test our model on Pascal VOC 2007 dataset and compare its accuracy with all other object detection systems. Finally, in Table 1 we can see that our model yields more accuracy then Fast R-CNN and in some cases it is also better than other detection systems but till now our model is not suitable for Real-Time object detection like YOLO [6] and SSD [5].

1.1 Capsules on Pascal VOC 2007

We perform experiments on PASCAL VOC dataset that has 20 object classes. We use VGG 16 as the base network, which is pre-trained on ImageNet small dataset and later train our model on PASCAL VOC 2007 trainval dataset and evaluate test result on VOC 2007 test set. We measure our model's detection accuracy in mean Average Precision (mAP). Process of detecting any object from an image is first extract features from input images then, classifiers are used to identify objects in feature space. Then, localizers are used to set bounding boxes over identified objects. We use VGG 16 model as base network, so we compare our model from object detection systems that use this model (Fast).
We compare our model with Fast R-CNN and Faster R-CNN since Fast R-CNN is one of that models that yield highest accuracy on Pascal VOC 2007 dataset. All these methods use same training and validation data and pre-trained model of VGG 16. To check the performance of the model, we used detection analysis tool [19] like SSD.

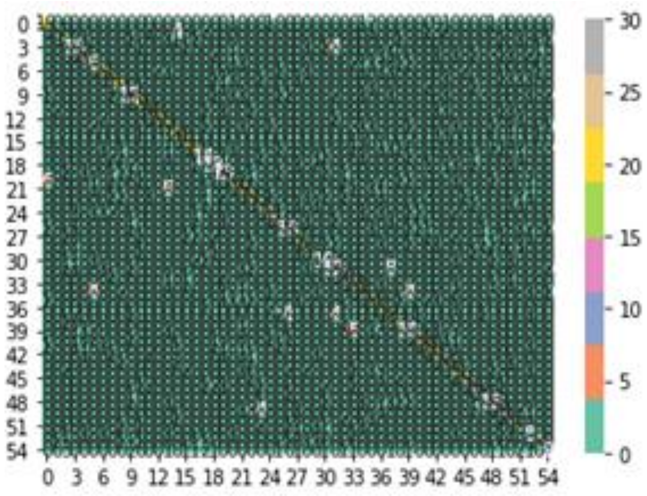
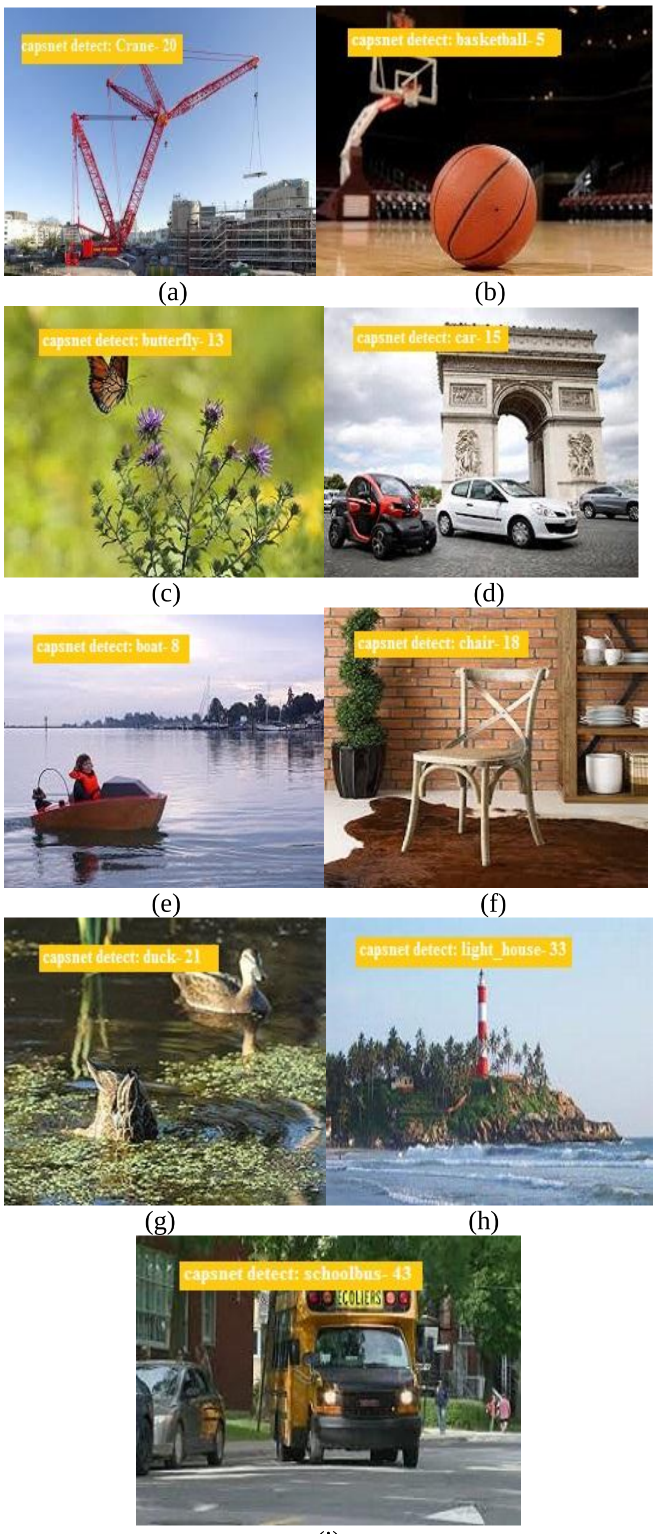


