RESIDUAL CAPSULE NETWORK

A Thesis

Submitted to the Faculty

of

Purdue University

by

Sree Bala Shruthi Bhamidi

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2019

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

GururBrahma GururVishnu GururDevo Maheshwaraha
Guru Saakshaat ParaBrahma Tasmai Sri Gurave Namaha

This thesis is a dedication to my parents Lakshmi Kameswari Bhamidi and Seshasai Bhamidi for believing in me and my sister, Sarvani Bhamidi for giving me that extra push, I needed everytime I fell back.

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Dr. Mohamed El- Sharkawy for his patience. The door to his office was always open whenever I ran into a trouble spot or had a question about my work. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

I would also like to extend my gratitude to the Department of Electrical and Computer Engineering, IUPUI, for the IoT Laboratory and the facilities made available to us. A special thanks to Sherrie for making this process easy and helping us with the deadlines.

Last but not the least, thanks to my friends Saranya Chitta and Priyanka Vedullapalli for helping me get through the sleepless nights.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
BN	Batch Normalization
CapsNet	Capsule Network
CIFAR	Canadian Institute For Advanced Research
CNN	Convolution Neural Network
CV	Computer Vision
DCNet	Dense Capsule Network
DCNet++	Diverse Capsule Network
EM	Expectation-Maximization
FC	Fully Connected
GPU	Graphics Processing Unit
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
NN	Neural Network
ReLU	Rectified Linear Unit
ResNet	Residual Network
YOLO	You Look Only Once

ABSTRACT

Bhamidi, Sree Bala Shruthi. M.S.E.C.E., Purdue University, August 2019. Residual Capsule Network. Major Professor: Mohamed El-Sharkawy.

The Convolutional Neural Network (CNN) have shown a substantial improvement in the field of Machine Learning. But they do come with their own set of drawbacks. Capsule Networks have addressed the limitations of CNNs and have shown a great improvement by calculating the pose and transformation of the image. Deeper networks are more powerful than shallow networks but at the same time, more difficult to train. Residual Networks ease the training and have shown evidence that they can give good accuracy with considerable depth. Putting the best of Capsule Network and Residual Network together, we present Residual Capsule Network and 3-Level Residual Capsule Network, a framework that uses the best of Residual Networks and Capsule Networks. The conventional Convolutional layer in Capsule Network is replaced by skip connections like the Residual Networks to decrease the complexity of the Baseline Capsule Network and seven ensemble Capsule Network. We trained our models on MNIST and CIFAR-10 datasets and have seen a significant decrease in the number of parameters when compared to the Baseline models.

1. INTRODUCTION

A dictionary definition to machine learning includes:

"A branch of artificial intelligence in which a computer generates rules underlying or based on raw data that has been fed into it"

Machine Learning usually refers to tasks associated with Artificial Intelligence (AI) such as, recognition, control, prediction, etc. One of the important reasons why machine learning (ML) is important is to help us understand the behavioral tendency of animals and humans. Few of the other reasons being to extract the relationships among piles of raw data, understand and produce the desired characteristics of machines in a working environment, helping the machines adapt to the ever-changing environment. With a constant change in technologies and its events, ML can help us track the changes and maintain a record of the same.

Convolutional Neural Network (CNN) is a bizarre combination of mathematics and biology, with a little of Computer Science involved. But this combination has been the most influential innovations in the field of Computer Vision. CNNs have become an integral part of services of every company, big or small. Companies like Facebook, Google, Amazon, and Instagram have been using neural nets to improve their tagging algorithms, photo search, product recommendations or even to improvise their home feed. CNNs are one of the reasons why deep learning is so popular today. They can do things that people never imagined a computer would be capable of doing. Nevertheless, they do come with limitations and few fundamental drawbacks.

Even with a history of more than 20 years, CNNs have shown that there is always a room for improvement. Since the concept of Deep Convolutional Network [1] has been rolled out, we can see a significant improvement in challenging tasks like Image Classification and Image Recognition. Though the deeper networks did improve the performance of the neural network models, stacking of layers brought in a new problem of vanishing gradients. This problem has been alleviated with the introduction of a new network Residual Network (ResNet) [2]. The new network introduces Skip connections between the layers i.e., the outputs of the previous layers are added to the outputs of the stacked layers in a feedforward manner. Adding these connections not only decrease the number of parameters, but also helps in concatenating the feature maps for a better gradient flow across deeper networks.

1.1 Motivation

As discussed above, Convolutional Neural Networks are our go-to algorithm when it comes to object recognition or object detection. Even after modelling the networks to emulate the human brain, there are many things that are very unlike the brain that are making the CNNs work not as well as they could. One thing that is missing in Neural Nets is the notion of entity. Geoffrey Hinton in his paper [1], has discussed the idea of introducing the concept of entities into the architecture. Convolutional Neural Nets use multiple layers of feature detectors which are replicated across space. These feature extractors with subsampling pooling layers, attend only to the active features.

"The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster." - Geoffrey Hinton

Pooling gives only a small amount of translational invariance at each level that is, the exact location of the most active feature extractor is ignored. Pooling also reduces the number of inputs to the next layer of the feature extractor. The downside to pooling is that they fail to use an underlying linear manifold which would deal easily with the effects of viewpoint. We do not want the neural activities to be invariant of the viewpoint instead we want the knowledge of the viewpoint which can be applied to a new viewpoint.

Convolutional Neural Nets try to make the neutral activities invariant to small changes in viewpoint by combining the activities of the pool. But it is better to aim at equivariance where the changes in viewpoint correspond to neural activities.

1.2 Contribution

This thesis primarily focuses on reducing the number of parameters in Hinton's Capsule Network. The proposed frameworks: Residual Capsule Network and 3-Level Residual Capsule Network are examined by modifying the baseline Capsule Network to reduce the parameters without a significant drop in the accuracy. The proposed architectures integrate features of Residual Network and Capsule Network for Image Classification.

