Image Classification using Capsule Network for Scene Graph Generation

Vijay Tadikamalla
Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Telangana, India - 502285
cs17btech11040@iith.ac.in

Vineeth N Balasubramanian
Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Telangana, India - 502285
vineethnb@iith.ac.in

Abstract

Convolutional Neural Networks (CNNs) have played an essential role in the field of deep learning. Many variants' of CNNs have proven to be very successful in many Computer Vision tasks across different domains. However, there are two main drawbacks of CNN's: their failure to take into account of important spatial hierarchies between features, and their lack of rotational invariance [2]. To address these concerns, Hinton et al. proposed a novel type of neural network using the concept of capsules. With the use of dynamic routing concept and reconstruction regularization, the capsule network model would be both rotations invariant and spatially aware [1]. These properties of Capsule Networks can prove out to be very useful for tasks like Scene graph generation, where understanding a visual scene goes beyond recognizing individual objects in isolation. However, there is a long way before we can use Capsule Network models for scene graph generation as training a Capsule Network model requires significant computational resources and currently they were just tested on simple and low dimensional datasets like MNIST. So, we attempt to perform image classification on datasets with higher dimensionality and complexity like CIFAR-10 and CIFAR-100 using Capsule network model.

1 Introduction

The aim of scene graph generator is to infer a visually grounded graph from a given image. This graph comprises of localized entity categories, along with predicate edges denoting their pairwise relationships [8]. This is often formulated as the detection of $\langle subject, predicate, object \rangle$ triplets within an image, e.g. $\langle man, riding, horse \rangle$ in Figure 1. Currently, most of the state-of-the-art methods achieve this goal by using Convolutional Neural Networks (CNNs).

We find that models using CNNs tend have few limitations. The first is the pooling operation used in CNNs. For example, if the pools do not overlap, pooling loses valuable information about where things are. We need this information to detect precise relationships between the parts of an object. In short, they fail to take into account of important spatial hierarchies between features. So, as long as certain key features of an object are present in the test data, CNNs classify the test data as the object, disregarding features' relative spatial orientation to each other. This causes false positives. The second limitation is their lack of rotational invariance. This limitation of CNN's would cause the network to incorrectly assign the object another label, causing false negatives. One way to eliminate these problems is with excessive training for all of the possible angles, but it

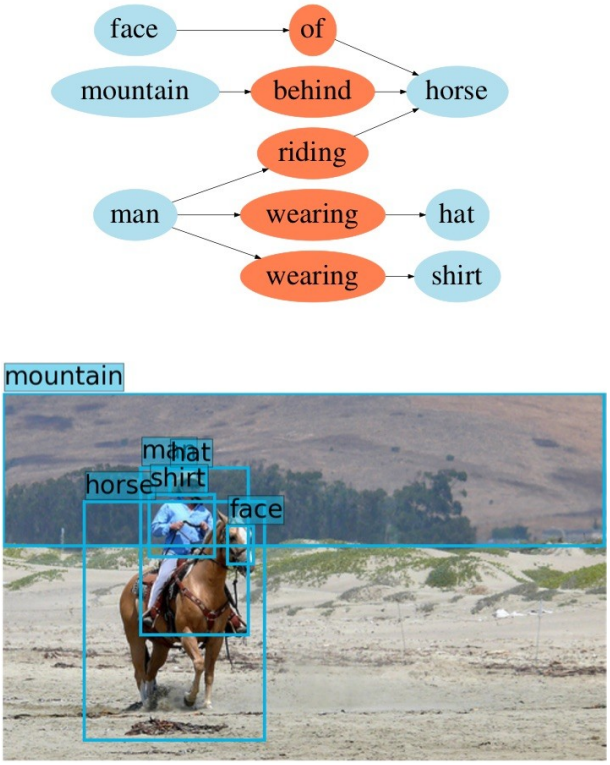

Figure 1: Scene graph of the given image [10]

usually takes a lot more time and computational resources. Moreover, convolutional neural networks can be susceptible to white box adversarial attacks [4] and the so-called "fast gradient sign method" [5].

These drawbacks of CNNs, limit their capabilities. So, to provide better performance and results on a tasks like scene graph generation, a model must be spatially aware and achieve "equivariance" [2]. To overcome these limitations, Hinton et al. propose a novel type of neural network using the concept of Capsules in a his recent paper [1].