Figure 5: Confusion matrix of 55 classes, which describe performance or the model.


capsnet detect: Crane- 20
capsnet detect: basketball- 5
(a)
(b)
capsnet detect: butterfly- 13
capsnet detect: car- 15
(c)
(d)
capsnet detect: boat- 8
capsnet detect: chair- 18
(e)
(f)
capsnet detect: duck- 21
capsnet detect: light_house- 33
(g)
(h)
capsnet detect: schoolbus- 43
(i)
Figure 6: COD run on web images. Although it is accurate but this model does not able to detect more than one object in images(c, d, i)

2 Main Results

This paper contains following main results:

1. First capsule networks based model which achieves a better performance then Fast R-CNN on Pascal VOC 2017 and CIFR 100 dataset.
2. Achieves 67. 3% mAP on Pascal VOC 2017 dataset.
3. Fine tuning VGG 16 model with capsule networks.

3 Overview of Capsule Networks [9]

Computer graphics deals any object type with its various instantiation parameters. To obtain the image we apply some rendering function on these instantiation parameters. Fig 5 shows computer graphics rendering to obtain an image. Easily we can say that computer graphics takes the internal representation of objects and produces an image but our brain does its opposite, Hinton follow this idea and call this "Inverse Graphics". In inverse graphics, we start with an image and try to find out the type of objects it contains and what their instantiation parameters are. Fig 6 shows the representation of Hinton's proposed inverse graphic.

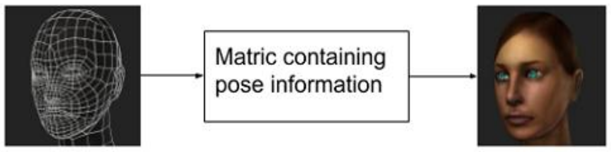


Figure 7: Computer Graphics rendering (simplified)

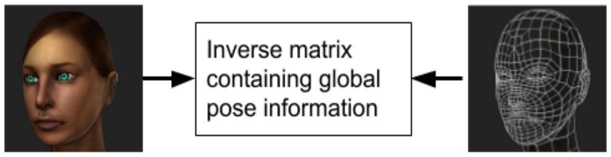


Figure 8: Inverse Graphics (Hinton's Proposed)

Hinton's capsule networks are based on this idea. Capsule network is a neural network that tries to perform inverse graphics. Capsule network is composed of many capsules and capsules is any function that tries to predict pose and instantiation parameters of an object at a given location. Three key features that distinguish capsule networks from CNN are Inverse graphics, Squashing function and Dynamic Routing. Capsule's output is generated in vector form and length of vector represent estimated probability of object that capsules are looks for in the image but no vector length should greater than one, to check this we apply squashing function.

Capsule Network replace max-pooling of CNN's with routing-by-agreement and scalar-output with vector-output. Routing-by-agreement preserves detailed information about object location and its pose but in CNN pooling layer tend to lose information, such as the precise location and pose of objects. Object-detection requires precise location and pose of objects. In any capsule architecture first, we apply a couple of convolutional layers and reshape obtained outputs to get vectors and squash them. Every capsule in the first layer try to predict the output of every capsule in next layer but capsule's outputs are only routed to appropriate capsule in next layer, called Routing-By-Agreement. Like this we can determine the pose and location of the object.

V. CONCLUSION AND FUTURE SCOPE

CNN models are unable to identify position of one object relative to another and are translation invariants so it requires lots of data for generalization. In this paper, we have proposed a model which can easily identify object that hold spatial relationship [20] between features and label it correctly. Also, our model needs less data as compared to CNN based model. We conclude by noting that our proposed model is not scalable and has not achieved a state of the art performance like Faster R-CNN [1], YOLO [6] and SSD [5] and as of now it is not suitable for Real-Time Object-Detection but our model achieves accuracy competitive with Fast R-CNN on Pascal VOC 2007 dataset.

As a part of future work, we would be going to do changes in our model to achieve a better performance in Real-Time Object-Detection. We expect that our model will easily give the benefits of the capsule networks [9, 10] in the field of object-detection

REFERENCES

[1] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[2] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," International Journal of Computer Vision, 2014.

[3] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[4] J. Dai, Y. Li, K. He and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," 2016.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single Shot MultiBox Detector" IEEE European Conference on Computer Vision, 2015.

[6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015.

[7] Ross Girshick, "Fast RCNN" IEEE International Conference on Computer Vision, 2015.

[8] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2014.

[9] S. Sabour, N. Frosst and G. E. Hinton, "Dynamic Routing Between Capsules," 2017.

[10] G. E. Hinton, A. Krizhevsky and S. D. Wang, "Transforming Auto-encoders," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2011.

[11] D. Wang and Q. Liu, "An Optimization View on Dynamic Routing between Capsules" Workshop track in International Conference on Learning Representations 2018.

[12] E. Xi, S. Bing and Y. Jin, "Capsule Network Performance on Complex Data," 2017.
[13] A. Jaiswal, W. AbdAlmageed and P. Natarajan, "CapsuleGAN: Generative Adversarial Capsule Network," 2018.
[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" 2017.
[15] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers and A. W. M. Smeulders, "Selective Search for Object Recognition" International Journal of Computer Vision 2013.
[16] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015.
[17] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2015.
[18] L. Zhu and H. Yuan, "Spatial Relationship for Object Recognition" International Conference on Learning Representations 2015.
[19] Hoiem, D., Chodpathumwan, Y., Dai, “Diagnosing error in object detectors”, IEEE European Conference on Computer Vision, 2012.