We also discuss the concepts of few long-standing architectures such as Residual Network [3], Capsule Network [1] and YOLOv3 [10]. This thesis modifies the baseline Capsule Network to make the network deeper and handles the problems that come with it. Residual Network is known for dealing deeper networks to help handle sophisticated deep learning tasks and models. The proposed Residual Capsule Network is capable of reducing the number of parameters by 6.09% by compromising on the accuracy by 0.01% when compared to Baseline Capsule Network and 34.74% reduction on architecture built by Dense Convolutional Network at the cost of 0.09% accuracy on MNIST dataset.

When it comes to complex datasets such as CIFAR-10, the Residual Capsule Network has shown a room for improvement in terms of complexity. Additional layers were added to Residual Capsule Network to get promising results in terms of complexity as well as accuracy. The proposed 3-Level Network outperformed seven ensemble Capsule Network in reducing the number of parameters by 89.35% and affecting the accuracy by 2.98% on CIFAR-10 dataset. On the other hand, the proposed 3-Level Network was successful in reducing the number of parameters by 19.40% when compared to DC++Net with a compromise of 3.29% in accuracy.

2. BRIEF OVERVIEW OF CONCEPTS

Artificial Intelligence has been trying to narrow the differences between the proficiencies of humans and machines. Researchers especially in the field of Computer Vision are working towards making things happen. The agenda for this field is to enable machines view the world as the humans do. Birds have inspired humans to build flights and horses have inspired us in building cars. We draw resemblance between machines and human brain. We are still progressing in making the machines to be as powerful as human brain. This chapter gives a brief introduction of the concepts of Convolutional Neural Network and the architectures used in the contribution towards the proposed frameworks.

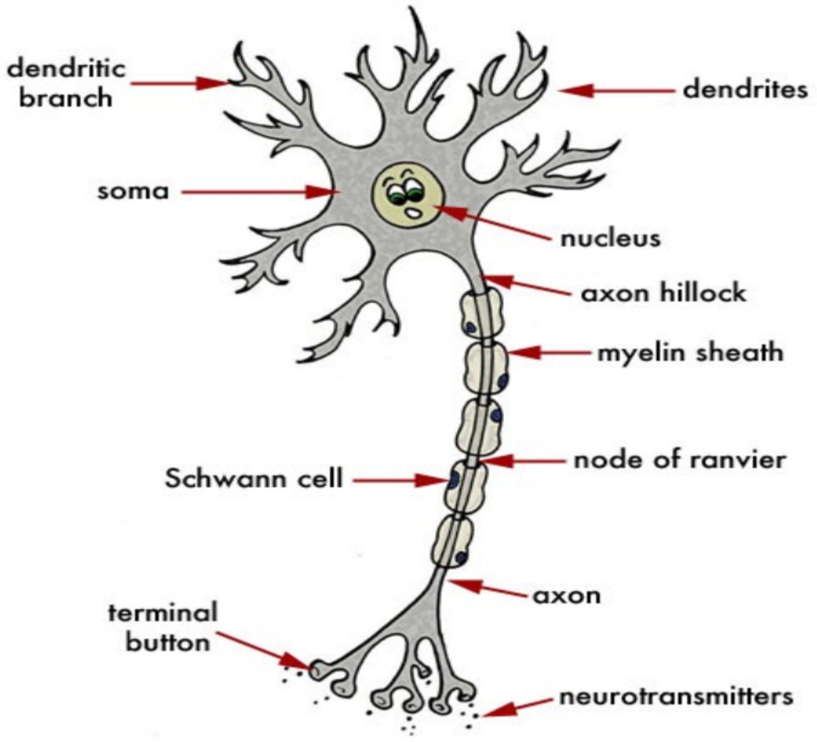


Fig. 2.1. Nerve Cell [24]

The human brain comprises of neurons (or nerve cells) which process and pass the information received from the senses. Many nerve cells like the ones described above are arranged in our brain to form a network. The information from one end of the nerve is passed on to another using electrical impulses. The dendrites carry impulse received from synapse of adjacent neuron to the nucleus of the neuron known as soma. The electrical impulses are further processed and transmitted to the axon. The axon carries the impulse from soma to synapse which is further transmitted to another neuron. Thus, making the human brain, a complex network of neurons.

A similar concept of network of neurons is adapted in machine learning. Here, the neurons are artificially created by the computer.

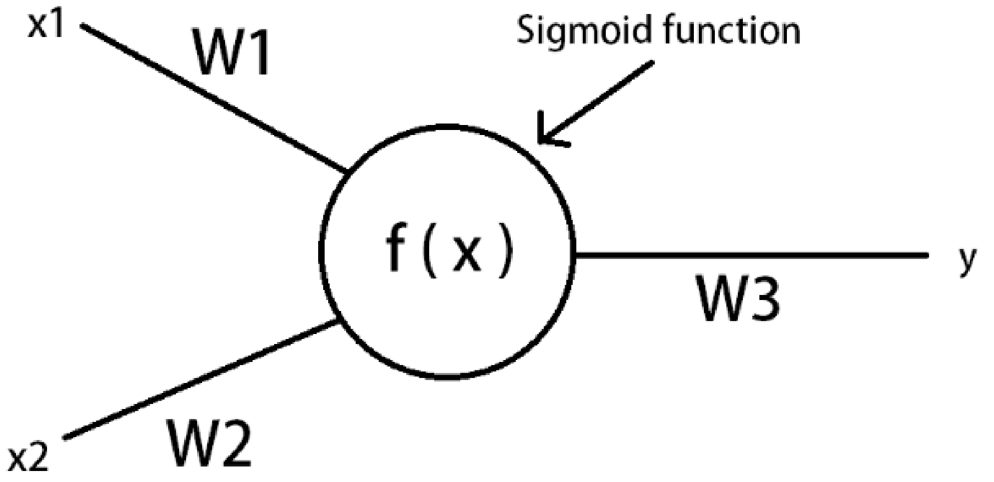


Fig. 2.2. Artificial Neuron [24]

If Fig. 2.2 is assumed to be an artificial neuron, the transmission of information (i.e., the flow of data) in the network is through a connection made by each neuron. Each of these transmissions has a specific weight. In this case, if we consider the input data to be x1 and x2 then, these inputs pass through connections W1 and W2 respectively which are adjusted depending on the connection weights. The weights of the connections are adjusted by multiplying the input with its weight and adding them together. For processing this data, an activation function f(x) is used. The values of the weights are assumed and calculated for the first iteration. The final prediction or output is the product of activation function f(x) and weight W3.

