APPLICATION OF CAPSULE NETWORKS FOR IMAGE CLASSIFICATION ON COMPLEX DATASETS

BY

DEBORSHI GOSWAMI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Industrial Engineering
with a concentration in Advanced Analytics
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Assistant Professor Ruoyu Sun

ABSTRACT

Capsule Network, introduced in 2017 by Sabour, Hinton, and Frost [1], has sparked great interest in the computer vision and deep learning community and offers a paradigm shift in neural computation. In CapsNet, Sabour et. al. replace classical notions of scalar neural computation with a vectorised approach. This allows CapsNet to describe input images not only by the presence of constituent features but also by the pose of detected features, thus imparting view-point and pose invariance. Hinton's group and the research community at large have applied CapsNets to a number of specific problems and achieved state-of-the-art performance. In contrast, this thesis studies CapsNet by applying it to complex real world datasets like CIFAR10 and CIFAR100 where the CapsNet's performance is still unproven. We investigate the operational characteristics of CapsNet for the CIFAR10 problem and identify several practical limitations of Capsules that inhibit their performance in an industrial setting. The contribution of this research is the introduction of residual blocks of primary capsule layers. We developed a novel architecture for CIFAR10 classification, called ResCapsNet, and find that the model increases validation accuracy to 78.54% from 71.04% achieved by the baseline CapsNet, at the marginal cost of increasing the number of parameters from 22 million to 25 million. In addition, to extend the generalization of capsules into deeper networks, we discuss the application of Capsules as hidden layers in CIFAR100 classification and show that Capsules are largely ineffective in a latent unsupervised setting. For active supervision of hidden capsules, we propose methods to train hidden capsules as super-class detectors prior to final classification.

ACKNOWLEDGEMENTS

I have to start with an absolute and unfiltered sense of gratitude toward my adviser, Assistant Professor Ruoyu Sun, to whom I am deeply grateful for giving me the opportunity to study and conduct research on something as interesting as Capsule Networks, in-spite of knowing how inexperienced I was when I started. Have gave me the chance, and when I faltered he gave me another one. His kindness and encouragement speaks volumes of how he views his students and empathizes with them. His constant search of knowledge and excellence in his academic career is one of the reasons that I aspire to come back to academia one day and become a researcher.

A tremendous thanks to the National Center for Super-Computing Applications at UIUC, and Professor Volodymyr Kindratenko for allowing me access to the life-giving resource that is Nano Cluster. Many a student and researcher would be left stranded if not for the Tesla V100s that keep our models alive and I am certainly one of those students.

A big thanks to my department, for encouraging such a diverse set of research, academic, and professional interests among its Graduate students. Department of ISE's affinity to Applied Mathematics, Machine Learning, and Optimization had enthralled in this beautiful world of Data Science from day one of class. Thank you, not just for the opportunity to conduct this thesis, but for the last two years as a whole. Thank you for having me here at UIUC and in your department, for teaching me and for helping me navigate the strange world of Master's Degrees.

My parents and my girlfriend, Sindhura, who inspite of being five thousand miles away, have still managed to see me through thick and thin, have given me courage, have loved me through the toughest moments of being away from home. Thank you.

Thank you Sharada Srinivasan, Gandhar Kunte, and Vikram Kamath for being my closest circle of support in the United States. Thank you Sathwik Tejaswi, for the rambling sessions of brainstorming potential A.I. 'breakthroughs' and me being horribly wrong on every one of those ideas. Thank you Naman Shukla for keeping it real, always, and for showing me what excellence should look like. Thank you Dhayabharan Ponsingh for being my confidante, Nishanth Velugula and Kavjit Durairaj for just being so darn cool, Karthik JVN for your pearls of wisdom, and Siddharth Chakravarthy for your casual professionalism. Being around you all has made my experience at UIUC truly memorable.

Dedicated to my parents whose unconditional love, support, and belief in me put me where I am today, and to the love of my life, Sindhura Jois.

TABLE OF CONTENTS

Chapter 1: Review of Convolutional Neural Networks

Convolutional Neural Networks (here onward CNNs), are one of the pinnacle of breakthroughs in machine learning algorithms. Most Prominently, CNNs are used in image classification, image segmentation, object detection, action identification and large scale video recognition among many other applications of machine learning. The concept, used with multitudes of variations, has given state of the art results in all of the above applications. They have even been successfully used in Natural Language Processing settings like text classification. The invention of CNNs allowed for the departure from traditional fully connected artificial neural networks that were heavy on computer memory, thus leading to the generalizability of deep learning models to process larger and more complex datasets. In the context of this thesis, exclusive focus will be put on the image classification task in computer vision, which was the original inspiration for CNNs.

CNNs have been around in the engineering community since the late 1980s. Inspired by neuroscientific developments at the time and prior research in Neocognitrons [2] by Fukushima, Deep Convolutional Neural Nets were first proposed by LeCun et. al. in 1999 [3]. However, up until the mid-2000s, academia and industry alike agreed that CNNs could not be practically used for Image Classification due to existing convergence issues of the popular optimization algorithm Gradient Descent, combined with limitations in comput-

ing power. But, as aptly observed by The MIT Press, 2017 [4], CNNs saw a renaissance with the emergence of high performance computing on Graphics Processor Units(GPUs), and with the availability of large labelled image datasets that could enable supervised learning on CNNs. Since then, CNNs have led arguably one of the greatest success stories in deep learning research.

1.1 The Convolution Operation

The backbone of a CNN is the convolution operation. The key idea is to convolve a filter of trained weights across an input image-space to extract key features within the image. The extraction is performed using a simple element-wise multiplication of the filter weights and features of the input that lie within the receptive field of the convolutional filter. The output is a feature map that corresponds to locations in the input where the filter has observed the feature it was looking for. The variety of features detected can be increased by increasing the number of channels in the convolutional kernel. The key here is training the filter to extract specific patterns in an image that can add up to more meaningful representations that allow for classification of the input. Training, as with any deep learning algorithm, is done through the backpropagation algorithm where Gradient of the loss with respect to function operations within the algorithm are propagated backwards across hidden layers so that the weights may be updated. Another key aspect of Convolutional Layers is the introduction of non-linearity into feature maps after the convolution operation has been performed. This allows the algorithm to model complex non-linear relationships in images. When CNNs were first introduced, a number of non-linear activation schemes had already existed, for example, Sigmoid and Tanh transformations. However, the most influential among non-linearities were developed after CNNs were already introduced,

namely the Rectified Linear Activation Unit(ReLU). The technique of ReLU is to simply mask negative values in a feature space with 0, while allowing all positive values through.

It was soon discovered that stacking convolutional layers one after the other, in a deep architecture, allowed deep learning algorithms to model even the most complex features in an image. Lower level layers modelled simple, rudimentary patterns like lines and curves while the more higher level layers were able to model complex shapes. The convolution operation itself provided the advantage of being translation invariant. This means, a convolutional kernel can extract a feature regardless of its translations within the image. This was a major advantage in a field of neural network research that historically suffered from models not being able to correct for shifts in a feature's position within the image. The Convolution operation also enabled another very important advantage which is perhaps more responsible for its explosive success, parameter sharing. The major disadvantage in state of the art Artificial Neural Networks before CNNs was a parameter explosion on increasing the depth of an ANN. A fully connected layer introduces weights for every single possible connection between a lower layer and a higher layer. Naturally, a model with only fully connected layers becomes both computationally burdensome as depth increases, and also tends to overfit supervised learning tasks due to overparameterization. In comparison, the convolution operation enables usage of the same set of weights (inside the filter) across the whole image. The parameters saved through this particular usefulness of convolutions enabled the a marked increase in depth of ANNs using convolutional layers, while at the same time reducing the risk of overfitting compared to Fully Connected Layers.

1.2 Disadvantage of Very Deep CNNs

One important disadvantage that researchers started observing with the advent of Deep CNNs was that the depth of CNNs could not be increased indefinitely to model increased complexity in the data. It was observed after a certain depth the performance of CNN starts deprecating. The reason for this was the famous Vanishing Gradient problem. When the gradient of the loss is computed, it converges to zero on multiple iterations of the chain rule (the foundational procedure of Backpropagation). This means that the deeper the network, the more iterations of chain rule required to compute the gradient at shallow layers and hence the Vanishing Gradient problem arose. This particular problem was solved, in part, by the introduction of Deep Residual Networks by He et. al. in 2015 [5]. The core principle was to introduce skip connections across sets of convolutional layers so that gradients can flow quickly back to shallow layers, thereby helping solve the vanishing gradient problem.