At present, Capsule Networks have shown their potential by achieving a state-of-the-art result [1] of 0.25% test error on MNIST without data augmentation, such as rotation and scaling, better than the previous baseline of 0.39%. Capsule Network is a promising concept in deep learning, yet its true potential is not fully realized thus far, providing sub-par performance on several key benchmark datasets with complex data.

So, in this paper, we aim to extend and improve perform of Capsule network model on datasets with higher dimensionality and complexity.

2 Related Work

2.1 Overview of Capsule Networks

Capsule networks represent a recent breakthrough in neural network architectures. They introduce an alternative to translational invariance other than pooling through the use of capsules.
There are some fundamental differences between Capsule networks and CNNs like the idea of translated replicas of learned feature detectors forms the basis of CNNs. In other words, information about properly trained features gathered in one position can be spread out to other positions. In contrast, CapsNet replaces the scalar-output feature detectors from CNNs with vector-outputs, it also replaces the max-pooling subsampling technique with routing-by-agreement, so it enables the duplication of learned knowledge across space. To summarize, the Capsule Networks give us the opportunity to take full advantage of intrinsic spatial relationships and model the ability to understand the changes in the image, and thus to better generalize what is perceived.

2.2 Vector-to-Vector Nonlinearity

The length of the output vector of a capsule represents the probability that the entity represented by the capsule is present in the current input. So, a non-linear **"squashing"** function is used to maintain the relationship between length of vector and probability [1].

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

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

2.3 Dynamic Routing

Dynamic routing can be viewed as a parallel attention mechanism that allows each capsule at one level to attend to some active capsules at the level below and to ignore others. Capsules output a vector, so for each potential parent, the capsule network can increase or decrease the connection strength. This routing by agreement is much more effective at adding invariance than the primitive routing introduced by max-pooling [1].

2.4 Loss function

The marginal loss function enhances the class probability of the true class, while suppressing the class probabilities of the other classes [1].

$$\begin{aligned} L_k = {} & T_k max(0, m^+ - ||v_k||)^2 \\ & + \lambda(1 - T_k) max(0, ||v_k|| - m^-)^2 \end{aligned} \tag{2}$$

where $T_k = 1$ iff a digit of class k is present and $m^+ = 0.9$ and $m^- = 0.1$.

2.5 Reconstruction Regularization

Traditional CNN's prevent overfitting by using dropout, Capsule networks are regularized with a reconstruction autoencoder.

2.6 Other Works

Recently, a new Capsule network architecture "DeepCaps" got published [6]. It uses a novel 3D convolution based dynamic routing algorithm. It was able surpass the previous state-of-the-art results in the capsule network domain on CIFAR-10, SVHN and Fashion MNIST, while achieving a 68% reduction in the number of parameters.

3 Experimental details

The Hinton's Capsule Network model [1] comprised of only one convolution layer and one fully-connected capsule layer. It worked for simple datasets like MNIST but the its performance on datasets with more complex objects such as CIFAR-10 is not on par with the CNNs, due to the nature of complex shapes in CIFAR-10 in comparison to MNIST.

3.1 Dataset

We perform experiments on three well-known standard datasets: MNIST, CIFAR-10, and CIFAR-100.

3.2 Model Architecture

We modify Hinton's MNIST model [1] to handle complex datasets like CIFAR-10, Cifar-100 etc .

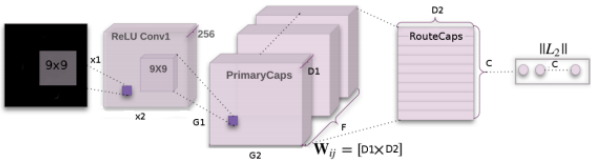

Figure 2: Modified CapsNet architecture [1]

This first convolutional layer produces 256 feature maps, using a 9 x 9 kernel and valid padding. For the convolution within the primary capsules we also kept a kernel size of 9x9, x_1 and x_2 entirely depend on the size of the input image. G_1 x G_2 are the dimensions of each capsule, and these values are computed automatically based on x_1 and x_2. Other parameters we tuned for different datasets including D_1 and D_2 (the dimensions of the output vectors in primary and routing capsules), F (the number of channels in the primary capsule layer), and C (the number of classes).

3.3 Approach and Methodology

So, to handle more complex shapes, we tried the following approaches on the above model.