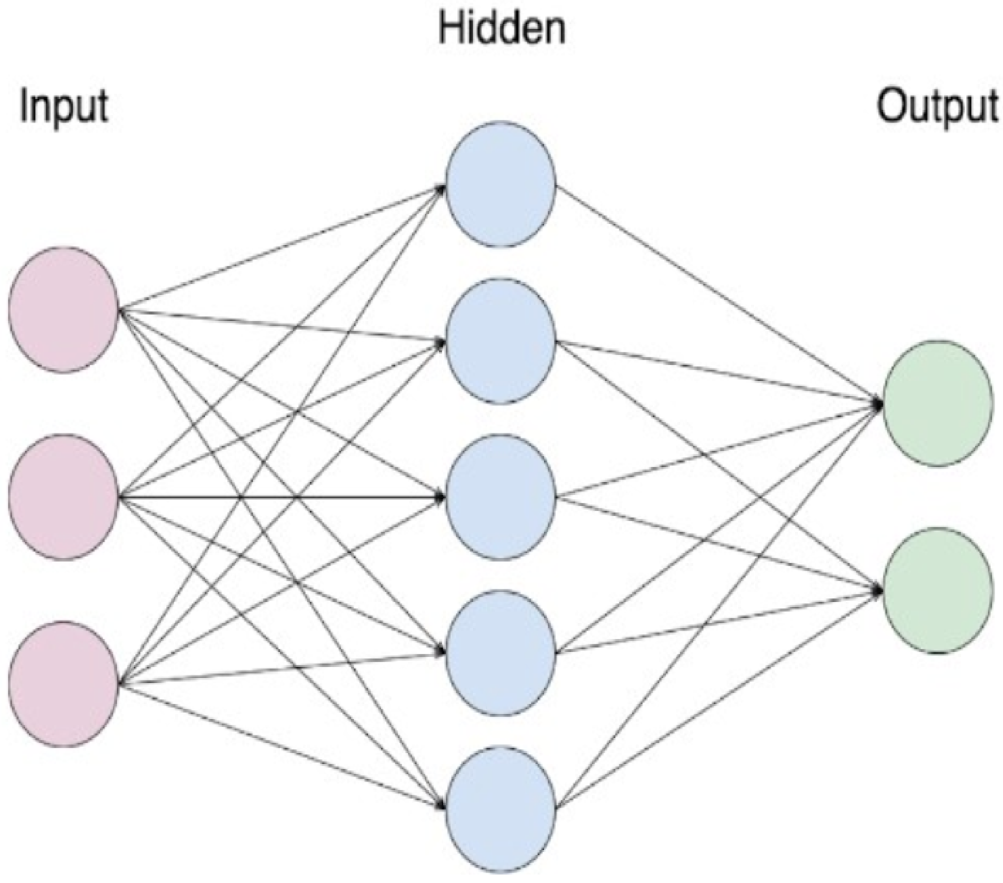


Fig. 2.3. Artificial Neural Network [25]

Drawing the analogy, an artificial neural network can be structured into three different layers for receiving, processing and transmitting. In other words, the input layer, hidden layer and the output layer as shown in Fig. 2.3. The input layer receives information from external sources and transfers it to the hidden layer. The hidden layer is a collection of neurons with activation function to process the information received from the input layer. Problems with complex decisions can have more than one hidden layer. The output layer collects the information transmitted by the hidden layer and presents the output in the designed form.

2.1 Convolutional Neural Network

A Convolutional Neural Network is an algorithm which can take an input image, assign learnable weights and biases to various objects present in an image and be able to distinguish one from another. The network comprises of input layer, output layer and several hidden layers in between. The pooling layers, fully connected layers and normalization layers are the hidden layers that converge either using multiplication or dot product. The inputs and outputs of these layers are masked by an activation function (usually RELU), followed by a final convolutional layer.

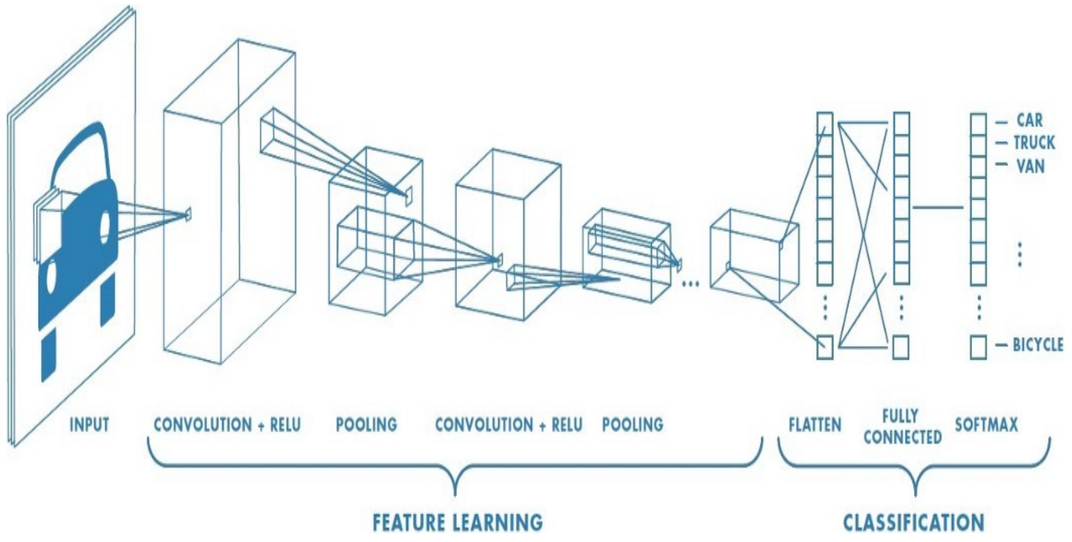


Fig. 2.4. Convolutional Neural Network [26]

The architecture of Convolutional Network is comparable to that of the structure of the human brain and the neural connectivity of the Visual Cortex. Individual Neurons respond to the stimuli only in a restricted region of visual field known as the Receptive field.