1.3 Review of popular CNN Architectures

Over the last two decades, researchers have made astounding progress in image classification using CNNs. The foundational 7 layer CNN, LeNet [3], architecture proposed by LeCun et. al. in 1998, classified handwritten digits on cheques(MNIST) with an accuracy of 98.5%, which was a remarkable result for its time. AlexNet proposed by Krizhevsky et. al. in 2012 [6], resembled LeNet but was much deeper with more filter channels per layer. AlexNet achieves a Top1 error rate of 37.5% on the ImageNet dataset.

A very popular architecture introduced by Simonyan et. al., 2015, was

VGGNet [7] which was highlighted by a simple, yet powerful deep CNN architecture. The network uses simple 3x3 convolutional filters stacked on top of each other with increasing depth. A version of VGGNet, VGG16, can be seen used widely not only in image classification tasks but also as base networks in object detection and image segmentation tasks as well. VGGNet resulted in an outstanding performance in the ImageNet Image Classification challenge giving a Top 1 error rate of 24.7%. It also represented a marked shift in pushing the limits of depth of CNNs. However, VGGNet suffered a crucial bottleneck called the Vanishing Gradient problem as explained earlier. Resnet-18 [5] originally introduced by He et. al. in 2015 delivers a Top 1 error rate of 27.88 on ImageNet Validation set. However, since Resnets allowed a major increase in depth of a CNN, the original paper even introduced a 152 layer residual network that gave a top 1 error of 22.16%. This was a major improvement from classical Deep CNNs.

1.4 Limitations of CNN

While CNNs prove successful in a number of image classification tasks, they prove inadequate to deal with dimensions of affine transformations on the input image other than translations. Of course, some data augmentation techniques like random scaling, cropping, and rotations, help to generalize a model beyond the ability to adjust to local translations. However, to capture all possible affine transformations, the amount of labelled data needed for augmentation increase exponentially. Further, some transformations may be innately difficult to capture in a model.

A very good example of such an affine transformation is pose variance. When a CNN learns to recognize an object it is normally give example images

of objects captured from specific angle. CNNs however fail to deal with a change in pose of the object, i.e., an image of the same object from a different perspective. This proves to be a major drawback of CNNs as real world images are often subject to shifts in pose and perspective.

Researchers solved this problem in CNNs using MaxPooling, which is a way of decreasing the size of the feature maps and increasing the receptive field of higher level convolutional kernels. This allowed higher layers in CNN to capture disparate characteristics of the image at different parts thus solving the problem of pose variance to some extent. However, MaxPooling in CNN leads to a large amount of information loss. Even though Pooling works surprisingly well in practise, it still does not solve the core of the problem of pose variance. The focus in this thesis will be on a family of CNN models that do achieve pose invariance. Capsule Networks [1], introduced by Sabour et. al., 2017, is a promising new architecture that claims to get closer to the goal of achieving complete equivariance, which is the ability of CNN models to handle affine transformations of any kind in the input image.

Chapter 2: Introduction to Capsule Networks

In 2017, Sara Sabour, Geoffrey Hinton, and Nicholas Frost [1] introduced a new technique of neural computation which departs from traditional approaches of CNNs. One major difference is changing the scalar outputs of neurons to vector outputs. The authors define Capsules as vector entities that encode instantiation parameters of a detected feature, for example, its pose, skewness, rotation, etc. In turn, the magnitude of the vector describes the probability of existence of the feature. This shift in computational paradigm allowed Sabour et. al. to introduce an algorithm called Dynamic Routing by Agreement. The concept behind Dynamic Routing is to calculate the agreement of Capsule outputs between lower and higher level capsules independent of model training. The result of Routing is therefore based on the input image and the pose of its object.

It is easy to see how vectorization of neural outputs can enable the computation of 'agreement' using inner products. A higher agreement allows the algorithm to assign a higher coupling coefficient to the output of a lower level capsule. In theory, the algorithm dynamically allocates the correct low level features towards the assembly of high level features. Learning the instantiation parameters allows the network to recognize varying instances of the same object instead of re-learning every variation of features for the object. The authors show that such a technique generalizes better, and is more robust towards pose variance and perturbations than conventional CNNs. Capsule

Networks also reportedly require less data to train than CNNs for the same task.

One of the main advantages of CapsNet is the elimination of the Max-Pooling layer that is commonly used in conventional Deep CNNs. While max-pooling to a degree solves the problem of pose invariance in CNNs, it tends to lose information as well. CapsNet has been shown to allow high levels of information retention.

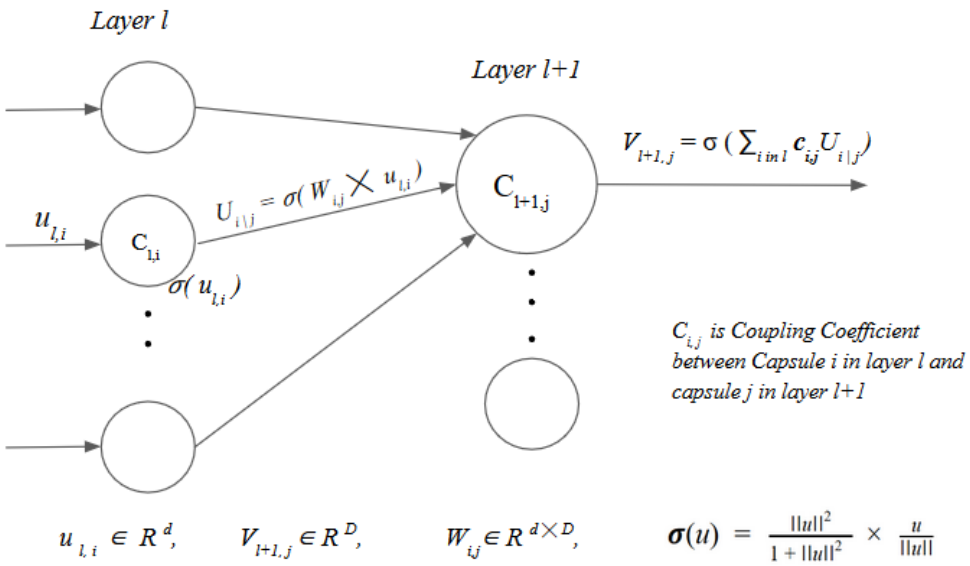


Figure 2.1: Schematic Representation of the Capsules

Algorithm 1: Dynamic Routing, proposed by Sabour et. al. [1]

Given: $u_{j|i}$, r, l, $l+1$
$b_{i,j} = 0$ for all capsule i in layer l and j in layer $l+1$
for $i = 0$ *to* r **do**
    for all capsule i in layer l, $c_i = Softmax(b_i)$
    for all capsule j in layer l+1, $v_j = Squash(\sum_i c_{i,j} u_{j|i})$
    for all capsule i in layer l and j in l+1, $b_{i,j} = b_{i,j} + u_{j|i} v_j$
end
return v_j

2.1 CapsNet applied to MNIST

Sabour et. al. designed the original CapsNet [1] implementation around the MNIST classification task. The network consists of only 3 layers; 2 Convolutional layers and a Fully Connected layer. The First convolution extracts features from the input single channel image, converting the 26 × 26 image into a 9 × 9 × 256 feature map. The second layer (Primary Capsule) consists of 8-dimensional capsules where each capsule is a group of 8 convolutional units. The capsules are spread over 32 channels. Dynamic Routing is implemented between the primary and secondary capsules. Each secondary capsule is a 16 dimensional vector containing the instantiation parameters of a digit. A 10 digit classification requires 10 such secondary capsules. The authors have used 3 routing iterations during dynamic routing. At each routing step, the algorithm computes the similarity between each primary and secondary capsule and assigns a higher coupling coefficient to capsules that match closely. Sabour et. al. introduce non-linearity into system using the squashing function given by $\sigma(u)$ in **Figure 2.1**. The function squashes capsules to have a maximum length of one. Capsules with low magnitudes are squashed toward zero, while capsules with high magnitudes possess an almost unit length after squashing.

2.2 Margin Loss

CapsNet uses a novel Margin Loss to establish presence of a digit. As is the basic idea of capsules, the length or magnitude of a digit capsule (secondary capsule) represents the probability of it existence. Margin loss penalizes the neural network if the length of the capsule representing the true class is not high enough. At the same time it penalizes the network if an incorrect capsule has a high magnitude.

2.3 Reconstruction Regularization

Sabour et. al. propose three fully connected layers for reconstruction regularization. The reconstruction loss is scaled down so as to not overpower the digit classification loss. Loss is calculated by the sum of squared differences between the true pixel intensities and the output of the final fully connected layer.