- Increasing the no. of primary capsules
- Modifying the non-linearity
- Stacking more fully-connected capsule layers
- Decreasing the no. of routing iterations and stacking more fully-connected capsule layers.
- Dynamic Routing using 3D convolution [6]
- Class Independent Decoder Network [6]

After training the CapsNet model with above modifications, we found that stacking more fully connected capsule layers or increasing the no. of primary capsules leads to huge spike in the training time as dynamic routing used in capsule networks is an extremely computationally expensive procedure. We also noticed the dampening of the gradient flow. This was caused because as the no. of capsules increase, the coupling coefficients tend to decrease to very small values.
We also tried new activation functions like $(1 - \frac{1}{e^{|x|}})\frac{x}{|x|}$ in place of the original squash function but the results very not upto the mark.

Finally, we were able to able better results, when we decreased the no. of routing iterations and stacked more fully-connected capsule layers. This happened because as we decrease the no. of routing iterations, the computational complexity introduced by multiple layers needing dynamic routing decrease. This prevents diminishing gradients and finally improves the model performance.

However, the best results were obtained by the DeepCaps model[6].

3.4 Computing resources

Most of the work was done on Google Colab which provides Nvidia Tesla K80 GPU, 12.72 GB of RAM, and Intel(R) Xeon(R) CPU 2.00GHz

3.5 Software used

All the models trained and used during for the experiments were written in Python. The main learning frameworks and libraries included Keras, Tensorflow, Numpy and Scipy, etc.

4 Results and Conclusions

The performance of the models was evaluated on three datasets: MNIST, CIFAR-10, and CIFAR-100.

Even though our results are slightly below or on-par with the state-of-the-art results on datasets like CIFAR-10 and MNIST. We highlight that we were able to achieve a near state-of-the-art performance across the datasets like CIFAR-10 and MNIST without performing data augmentation and ensembling.

Model	MNIST	CIFAR10	CIFAR100
DenseNet	98.6%	98.41%	82.62%
ResNet	99.59%	93.57%	80%
Our results	99.3%	86.40%	26% *
Hinton's [1]	99.75%	89.40%	18% *
DeepCaps [6]	99.72%	91.01%	35.70% *

Table 1: Results of various models on various datasets. * tells that the results were reported after 25 epochs.

Also, please note that some of the above results were reported before they could converge. Due to limits in computational resources, the average training time was very large. So, we had to report their accuracy before convergence.

Figure 3: Sample MNIST test reconstructions with 3 routing iterations. The above five and the below five digit layers show the input and the reconstructed image respectively.

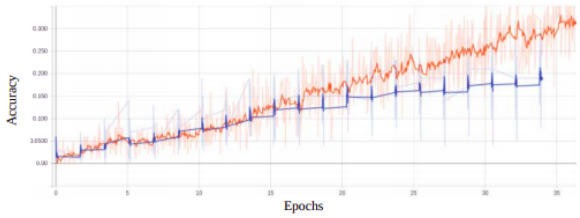

Figure 4: Training accuracy on CIFAR-100 dataset

Finally, we would like to conclude that Capsule networks are in their initial phase of development (same as CNNs 10-15 years ago). They are quiet promising concept of deep learning and their its true potential is not fully realized.

5 Future Work

- We will try improve the image classification accuracy on high dimensional datasets like CIFAR-100, ImageNet etc.
- We will also try to use Capsule network on more complicated computer vision tasks like object detection, image segmentation and finally scene graph generation.

6 References

1. Sabour, S., Frosst, N., Hinton, G.E, "Dynamic Routing Between Capsules", 2017
2. G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders,", 2011
3. Adam R. Kosiorek, Sara Sabour, Yee Whye Teh, Geoffrey E. Hinton, "Stacked Capsule Autoencoders", 2019
4. Ilyas, A., Engstrom, L., Athalye, A., Lin, "Query-Efficient Black-box Adversarial", 2018
5. Goodfellow, I.J., Shlens, J., Szegedy, "Explaining and Harnessing Adversarial", 2014
6. Jathushan Rajasegaran, Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Suranga Seneviratne, Ranga Rodrigo, "DeepCaps: Going Deeper with Capsule Networks", 2019
7. Danfei Xu, Yuke Zhu, Christopher B. Choy, Li Fei-Fei, "Scene Graph Generation by Iterative Message Passing", 2017
8. Ji Zhang, Kevin J. Shih, Ahmed Elgammal, Andrew Tao, Bryan Catanzaro, "Graphical Contrastive Losses for Scene Graph Parsing", 2019
9. Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, Devi ParikhGraph, "R-CNN for Scene Graph Generation", 2018
10. R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei, "Visual genome: Connecting language and vision using crowdsourced dense image annotations.", 2016.