2.1.1 Convolutional Layer

The Convolutional layer is the fundamental block of a CNN. The parameters of this layer comprise of a set of learnable filters which have a small receptive field extended over the depth of input image. Each Convolutional layer in a neural network has the following features: input tensor, number of images, width, height, depth of the image, convolutional filters whose height and width are hyper-parameters and the depth is same as that of the input image.

The objective of convolution is to extract the high-level features of the input image. The first layer of convolution extracts the low-level features of the input image such as: edges, colour, etc. Convolutional Networks are not restricted to one convolutional layer. With more layers, more high-level features are also extracted which helps in better understanding of the images in the dataset.

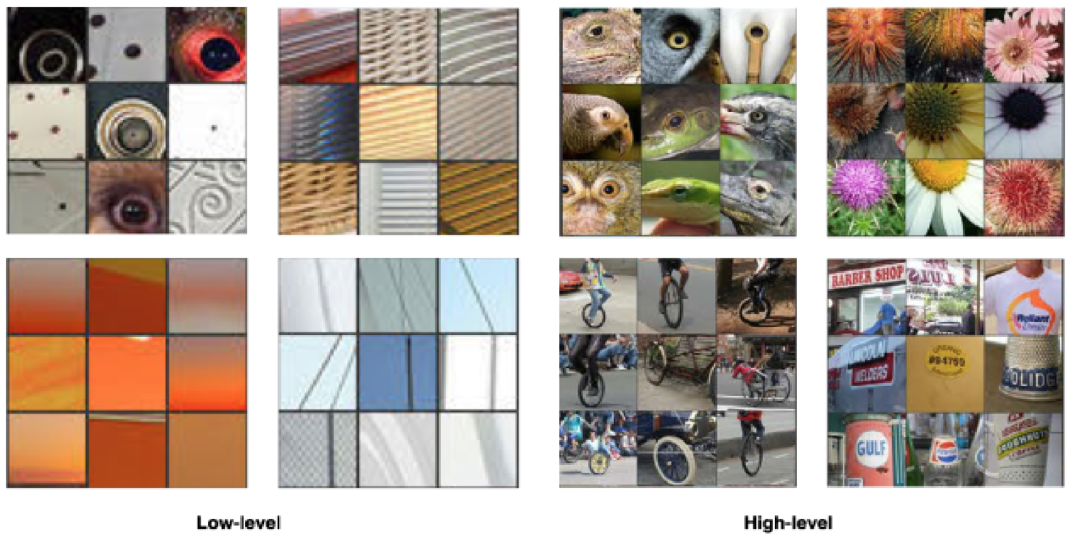


Fig. 2.5. Different Low level and High-level features [27]

There are two types of convolution results — one is done by applying Valid padding, in which the output of convolution is reduced in dimensionality compared to the input image, and the other is done by applying Same padding, in this the dimensionality of the input image is either increased or kept same.

The possible values for the size of padding P, is determined by the parameters of input W, size of the filter F and the stride S. the output size is given by:

$$\text{Output size} = (W - F + 2P) / S + 1$$

2.1.2 Pooling Layer

Like the Convolutional layer, Pooling layer is responsible for reducing the dimension of the convolved feature. This helps in reducing the computational power required to process the data by decreasing the dimensionality. It is also useful for extracting the rotational and positional features, hence effectively training the model. There are two types of pooling Max Pooling and Average Pooling. Max Pooling returns the maximum value from the cluster of neurons covered by the filter. Average Pooling, as the name suggests returns the average value of the cluster of neurons covered by the kernel.

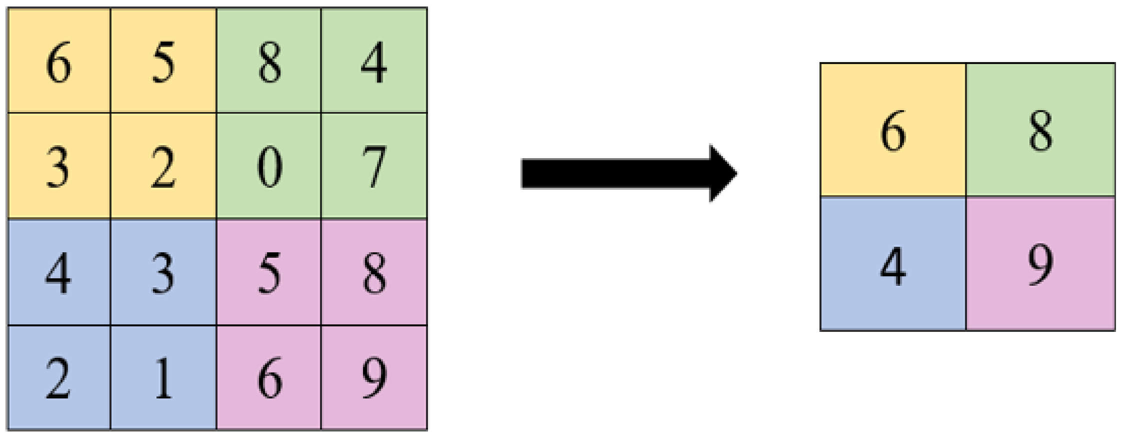


Fig. 2.6. Max Pooling

Max Pooling also suppresses the noise from the activations along with reducing the dimensionality. On the other hand, Average pooling reduces the dimensionality as part of suppressing the noise. Clearly, Max Pooling performs better than Average Pooling.

2.1.3 Fully Connected Layer

Fully Connected Layer connects every neuron from the one layer to every neuron in another layer. Fully Connected layers are way of learning non-linear combinations of high-level features as represented by the output of the convolutional layer. The fully connected layer takes the input and gives a vector output whose dimension is equal to the number of classes. This layer predicts the features of a class based on the output received from the previous layer. In other words, the FC layer strongly correlates the high-level features to a particular class and calculates the probabilities for different classes from the assigned weights.

2.2 Residual Network

The universal function approximator theorem [13] states that:

"An arbitrary continuous function, defined on [0,1] can be arbitrary well uniformly approximated by a multilayer feed-forward neural network with one hidden layer (that contains only finite number of neurons) using neurons with arbitrary activation functions in the hidden layer and a linear neuron in the output layer."