2.4 Performance on MNIST

The authors tested Capsule Networks for two types of classification tasks; image classification with one class per image, and image classification with multiple overlapping classes. The results on single class classification was reported as a 0.25% test-error rate which is state-of-the-art considering only much deeper networks can achieve these results. Without routing, the model performs worse by a margin of 0.04%, which mildly demonstrates the effectiveness of Dynamic Routing. The effectiveness is clearly demonstrated in the multi-class classification task with overlapping digits. The authors reported a 5% error rate which is on par with the sequential attention model of Ba et. al. [8] even with a much higher digit overlap. The major observation however was in the reconstruction of overlapping digits. Even with highly overlapping

digits, the model was able to color code digits separately, showing that Secondary capsules were actually encoding digit information which was later used during reconstruction.

2.5 Matrix Capsules with EM Routing

Shortly after the release of Capsule Networks with Dynamic routing, Hinton et. al. [9] published another paper with a vastly more sophisticated perspective on Capsules. In this version, routing is framed as an Expectation Maximization problem rather than an Agreement problem. Routing, in this algorithm is a clustering-like operation. Each higher level capsule is assumed to be a Gaussian that explains data from lower level capsules. In theory, if we have n higher level capsules and m lower level capsules then EM routing tries to find n Gaussian mixtures from m data points and appropriately assigns them to higher level clusters. The Capsule itself has a slightly different structure than the one proposed in the Dynamic routing paper. Each capsule consists of a 4 x 4 pose matrix M, that defines the instantiation parameters of a feature from the input image. In simpler terms, if a Capsule describes a circle, then matrix M for that capsule describes the properties of the circular shape as observed in the image and only depends on the input image. In addition, Hinton et. al. propose an additional scalar A that represents the 'activity' of the Capsule. This is slightly different from the first CapsNet work where the magnitude of Capsules represent the activities. The separation of 'pose' and 'activity' decouples the two types of information. This separation of magnitude and 'direction' is actually a common trick used in computation (for example, weight normalization by Kingma et. al. [10])

This new architecture was tested on the SmallNORB dataset. The best performing model achieves a test error rate of 1.4% which beats the state-of-the-art in SmallNORB classification. More significantly, the architecture was tested for adversarial attacks. In theory, Capsules should be able to resist adversarial perturbations as the pose matrices should describe said perturbations without directly affecting activities. Hinton et. al. tested their Matrix Capsule architecture on the FGSM adversarial strategy (Goodfellow et. al. [11]) and found that CapsNet is much less susceptible to adversarial attacks than conventional CNNs.

Matrix Capsules also reduced the number of trainable parameters by a large margin. The only downside of this architecture is its specificity to the problem at hand. The paper did not report performances of experiments on other conventional datasets, and it is not clear whether this is because the new CapsuleNet does not work well on those problems. Another possible issue one might observe is that the number of capsules in each higher layer is an important hyper-parameter that may need to be tuned for each specific problem. In the classical Gaussian Mixture clustering problem, the quality of clusters largely depends on the number of pre-defined clusters that EM is trying to group data points into. With EM routing in CapsNet, one may need to find the appropriate number of higher-level capsules for each individual problem. This is especially a concern when it comes to applying CapsNet in industry grade problems.

2.6 Current developments in Capsule Networks

One of the most interesting works following the original CapsNet paper of Sabour et al. is Sparse Unsupervised Latent Capsules[12] developed by Rawlinson et. al. who argue that the generalizability of CapsNet depends on the ability to use Capsule blocks in hidden layers. The reasoning of why this the case is further explained later when this thesis explores the same notion.

Rawlinson et. al. [12] test Capsules in hidden layers where the only mode of training is through backpropagation of Gradients and not through direct supervision. Note that in both, Capsules with dynamic routing and EM routing, Hinton's team use a custom loss function that directly supervises the Capsule's training. The researchers found that without direct supervision, hidden (latent) capsules lose their viewpoint invariance and pose equivariance properties that are desired. In their experiments, only a few capsules are active after training, and are responsible for almost all of the transfer of information from lower to higher layers. This thesis will show, how the lack of supervision of hidden capsules, in fact harms a model's performance. As an alternative, they propose a joint sparsification and online boosting algorithm that softly selects k winning capsules after routing before passing the information to higher layers. Experiments show that this preserves some of the desired qualities of capsules. They argue that sparsification is the key to increasing the depth of CapsNet. This thesis largely concurs with the findings of Rawlinson et. al. but will also point out a few pitfalls to their approach.

CapsNet has sparked a lot of interest in the field of medical AI. The following literature review provides notable examples where authors have extended CapsNet research with novel techniques. Mobiny et. al. use CapsNet to detect lung nodules during lung cancer screening [13]. The authors show that CapsNet performs significantly better than Deep CNNs when a small number of training samples are available. This is an interesting finding that validates the usefulness of CapsNet in many real world problems with less available data. Mobiny et. al. also use Transposed Convolutions during image reconstruction to reduce the number of parameters while improving performance during reconstruction. CapsNet has been used successfully in image segmentation problems as well. LaLonde et. al. [14] propose a U-Net (Ronneberger et. al. [15])like architecture for segmentation of pathological lungs in CT scans. Their SegCaps [14] model outperforms U-Net [15] in segmentation accuracy while drastically reducing network size compared to the original U-Net. Spectral CapsNet [16], proposed by Bahaduri to make diagnoses on patient time-series data, aims to increase the efficiency of EM-Routing algorithm of Hinton et. al. [9] by simplifying the computation of pose vectors of higher level capsules. Bahaduri's architecture uses one dimensional vectors rather than matrices for pose computation due to the sequential nature of the data.

With HitNet [17], Deliege et. al. [17] analyse the core behaviour of Sabour et. al.'s CapsNet [1] and identify functional aspects of the model that might in fact be counter-productive to the stated aim of CapsNet. The final Capsule output of the Sabour et. al. model is a 10 x 16 DIGITCaps layer where each 16 dimensional vector corresponds to a class (in the case of MNIST a digit). Based on the input image, the vector of the true class encode input dependent pose and feature information, while its magnitude represents the model's pre-

diction of probability of the classes presents. The margin loss is formulated in a way that pushes the magnitude of the true capsule far away from incorrect capsules and it does this by increasing the length of the correct capsule while squeezing the others. Now, let us consider a unit hypersphere which contains all ten 16 dimensional vectors. The incorrect vectors are points close to the center while the correct class is a point close to the surface of the hypersphere.

Deliege et. al. [17] argue that training the model in this way leads to loss of control over the part of hypersphere aimed at by the model to describe the true class. Additionally there is no way to ensure that two input images of the same class result in capsules that are close to one another. This is compounded by a squashing function that universally squeezes all capsules, meaning if many features within a capsule are large then none of the features will remain large after squashing. This inhibits how well a capsule can explain the input it is encountering. Deliege et. al., formulate an alternative 'Centripetal loss' that inverts the notion of training proposed by Sabour et. al. Instead of increasing the length of true DIGIT Capsules, Centripetal loss pushes true capsules toward the center of the hypersphere described above, while 'throwing' incorrect capsules out towards the surface. This ensures an added layer of control to the CapsNet model by grouping together capsule representations of the same class. They achieve model training by introducing a novel 'Hit-or-Miss' layer. Additionally, the researchers use Batch Normalization combined with Sigmoid activation to mimic the squashing while preserving large features.

The 2D perspective of Matrix Capsules [9] has been generalized to a 3D setting for video action recognition by Duarte et. al., 2018 [18]. The group uses 3D Convolutional Capsules and Capsule Pooling to reduce the number of trainable parameters in their proposed network. Additionally they use a localization network, that uses the instantiation parameters inherent to capsules to track action in videos. The technique achieves state of the art performance on a conventional action recognition datasets.

Jaiswal et. al., 2018 [19], show that Capsules can be used as Discriminators in Generative Adversarial Networks (Goodfellow et. al. [20]) and can outperform Conventional convolutional GANs in MNIST and CIFAR10 datasets. Nguyen et. al., 2019 [21], introduce a Capsule Network based embeddings model for search personalization and knowledge graph completion. Their CapsE model uses a Capsules ability to encode intrinsic spatial relationships transforming input knowledge graph matrices into final vector whose length measures the plausibility of the knowledge graph triplet. A most important work by James O'Neill [22], extends the idea of Siamese Networks (Sun et. al., 2014 [23]) for face verification, using Capsule Networks. His proposed model outperforms strong baselines even when face pairs in the test set were previously unseen.

Natural Language Processing has seen some developments using CapsNet. In theory, it is natural to view Natural Language as highly pose variant. Semantically similar sentences may be written in completely unique ways which is analogous to objects in images varying in pose and viewpoint. Wang et. al., 2018 [24], propose recurrent neural layers paired with Capsule blocks for Sentiment Analysis tasks. There proposed model achieves state of the art in standard datasets, proving the effectiveness of Capsules even in a Natural Language setting.