It implies that the accuracy increases with increase in the number of layers but, there is a limit on the number of layers to be added to increase the accuracy. Since, neural networks are not universal function approximators, the learning process faces vanishing gradient and dimensionality problems. If we keep increasing the number of layers, the accuracy keeps increasing and at a certain point, the network becomes saturated. Once the network is saturated, increasing the layers would eventually degrade the accuracy. At this point, the shallower networks appear to be better that the deeper counterparts. This is known as degradation problem.

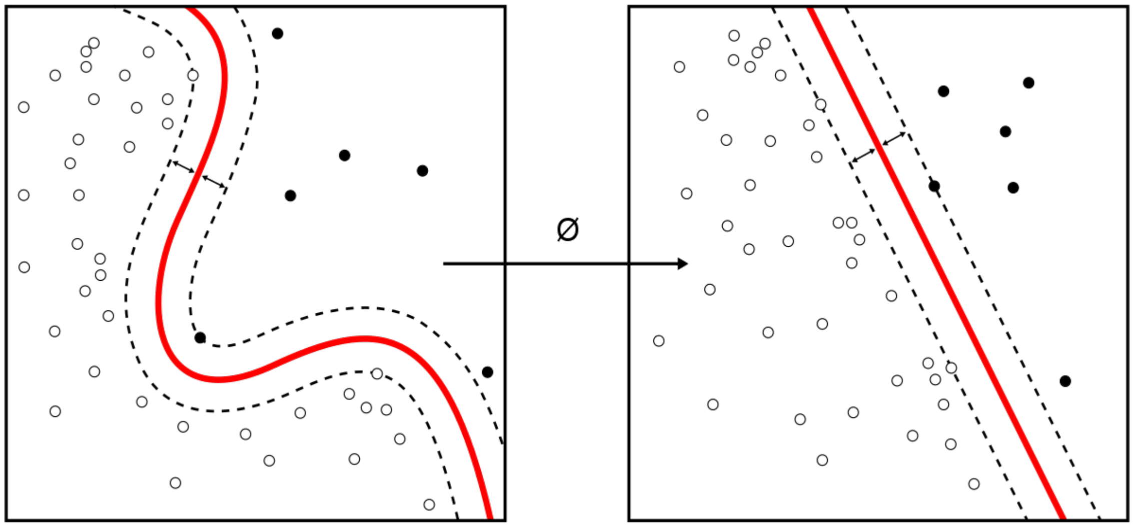
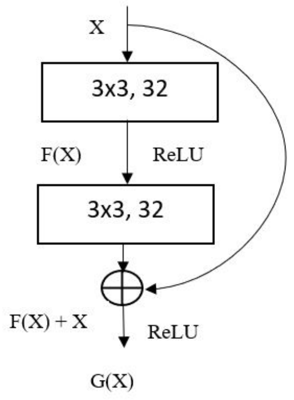


Fig. 2.7. Vanishing Gradient Problem [28]

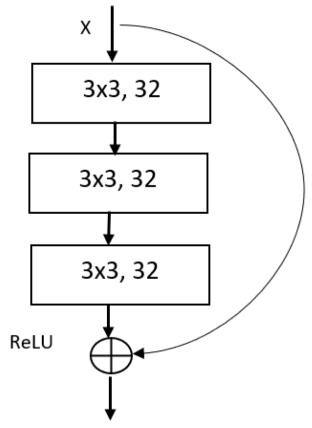
Residual Networks (ResNets) [3] are possibly the most innovative work in the field of Computer Vision/ Machine Learning community in the last few years. ResNets can train many layers, maybe hundreds or even thousands of layers without compromising on the performance of the network. Since then, the performance of many applications such as Image Classification, Object detection, Image recognition have remarkably increased.

In a feedforward network, a single layer with sufficient capacity is enough to represent any function. But the single layer maybe huge and the network is prone to overfitting. To avoid this, the network should be made deeper. Though the deeper networks did improve the performance of the neural network models, stacking of layers brought in a new problem of vanishing gradients due to the saturation of the network. Before the ResNet, there have been various ways to solve this problem. For example, [12] adds an auxiliary loss in the middle layer but it did not handle the problem. This problem has been alleviated with the introduction of a new network — Residual Network.

The concept of residual block is simple to understand. In a conventional neural network, the output of each layer is fed in as an input to the next layer. In case of networks with residual blocks, the output of each layer is fed into the next layer as well as into the layer that is 2-3 blocks away.



(a) Block-2



(b) Block-3

Fig. 2.8. Residual Network

The Residual Network adds Skip connections (Fig. 2.8) between the layers i.e., the outputs of the previous layers are added to the outputs of the stacked layers in a feedforward manner. As a matter of fact, ResNets werent the first to add skip connections in the network. Highway Networks [4] (Fig. 2.9) were the first of architectures to successfully train deep networks with large number of layers. Highway Networks with hundreds of layers can be optimized and trained effortlessly using gating units. These gates control the flow of amount of information across the connection. However, Highway Networks did not outperform Residual Networks.

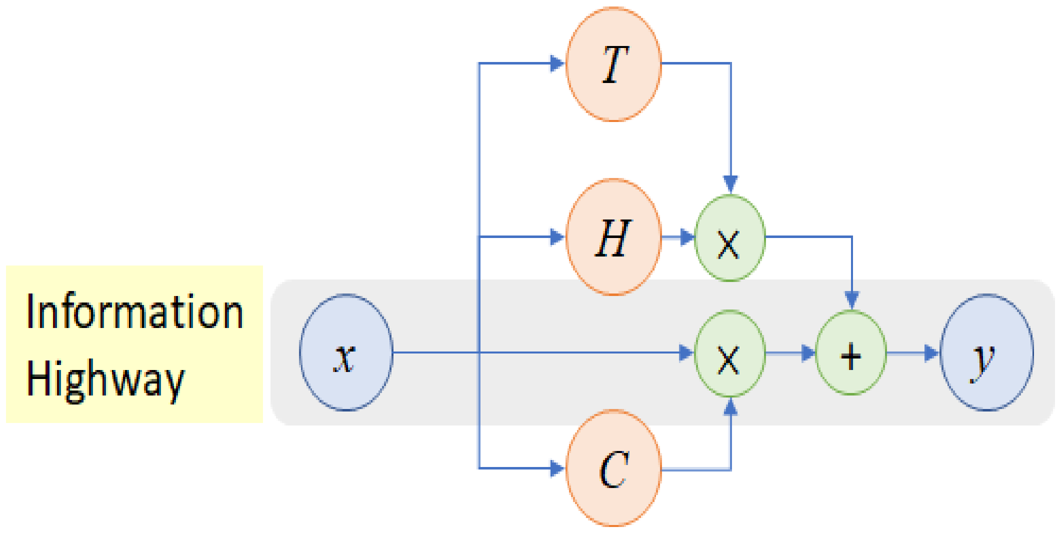


Fig. 2.9. Highway Circuit [29]

Giving a different perspective to the Residual Networks [3], the authors proposed a pre-activation residual block [14], in which the gradient flow through the skip connections to other layers without any hindrance. To overcome the degradation problem, we can skip train few layers using the skip-connections. The skip connections are also known as identity connections since we can directly learn an identity function trusting on the skip connections only.

From Fig 2.8, if we consider x to be the input and G(x) be the output distribution, then we can represent the difference to be:

$$G(x) = F(x) + x$$

$$F(x) = \text{Output} - \text{Input} = G(x) - x$$

The layers in the conventional network are learning the output G(x), whereas layers in the residual block are learning the residual F(x). Hence, called the Residual block. It is easier for the network to learn both the input and the residual output rather than just the input. This way, the network can learn identity function by setting residual to zero.

The ResNet architecture holds up to 152 layers. This includes convolutional, pooling and fully connected layers. We can say with confidence that this network gives better results (accuracy) compared any other model, provided the amount of training data is right. The Residual Network eliminate the issue of vanishing gradient. The gradient becomes very small that a large change in the input image does not affect the output much. With skip connections, the number of steps taken as the algorithm goes deeper into the model are reduced. This means that many of the linear functions, which have used the gradient/ weight function are not hindered with. Hence, it takes longer for the model to vanish or explode.

By introducing skip connections which are used as bypassing paths, Residual Networks have achieved a striking performance on Image classification and Image recognition tasks. We can say that skipping has effectively simplified the network and increased the learning speed by decreasing the impact of vanishing gradients as there are fewer layers to propagate through.

2.3 Capsule Network

CNNs have been accomplishing feats that have been exceeding peoples expectations. Convolutional Neutral Networks have been hugely successful in the field of deep learning and are main reason why deep learning is trending now. There are multiple features of the neural networks that are very unlike the brain, and which make them work inefficiently. Firstly, any complex engineered system should have various levels of structure. Neural networks have very few levels of structure. There are layers of neurons present in the neural network but are unlike the layers in our cortex. Another drawback of neural networks is that there is no explicit notion of entity. The Capsule Network proposed by Geoffrey Hinton addresses the above-mentioned problems of CNN.

The basic idea of Capsule Network is to take the neurons in what we call a layer and group them into subsets and have activities of neurons in those subsets represent different properties of the same entity. The neural networks should be able to decide what the entities are and how they interact with each other. As per the paper [1], each capsule represents the presence and the instantiation parameters of a multi-dimensional entity of the type that the capsule detects. A capsule detects a particular type of object or object-part. A capsule outputs two things: the probability that an object of that type is present and the generalized pose of the object which includes position, orientation, scale, deformation, velocity, colour, etc.

The output of a capsule goes to high-level capsules in the hierarchy, which is the probability that the entity is present and the generalized pose of the entity which in vision is going to be an object or part of an object, which includes all sorts of parameters like position, orientation, hue, frequency, etc. A typical capsule receives multi-dimensional prediction vectors from capsules in the layers below and looks for tight clusters of predictions, which differentiate them from the conventional neural networks. A capsule has a high dimensional pose space and if it finds a tight cluster, it outputs a high probability that an entity of its type exists in the domain. It also outputs the centre of gravity of the cluster, which is generalized pose of the entity. This is very good for filtering out the noise because high dimensional coincidences do not happen by chance.

The current way of object recognition by the conventional neural networks [4] is by using multiple layers of learned feature detectors. The feature detectors are local, and each type of detector is replicated across space. In CNNs, the spatial domains of the feature detectors get bigger in higher layers. Feature extraction layers are interleaved with subsampling layers that pool the outputs of nearby feature detectors of the same type. Pooling layers typically attend to the most active neurons and tend to not record the position of the neuron which gives only a small amount of translational invariance at each level. In other words, the precise location of the most active feature detector is thrown away (Fig. 2.10). Pooling reduces the number of inputs to the next layer of feature extraction allowing more types of features at the next layer.