Chapter 3: Thesis Objective

If Capsule Networks can be validated over multiple datasets, then their generalization capacity may be proven therefore making Capsule Networks useful in the industry. The Capsule Network proposed by Sabour et. al. in both their original papers fundamentally explore the MNIST data-set and the SmallNORB dataset. While SmallNORB classification is complicated in its own right, it still does not prove a Capsule Network's performance in real world images. Real world images offer a degree of irregularity both in background and foreground that make image classification a quite complex problem. Even at 10 classes, the CIFAR10 dataset offers vastly greater complexity than MNIST. Indeed, Sabour et. al. do not report state-of-the-art performance of their CapsNet implementation on the CIFAR10 dataset. They report a 10.6% error rate on a 7 model CapsNet ensemble. However, ensembles in the real world setting are difficult to deploy due to increased time and resource expenditure during inference serving. The key motivation behind this thesis is to validate the performance of a single CapsNet model on the CIFAR10 and CIFAR100 datasets. Further, the thesis will report a thorough investigation of the pitfalls of CapsNet in a real world image classification setting. Building on the findings obtained, a number of approaches and model architectures will be explored that can help implement CapsNet on the CIFAR10 and CIFAR100 datasets. The thesis draws inspiration from a number of key ideas prevalent in the field of Deep Learning and in specific, ResNet [5].

Chapter 4: Capsule Networks for CIFAR10

The original capsule network architecture with dynamic routing results in state-of-the-art performance on the MNIST dataset. The same architecture was chosen as a baseline model to evaluate performance on CIFAR10 as it provided a good starting point for understanding the effect of dynamic routing on more complex image classification tasks.

A number of minor changes were made to the original capsule architecture for this experiment. The input channels were increased to 3 as CIFAR10 is expressed in the RGB range. The input dimensions were increased from 28 x 28 to 32 x 32. In the original paper, Sabour et. al. cut random 28 x 28 patches from the original 32 x 32 images of CIFAR10. From here on the final secondary capsule which gives the classified outputs will be addressed as CIFAR capsules. The number of output channels of the CIFAR capsules were increased from 16 (original implementation) to 32 in order to encode more information from the image. This is because MNIST is a much more simplistic image set than CIFAR and hence, more encoding capacity is required to explain the classes. All experiments were carried out for the same constant step size, to maintain uniformity over test conditions.

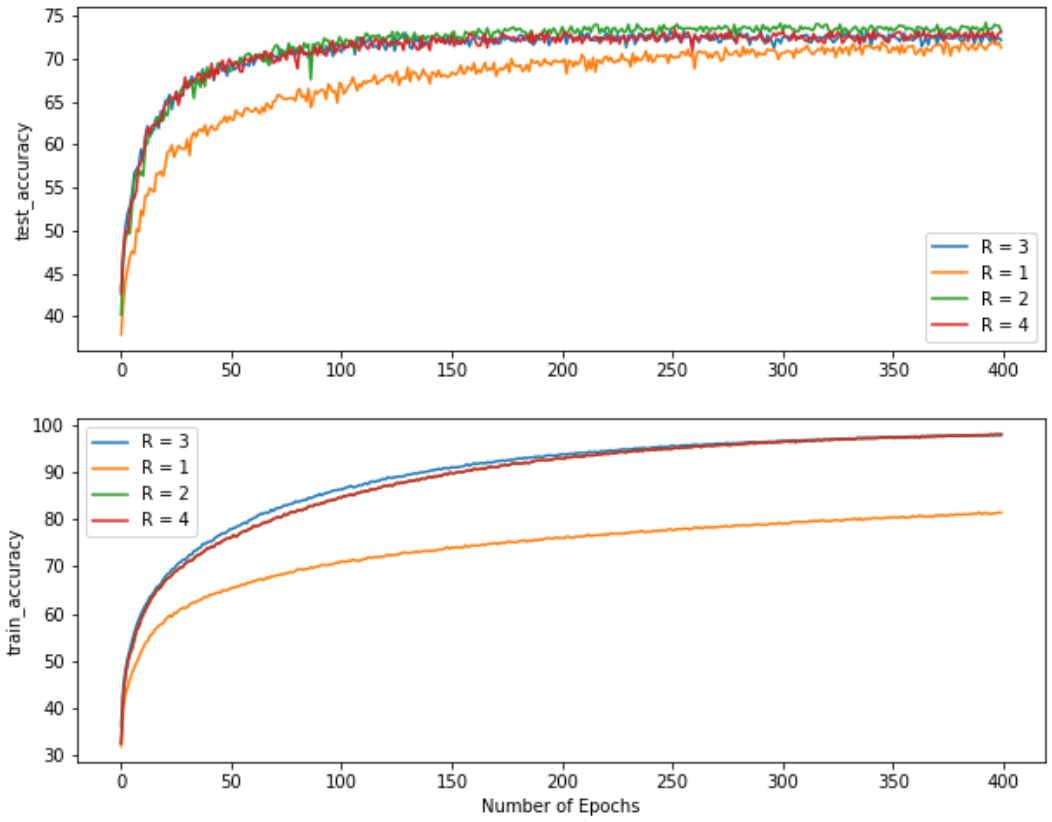


Figure 4.1: Comparison of performance in various routing conditions

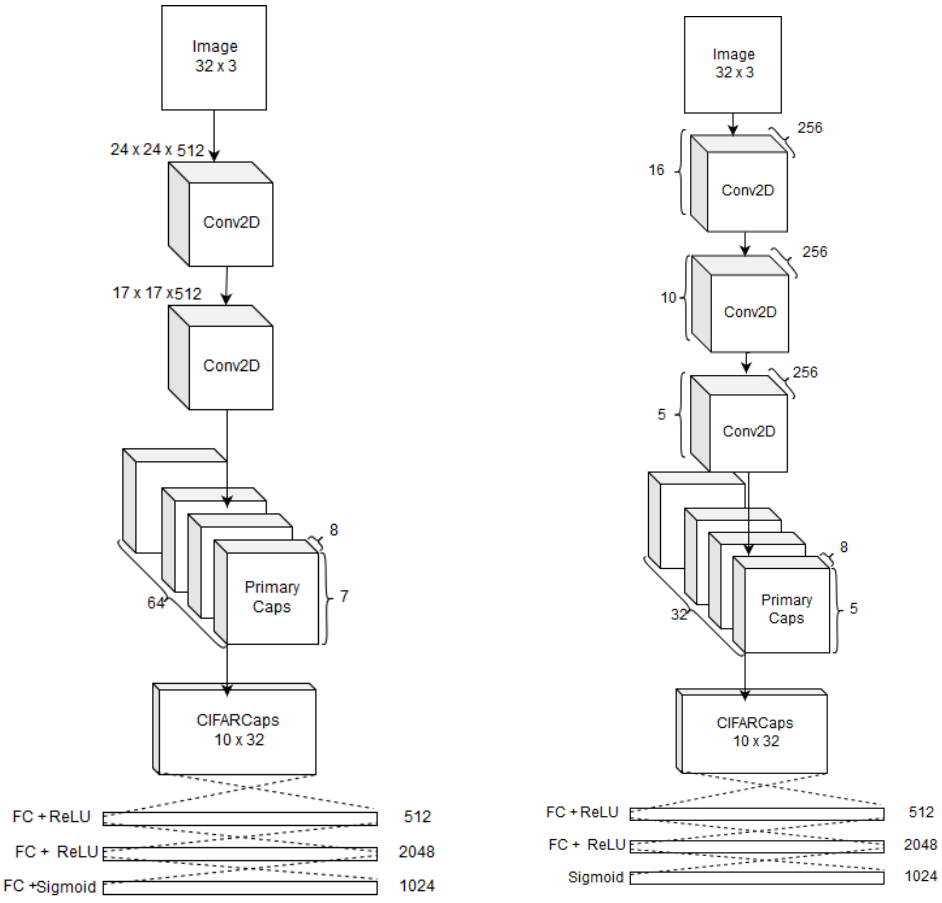
Figure 4.1 shows that the baseline Capsule Network implementation does not perform too well on the CIFAR10 dataset, which was expected. A three layer model does not have enough depth to accurately learn complicated feature representations. The high degree of overfitting points towards the model not learning high level representations but instead trying to fit as many low level features from the training set as possible. However, the experiment provides a good basis to understand the effect of dynamic routing in Capsules. The model achieves a higher test-accuracy with number of routing iterations greater than one. It is important to note that with one routing iteration the

model is almost identical to a standard neural net with 2 convolutional layers and a fully connected layer. Therefore this routing condition can be taken as a measure of performance of a standard neural net of similar depth on the CIFAR10 classification task.

While the authors report the best test accuracy at three routing iterations, the baseline performs best at two iterations on the CIFAR10 dataset. More iterations lead to greater overfitting. It can also be noted that dynamic routing allows the model to learn faster, evidenced by the test and train accuracies saturating much earlier for iterations greater than one. This is an important feature as the training time of model was noted to be much higher than conventional neural nets. This is owing to the fact that each forward pass during a training iteration requires n computations of coupling coefficients for n capsule routing iterations.

4.1 Stacking Convolutional Layers

Building upon the observations above, further experiments were carried out by increasing the depth of the network incrementally. The Primary Capsule layer was preceded by additional Convolutional layers to increase the feature extraction capacity of the network (shown in **Figure 4.2**). The experiments were performed for three routing iterations.



(a) Capsule Network with 1 additional Convolutional layers

(b) Capsule Network with 2 additional Convolutional layers

Figure 4.2

As shown in **Figure 4.3**, increasing the depth of the network does not observably improve the best test accuracy. When two convolutional layers are stacked before the primary capsule, the model learns features quicker, but saturates at around the same maximum test and train accuracy as the baseline model. Furthermore, increasing the number of convolutional layers does not decrease the tendency to overfit. The reason for the observed effects may be due to difficulties in model training and are not entirely clear at this point. We conjectured that vanishing gradients may be the cause. However, it is unlikely for vanishing gradients to take effect for networks this shallow. Further targeted experimentation is needed to identify the problem.

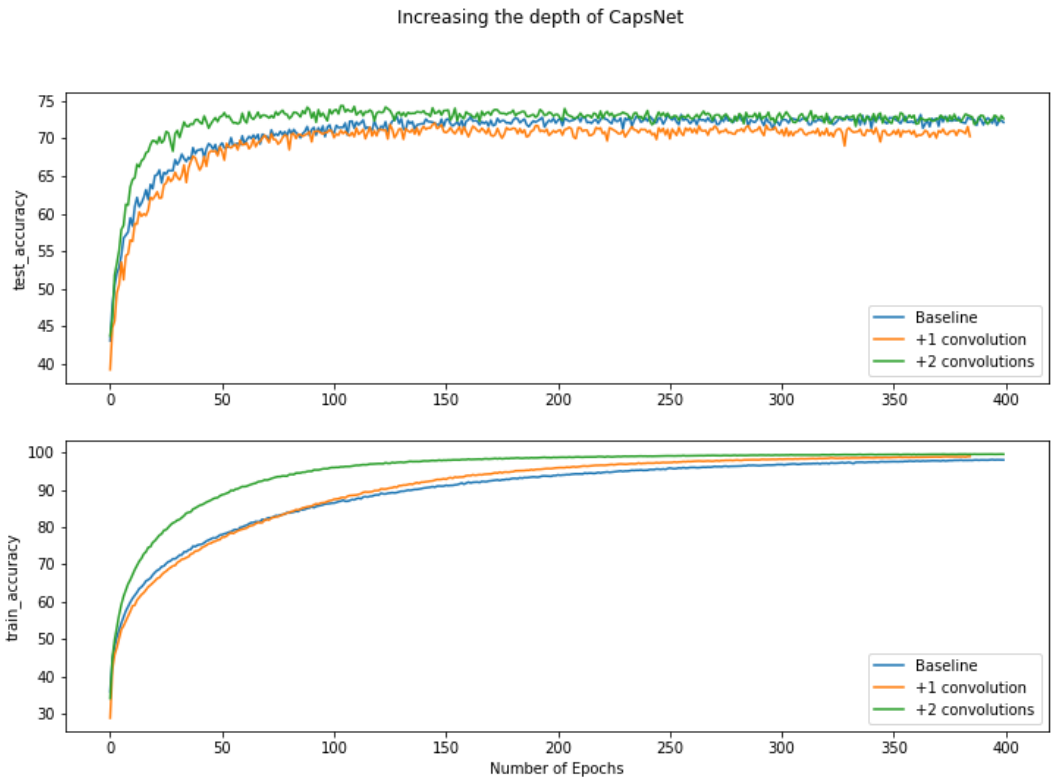


Figure 4.3: Performance comparison on increasing depth of capsules

4.2 Residual Capsule Networks

Based on preceding experiments, we propose to introduce residual blocks consisting of primary capsules. The architecture described in this section shows how one might construct Capsule Residual Blocks with skip connections joining a lower level primary capsule to the routing enabled secondary capsule. It should be noted that the Capsule Residual Block differs from traditional Residual Blocks introduced by He et. al. [5]. Unlike the original ResNet where output feature dimensions of residual blocks are independent of subsequent computation, our model forces primary capsules to share the same transformation weight matrices prior to routing. Therefore, we must force primary capsules in two separate branches to be identical in dimensions to maintain the same number of capsules. The optimal proposed architecture (**Figure 4.4**) passes the input image through a Convolutional layer to extract all features. The output of the convolutional layer is passed to a Primary Capsule which forms the Convolutional Capsule block preceding the skip connection. The output of the first convolutional layer is also passed to the wide residual connection through two additional Convolutional layers onto an identical Primary Capsule layer. The outputs of both Primary Capsules are merged and then squashed. The merger now produces a temporary capsule tensor which is dynamically routed through to the final CIFAR capsule layer. Note that in terms of total number of neural layers, this model differs from the nearest stacked convolution model in only one additional Primary Capsule layer. Also, weighted transformation of Capsules and subsequent routing takes place on the combined feature maps of one shallow and one deep capsule.

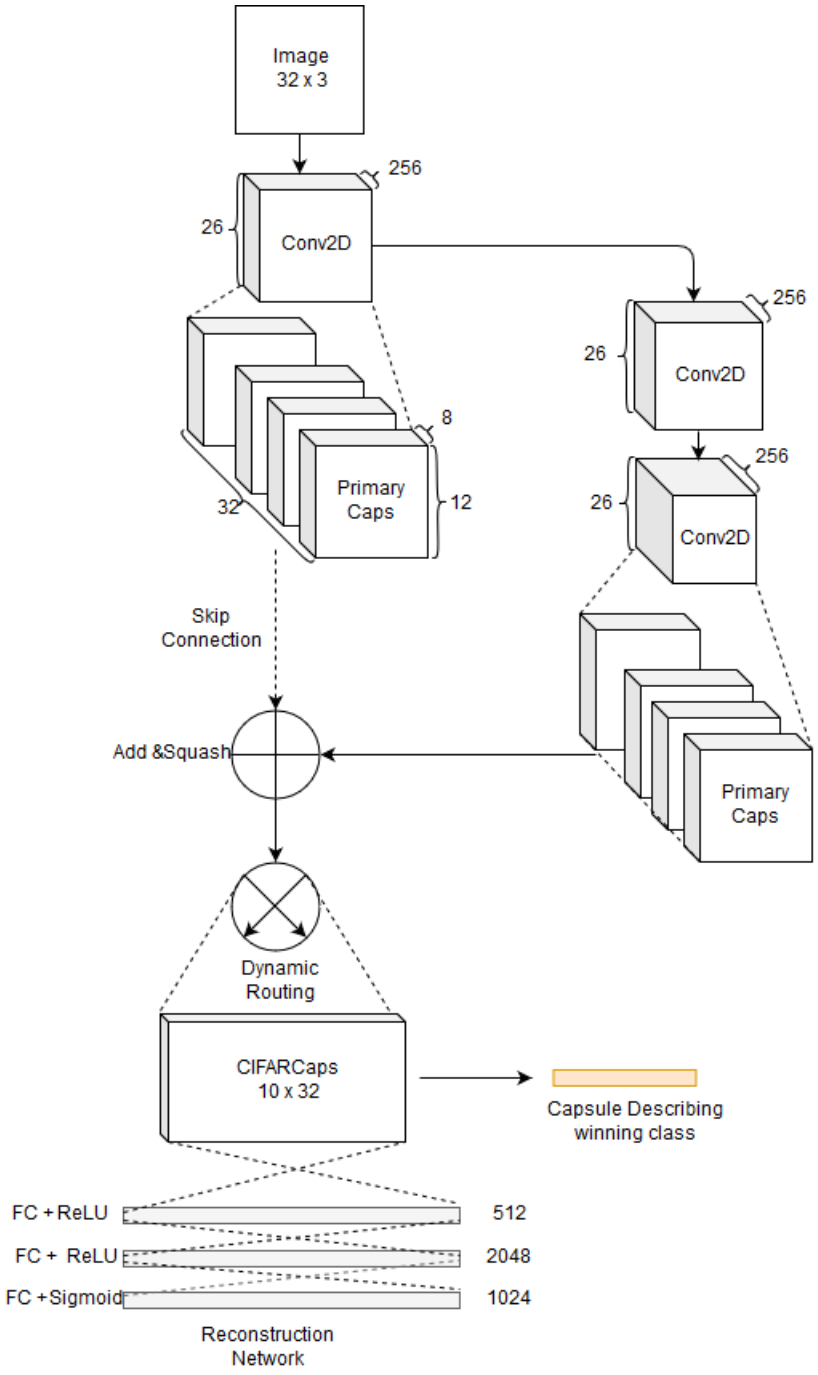


Figure 4.4: Architecture of Residual Capsule Network

In the experiments, we adopt a slightly modified version of the above architecture, where the signal splitting occurs at the output of the primary capsule rather than the first Convolutional layer. This was done due to ease of implementation. However, this makes the architecture slightly unstable due to numerous tensor manipulation operations.