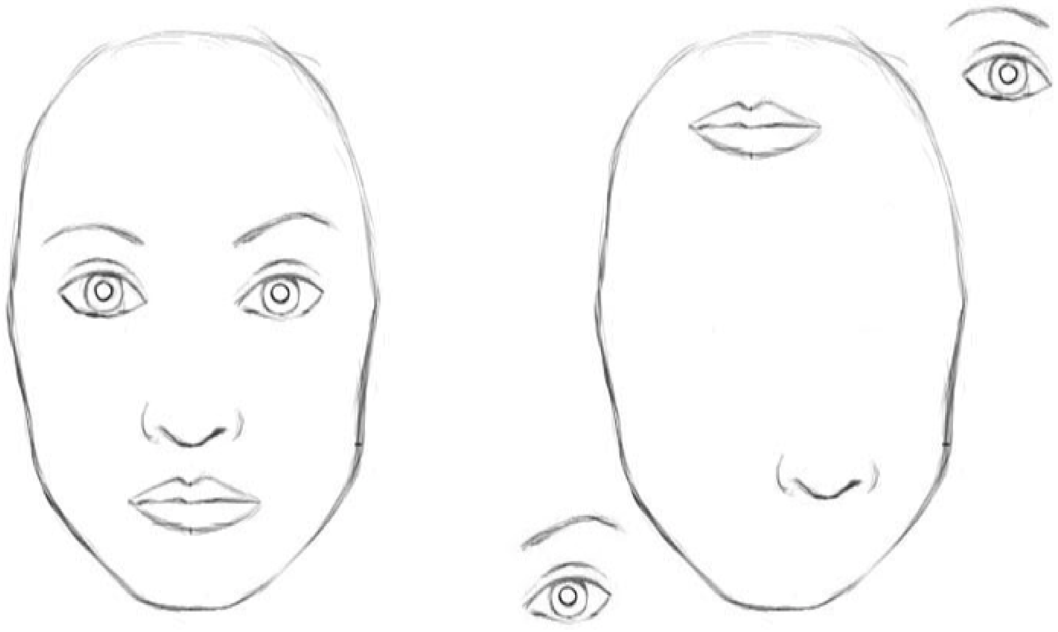


Fig. 2.10. To CNN, both pictures are the same. [30]

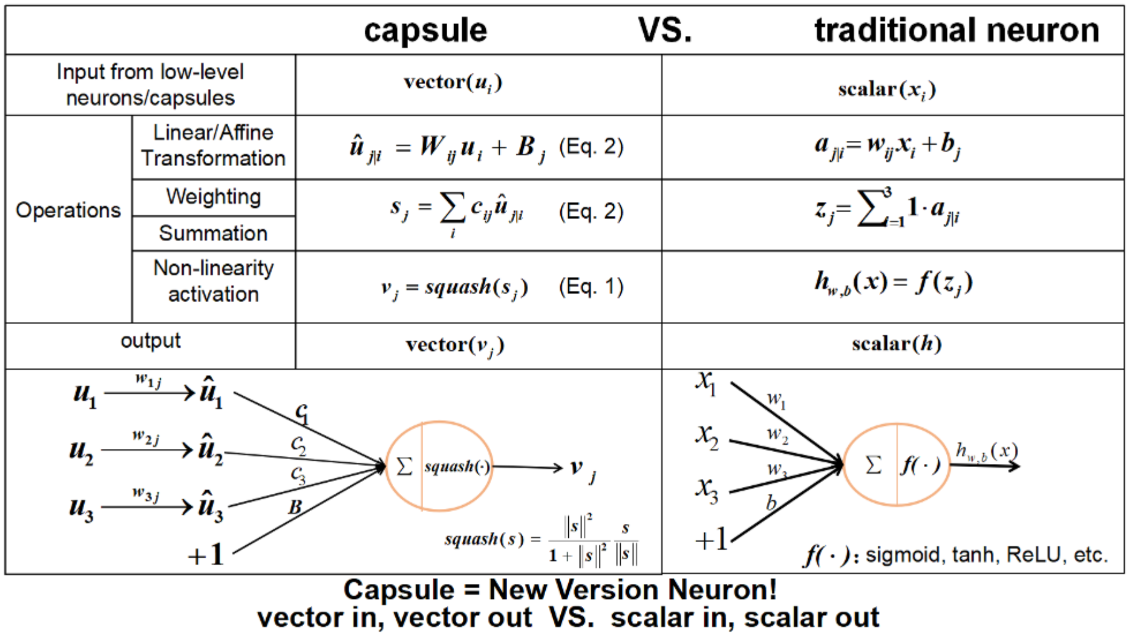
The main arguments against pooling that Hinton [7] discussed in few speeches is that, it is a bad fit to the psychology of shape perception. It does not explain why we assign intrinsic coordinate frames to objects and why they have such huge effects. Pooling solves the wrong problem: we do not want neural activities to be invariant to the viewpoint, instead we want the knowledge to be invariant to viewpoint where the same viewpoint can be applied to something with a new viewpoint i.e., disentangling rather than discarding. It does not make use of the natural linear manifold that perfectly handles the largest source of variance in images. Lastly, pooling is a primitive way to do dynamic routing.

We want equivariance, not invariance. There are two types of equivariance: if a low-level part moves to a very different positions, it will be represented by a different capsule, this is place-encoded equivariance. If a part only moves a small distance, it will be represented by the same capsule, but the pose outputs of the capsule will change, this is called rate-coded equivariance. Higher-level capsules have bigger domains so low-level place-coded equivariance gets converted into high-level rate-coded equivariance. As argued before, in each pool, the output only attends to the most active neuron in the pool. This is a very primitive way to do routing. A much better routing principle is to send the information to the capsule in the layer above that is best at dealing with it. This can be done by getting a capsule to ask for more input from lower-level capsules that vote for its cluster and less input from lower-level capsules that vote for its outliners. The total weight on the bottom-up inputs supplied by the one lower-level capsule is at most 1.

2.3.1 How a Capsule Works

The conventional neuron in a CNN can be described in three steps — scalar weighing of input scalars, sum of weighted input scalars and scalar-to-scalar nonlinearity. On the other hand, the capsule has vector forms of the above three steps and an additional step, affine transformation of the input — matrix multiplication of input vectors, scalar weighing of input vectors, sum of weighted input vectors and vector-to-vector nonlinearity.

Comparing the capsules with traditional artificial neuron:



		capsule	VS. traditional neuron
Input from low-level neurons/capsules		vector(u_i)	scalar(x_i)
Operations	Linear/Affine Transformation	$\hat{u}_{j\mid i} = W_{ij} u_i + B_j$ (Eq. 2)	$a_{j\mid i} = w_{ij} x_i + b_j$
	Weighting	$s_j = \sum_i c_{ij} \hat{u}_{j\mid i}$ (Eq. 2)	$z_j = \sum_{i=1}^{3} 1 \cdot a_{j\mid i}$
	Summation		
	Non-linearity activation	$v_j = squash(s_j)$ (Eq. 1)	$h_{w,b}(x) = f(z_j)$
output		vector(v_j)	scalar(h)

$u_1 \xrightarrow{w_{1j}} \hat{u}_1$
$u_2 \xrightarrow{w_{2j}} \hat{u}_2$
$u_3 \xrightarrow{w_{3j}} \hat{u}_3$
$+1$
c_1
c_2
c_3
B
Σ $squash(\cdot)$
v_j
$squash(s) = \frac{\|s\|^2}{1+\|s\|^2}\frac{s}{\|s\|}$
x_1
x_2
x_3
$+1$
w_1
w_2
w_3
b
Σ $f(\cdot)$
$h_{w,b}(x)$
$f(\cdot)$: sigmoid, tanh, ReLU, etc.