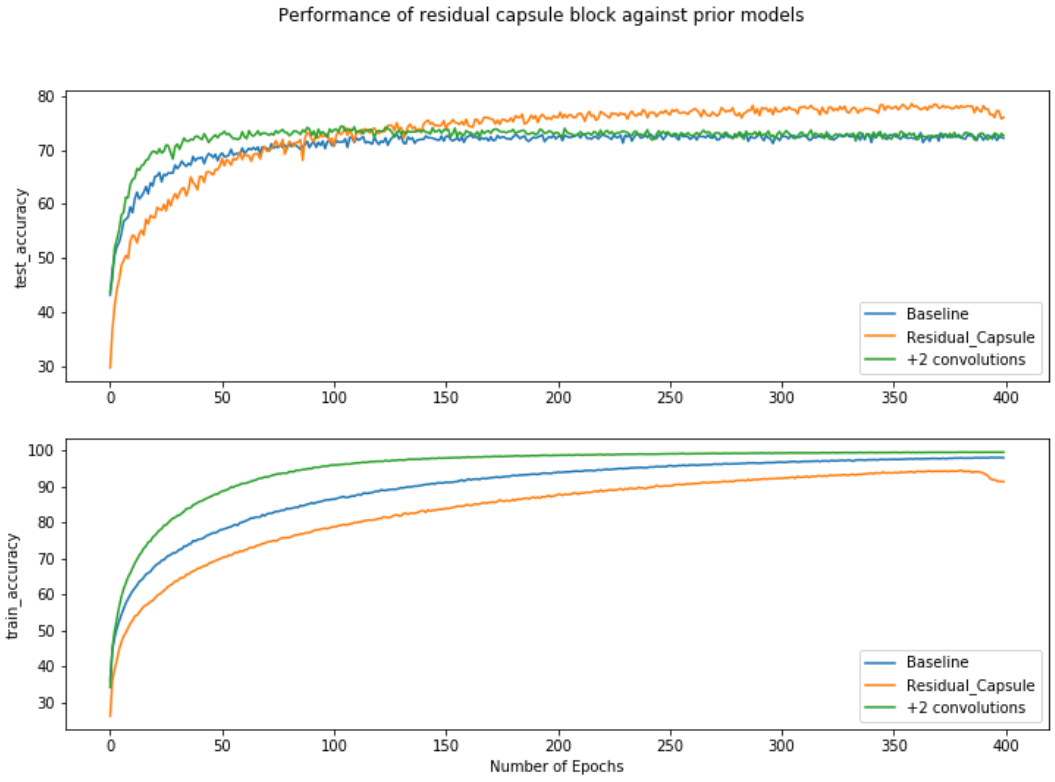


Figure 4.5: Performance of ResCapsNet against prior models

	Model	Routing iterations	No. of Params	Test Acc.(%)
1.	Baseline	1	22m	71.94
2.	Baseline	2	22m	74.16
3.	Baseline	3	22m	73.19
4.	Baseline	4	22m	73.44
5.	+2 Conv	3	27m	74.35
6.	ResCapsNet	3	**25m**	**78.54**

Table 4.1: Performances of various Capsule architectures on CIFAR10 classification

The first observation is the dramatic improvement in test accuracy when the Residual capsule block is used. Compared to the Baseline CapsNet with 1 routing iteration (a standard neural net), this model delivers a 6.54% increase in validation accuracy. It outperforms the nearest best performing CapsNet model with two additional stacked convolutional layers by 4.19% while using 2 million less parameters. It can also be observed that the model overfits much less this time around even though it takes greater epochs to achieve the same results as its simpler counterpart. The highlight of the result however, is that capsules placed in independent branches of residual blocks seem to reinforce each others learned features. Both Primary Caps layers share the same transformation weight matrices and are successful in imparting pose-invariance through dynamic routing. The technique reduces the need for any additional weights prior to routing apart from the one's introduced by convolutional layers. We conjecture that the model can be improved by introducing skip connections to more than two primary capsule layers placed at arbitrary depths within a deep CNN, prior to a final routing layer, that is, we propose to include additional nested branches of Capsules where each branch pushes deeper into the network and all capsules branches merge toward one routing module.

Chapter 5: Limitations of CapsNet

It is evident that Capsules and the mechanism of Capsule coupling show promise. With further research and application Capsule Networks may even replace traditional CNNs. However, there are a number of limitations that were discovered during research and experimentations. These limitations need to be addressed before Capsules can be applied in any useful way within the industry.

The practical aspects of training and inference serving are a cause for concern. While dynamic routing certainly increases the network's capacity to assemble parts of a whole, it is also a time consuming process. The more capsules within a network, more are the parameters that need to be computed at each routing iteration. Consequently, the wider or denser the network, the more time required for dynamic routing.

Investigating further into the problem of memory usage, in a typical CapsNet, the number of trainable parameters in a Capsule layer is orders of magnitude higher than a typical Fully Connected layer. Consider a transformation weight matrix $W \in R^{D \times d}$ where D is the dimension of a higher level capsule and d corresponds to a lower level capsule. For j secondary capsules and for a convolutional capsule with k channels and edge length l, we obtained $d \times D \times l \times k \times j$ many trainable parameters. Even if all the individual quantities in the above expression are reasonable, increasing even one parameter

leads to an explosion. Compare this to a fully connected layer that has simple one 2D transformational weight matrix. As a result, the number of capsules in the final layer are a major hindrance toward applying capsules to large problems. As we switch from CIFAR10 to CIFAR100 classification, we see that parameters explode by a factor of 10.

Indeed the number of discriminatively learned weights comprise the largest proportion of the original Capsule Network model parameters. This is in stark contrast to the philosophy of Convolutional layers which make deep learning models much more lightweight by sharing weights across an input feature map.

In the current dynamic routing regime, it is intractable to place a capsule at the final layer of CIFAR100 classification models due to parameter explosion. At the same time, training CapsNet requires a strongly supervised learning mechanism using the Margin Loss, which explicitly forces secondary capsules to encode pose information. It is not possible to achieve this unless Capsule layers are placed at the end of a neural network model. Hence, there is a depth limitation for supervised capsules.

Complex problems require larger depth. The depth of Capsule networks can only be increased by stacking standard convolutional blocks and for problems that require more than 10 layers of depth, Capsules in the final layer serve little to no purpose as the major contribution is that of prior convolutional layers.

Chapter 6: Supervised Hidden Capsules for CIFAR100

The inability to place capsules in hidden layers limits the generalizability of CapsNet. It is easy to see that performance gain can be maximized if hidden layer capsules behave the same way as Digit Caps in Sabour et. al.'s CapsNet [1], and encode the same equivariance relationships for hidden features. Sparse Unsupervised Capsules [12] by Rawlinson et. al., are a potential solution to this challenge, and can enable deeper Capsule layers. However, Rawlinson et. al. do not address the parameter explosion problem that limits the size and width of hidden capsule layers. Further, an unsupervised approach removes control over the kind of equivariance relationships and pose information that we would like to encode. We explored a possible solution that uses supervised hidden capsules for CIFAR-100 classification. However, the efficacy of the solution is still under research and validation. The model is therefore discussed as a concept and the experimental results highlight some of the failures of this approach.

The architecture takes advantage of the 20 superclasses of CIFAR100. Instead of learning to encode every individual class for all 100 classes the model forces capsules to encode superclass information and fine tunes the capsules using the final classification loss. Due to the complexity of CIFAR100, a head network is used to extract features prior to Dynamic Routing. Normally this can be a simple ResNet model. In this thesis, the Wide-Residual Network [25] by Zaguroyko et. al. is used. Two different losses are used, Sabour et. al.'s

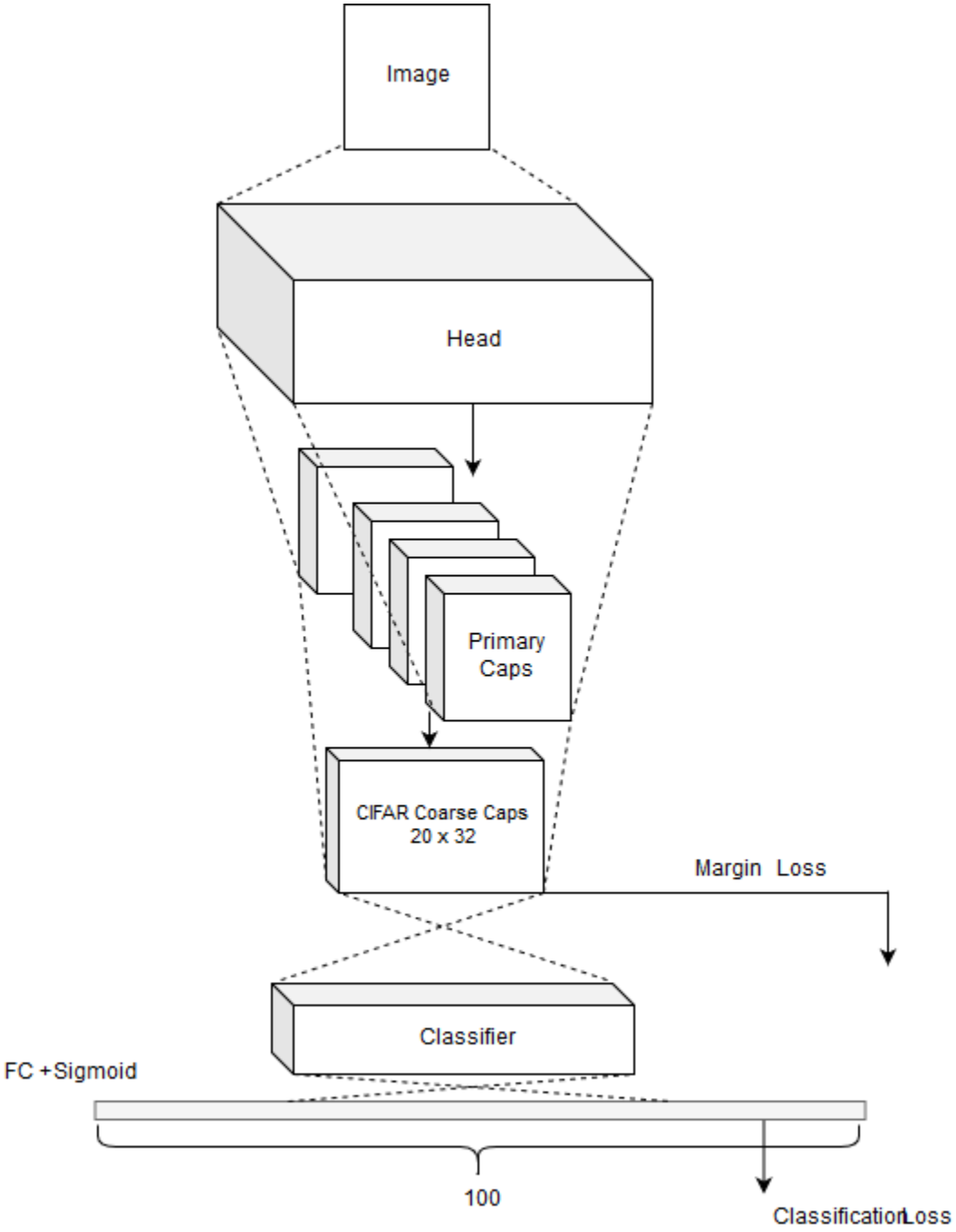


Figure 6.1: High level Concept for Supervised Hidden Capsules

Margin Loss and Cross Entropy loss.

The Wide ResNet blocks extract input features, and pass their output to the Primary Capsules. The Capsules are routed to 20 secondary capsules, called Coarse Capsules for 20 coarse classes, each being 32 units long. Margin Loss is then used to compute loss between the capsule output logits against true superclasses. Finally, two fully connected layers (in this case, 2 FC layers were chosen as the Classifier network) transform capsule outputs to a 100 dimensional softmax output. The fully connected layers are explicitly trained using Cross Entropy Loss computed against fine class labels, and gradients are backpropagated through the whole network in order to fine tune Capsules and the head network. By using capsules as hidden layers, the architecture reduces the number of learnable weights in the capsule layers by a factor of five while maintaining the width of capsule layers to encode as much information as possible. In essence each 32 dimensional coarse capsule is assumed to be an embedding of the input image specific to that coarse class (super-class). When the fully connected layers succeeding the capsules are trained, the capsules encode fine-tuned input image embeddings to resolve all fine classes within a coarse class. Fully connected layers share weights across all 20 coarse classes to make the model lightweight. The training of capsules combined with the squashing function ensure that for each input image the fully connected layers receive emphasis on the position of the winning capsule along with its encoded information.

Training the network can be done using multiple strategies. One strategy is to combine the final classification and superclass classification loss. The coarse margin loss and fine cross entropy loss may be added after scaling down

the margin loss. In theory this would emphasize training towards the final prediction of classes while weakly supervising capsules to encode coarse class information. The loss can be formulated as,

$$Loss = \lambda \times Margin\ Loss + Cross\ Entropy\ Loss$$

During experimentation, a range of scaling parameters for Margin Loss were tested but during each experiment, the model broke down completely and failed to train. There are two likely explanations for the observed behaviour, one being that Margin Loss entirely dominates the final classification loss no matter what the scaling coefficient. The more likely explanation might be due to some incorrect backpropagation of Margin Loss Gradients across the classification layer. The problem might be solved by performing two different and independent backpropagation steps where Margin Loss gradients do not pass the classification network but Cross Entropy gradients are allowed to backpropagate across the whole network.

As an alternative, the simpler and more assured method of training would be a two-step training protocol. First, we exclude the classification network while training capsules to detect coarse classes. In this step, only the margin loss is used against the output logits of Capsules. In the second step, the FC layers are connected to capsules for final classification of classes and Cross Entropy Loss is used to train the entire network. This method however, has not been tested and subject to future work.

One might question the worthiness of an architecture that takes advantage of pre-existing semantic groups among classes to supervise Capsule training.

But this is actually not a far fetched notion. Most real world problems with high class counts likely contain semantic groupings that can be made use of. In fact, the idea is similar to Hinton et. al.'s [9] work of finding latent Gaussians in higher level Matrix Capsules. Except in this case, it is a much more controlled method of making Capsules behave the way we want them to. Many Machine Learning problems require a clustering based pre-exploration of hidden communities in data. The same cluster assignments may be used to determine the number of hidden Capsules.

6.1 Unsupervised Hidden Capsules

Scaling Margin loss to zero results in an unsupervised hidden capsule layer. This is the case where, according to Rawlinson et. al. [12], only a few randomly assigned capsules remain active and are reinforced throughout the training period thereby losing properties of equivariance imparted by dynamic routing. We tested the performance of unsupervised hidden capsules against our Wide-ResNet benchmark classifier.

In **Figure 6.2**, it is clear that not only do unsupervised capsules lose desirable pose in-variance properties but also impede model performance. This should be quite likely if only a few Capsules are active and more signals are being 'squeezed' through the active capsules. The findings validate our previously stated hypothesis. It is either necessary to sparsify capsule outputs or to find a way to supervise hidden capsule training even if the training is weakly supervised.

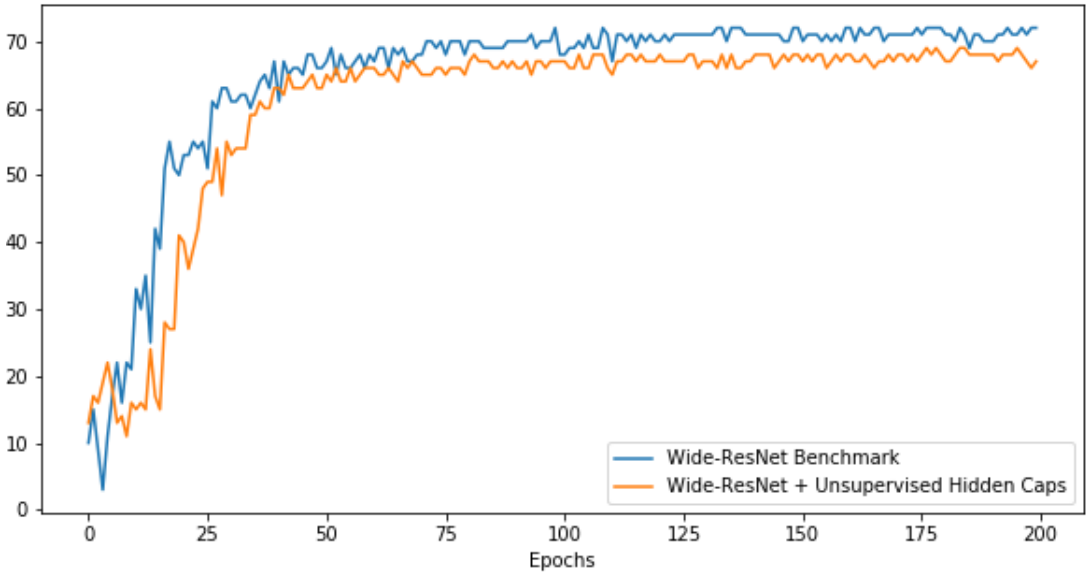


Figure 6.2: Performance of Unsupervised Hidden Capsules against Wide-ResNet baseline (Test Accuracy)

Chapter 7: Future Work

First and foremost, back-propagation of Gradients in the proposed Hidden Capsule architecture needs to be perfected. If the joint loss computation seems unfeasible then a two step training process will be explored. Possibly an intermediate verification step is required to monitor the Hidden Capsule's prediction accuracy with respect to coarse classes. If validated, only then can we proceed with the fine tuning step. We need to also explore the generalization of SparseCapsule ([12]) where sparsity constraints are introduced while simultaneously supervising Hidden Capsules. This would generate a much more robust model, combining the positives of both approaches.

Through the course of this thesis, an interesting question arose but could not be explored due to time constraints. A large portion of the Capsules limitations seems to be associated with how we are training the transformation weights. However, no researcher has so far explored if we need learnable weights between routing capsules at all. In Sabour et. al.'s CapsNet, a major reason why trainable weight matrices are used is to make lower and higher level capsules mutually compatible for Agreement computation. If we impose a constraint of equal dimensionality then agreement can simply be framed as a problem of finding the best non-linear combination of vectors at each capsule layer. This makes routing similar to existing notions of Attention Mechanisms.

A successful implementation of agreement routing without transformation weights would enable us to arbitrarily place routing modules in hidden layers, which is one step closer to the complete generalization of Capsule layers.

Leading up to the development of ResCapsNet, there was evidence to suggest that Dynamic Routing somehow inhibits learning representations in shallow layers. Normally, shallow neural networks tend not to overfit as easily as CapsNet does. Since vanishing gradient is unlikely at such depths, we conjecture that there may be two possible explanations for this. Dynamic Routing or any kind of Agreement based routing has a tendency to self-bias. Once a 'favorite' primary capsule has been chosen on account of random weight initialization, the same choice for all subsequent inputs is only reinforced as the coupling coefficients converge in favor of the 'favorite' capsule. This may lead to existence of inactive capsules which propagates back toward shallow layers, thereby limiting the amount to which they learn. The second argument is that Gradient backpropagation in Capsules Networks occurs through the 'unrolled' routing steps. The effect of accumulation of coupling coefficients on the differential with respect to a lower level capsule leads to either an exploding gradient or a very low gradient depending on the magnitude of coupling coefficient. A more theoretical approach is needed to examine model convergence during routing, complemented by experimental examination of gradients in shallow layers. Another interesting future line of research would be to run targeted experiments to examine the distribution of capsules that are inactive or active at the end of training.

In the broader context, our work on Residual Capsule block needs to be generalized to Matrix Capsules with EM routing. To a large extent the parameter explosion problem may be solved if our method of increasing network depth by sharing weight matrices across capsules can be integrated with the lightweight nature of Hinton et. al's. proposed Matrix Capsule architecture [9].

A major future work would be to study adversarial attacks on Capsules. The growing challenge for the AI industry is that of protecting CNNs against malicious attacks. In that respect, CapsNet may be at the frontline of adversarial resistance in neural networks. Hinton et. al. have tested Matrix Capsule's [9] resistance to adversarial attacks but only using the FGSM strategy [11] by Goodfellow et. al. A larger scale of study is required to determine the true qualities of adversarial resistance exhibited by both Matrix Capsules [9] and Vector Capsules of Sabour et. al [1]. A comparative adversarial attack study between the two types of Capsules applied on the same routing protocol, would show us if the decoupling of 'pose' and 'magnitude' in Matrix Capsules actually helps build adversarial resistance. We conjecture that Capsules could be modelled to have three distinct descriptive quantities rather than two. In addition to 'pose' and 'magnitude', Capsules could also model the 'perturbation' aspect of inputs to separate adversarial perturbations from the desired signal.

Chapter 8: Conclusion

In this thesis, we explored an explosive new field of research that has sparked wide-spread interest in the machine learning community. Within a year of its introduction, Capsule Networks had already beaten state-of-the-art benchmarks in a variety of computer vision and and even natural language related tasks.

This thesis took a closer look at the research behind Capsule Networks, focusing specifically on one aspect of its performance; Are Capsule Networks applicable to complicated, noisy, real world image classification problems? The CIFAR10 and CIFAR100 datasets were taken as test cases for this study. A number of limitations were discovered that made the generalization of CapsNet difficult. CapsNet natively has a tendency to over-parameterise and overfit in more complex problems. In order to extend Capsules to complex datasets, Residual Capsule Networks (ResCapsNet) was introduced. We found that our proposed model beat the baseline CNN in validation accuracy by 6.6% with only a 13% increase in the number of parameters. More significantly, it beat an equivalent capsule network of similar depth by 4.19% in validation accuracy, while using 8% lesser parameters. With this result, we found that primary capsules placed at different depths that are made to share transformation weight matrices prior to routing, retain qualities of pose and viewpoint in-variance. This finding is an important step towards generalizing CapsNet for larger problems.

Preceding experiments and contributions were all made under the condition of direct supervision of Capsule training through the margin loss. However, on the CIFAR100 problem, we found that placing secondary 'routing' capsules in hidden layers without direct supervision does not lead to improved validation accuracy. We argue that without a custom loss function directly supervising the training of the Capsule layer, Capsules tend to lose their pose and viewpoint invariance properties. We propose a method to integrate margin loss of the original CapsNet by Sabour et. al to train hidden capsules as super-class detectors. However, due to errors in network design and incorrect backpropagation, we failed to validate the performance of supervised hidden capsule. Alternative step-wise training protocols are an interesting future direction of research.

References

[1] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *CoRR*, vol. abs/1710.09829, 2017.

[2] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, Apr 1980.

[3] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, (London, UK, UK), pp. 319–, Springer-Verlag, 1999.

[4] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017. PMID: 28599112.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learn-*

ing Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[8] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[9] G. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with em routing," 2018.

[10] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *CoRR*, vol. abs/1602.07868, 2016.

[11] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[12] D. Rawlinson, A. Ahmed, and G. Kowadlo, "Sparse unsupervised capsules generalize better," *CoRR*, vol. abs/1804.06094, 2018.

[13] A. Mobiny and H. V. Nguyen, "Fast capsnet for lung cancer screening," *CoRR*, vol. abs/1806.07416, 2018.

[14] R. LaLonde and U. Bagci, "Capsules for object segmentation," 2018. cite arxiv:1804.04241.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[16] M. T. Bahadori, "Spectral capsule networks," 2018.

[17] A. Deliège, A. Cioppa, and M. V. Droogenbroeck, "Hitnet: a neural network with capsules embedded in a hit-or-miss layer, extended with hybrid data augmentation and ghost capsules," *CoRR*, vol. abs/1806.06519, 2018.

[18] K. Duarte, Y. S. Rawat, and M. Shah, "Videocapsulenet: A simplified network for action detection," *CoRR*, vol. abs/1805.08162, 2018.

[19] A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan, "Capsulegan: Generative adversarial capsule network," 02 2018.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[21] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, and D. Q. Phung, "A capsule network-based embedding model for knowledge graph completion and search personalization," *CoRR*, vol. abs/1808.04122, 2018.

[22] J. O' Neill, "Siamese Capsule Networks," *arXiv e-prints*, p. arXiv:1805.07242, May 2018.

[23] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 1988–1996, Curran Associates, Inc., 2014.

[24] Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu, "Sentiment analysis by capsules," in *Proceedings of the 2018 World Wide Web Conference*, WWW '18, (Republic and Canton of Geneva, Switzerland), pp. 1165–1174, International World Wide Web Conferences Steering Committee, 2018.

[25] S. Zagoruyko and N. Komodakis, "Wide residual networks," *CoRR*, vol. abs/1605.07146, 2016.