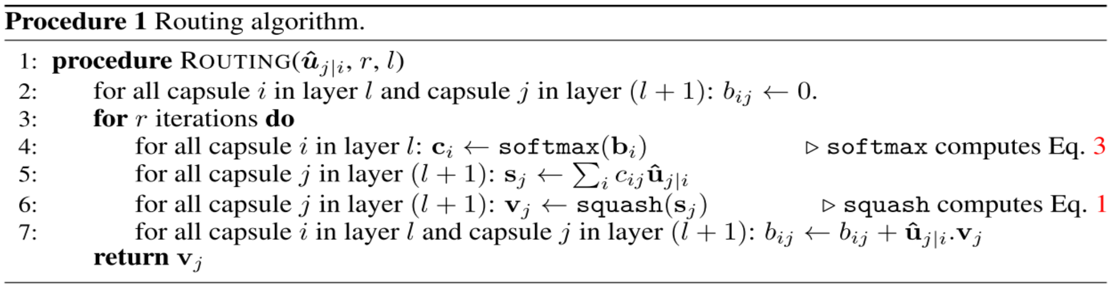
Capsule = New Version Neuron!
vector in, vector out VS. scalar in, scalar out

Fig. 2.11. Capsule vs Traditional Neuron [15]

A capsule i in a lower-level layer determines how to send its output vector to higher-level capsules j. It makes this choice by converting the scalar weight c_ij that will multiply its output vector and then be treated as input to a higher-level capsule. c_ij represents the weight that multiplies output vector from lower-level capsule i and goes as input to a higher-level capsule. [1] Each weight is a non-negative scalar and for each lower-level capsule i, the sum of all weights equals to 1. For each lower-level capsule i, the number of weights equals to the number of higher-level capsules. These weights are determined by the iterative dynamic routing algorithm. Each lower-level capsule i, defines its weights c_ij as the probability distribution function of its output fitting with each higher-level capsule j.

2.3.2 Dynamic Routing Between Capsules

The main idea behind routing algorithm is that the lower level capsules decide which higher-level capsule each entity belongs to. This is the essence of the dynamic routing algorithm.



Procedure 1 Routing algorithm.

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

Fig. 2.12. Dynamic Routing Algorithm [1]

The algorithm (Fig. 2.12) takes all capsules in a lower level l and their outputs u_hat, in addition to the quantity of routing iterations r. the very last line tells that the algorithm will produce the output of higher-level capsule v_j. Basically, this set of rules tell us the way to calculate forward pass of the network. If the activity vector has close similarity with the prediction vector (uij), we conclude that both capsules are highly related. The similarity score (bij) considers the likeliness as well the feature properties. Also, it remains low if the activation of input capsule is low since the length of prediction vector is proportional. The coupling coefficients (cij) is computed as the softmax of bij, to make it more accurate.

The dynamic routing does not completely replace backpropagation. In fact, the transformation matrix is trained with backpropagation using a cost function. We re-initialize the coupling coefficients to 0 for every datapoint before the routing calculation in order to calculate the output of a capsule.

2.3.3 Capsule Network Architecture

The Capsule Network has two parts: encoder (Fig. 2.13) and decoder (Fig. 2.15). Encoder includes Convolutional layer, Primary Capsule layer and the Digit Capsule layer. Whereas, decoder consists of layers of Fully Connected layers.

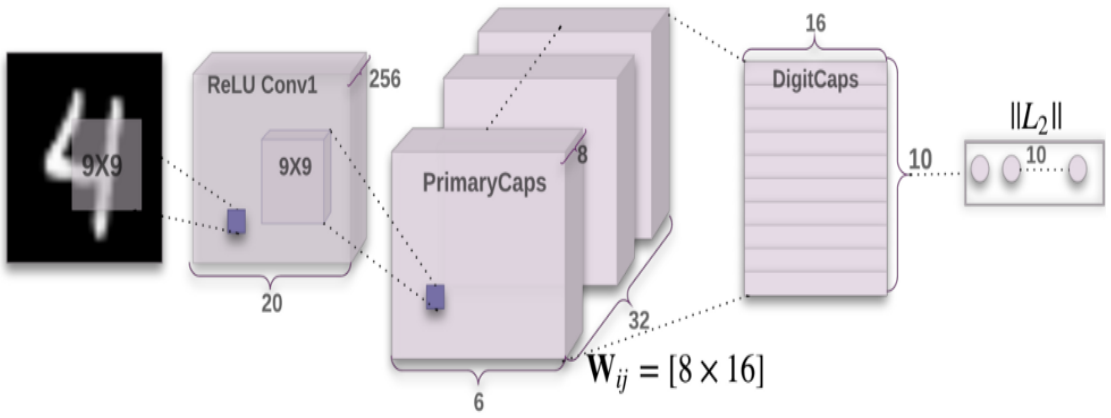


Fig. 2.13. Encoder Architecture [1]

The Convolutional layer detects the basic features in an image. The convolutional layer in Capsule Network has 256 filters of size 9x9x1 with a stride of 1 and a ReLU activation, the spatial dimension is reduced to 20x20. It is then fed into Primary Capsules, which is a modified convolutional layer supporting capsules. This layer has 32 primary capsules whose work is to produce combinations of the features detected by the convolutional layer. It generates 8D vector instead of a scalar. Primary Capsule layer is followed by DigitCaps layer, which applies transformation matrix to convert the 8D capsule to 16D capsule. The Dynamic routing takes place in this layer.

Squash function (Fig. 2.14) is applied in between layers to keep the length of the vector in between 0 and 1 similar to Sigmoid function. The squash function is given by the equation [1]:

where, s_j = weighted sum of weights

v_j = final output after applying squashing

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

Fig. 2.14. Squash Function

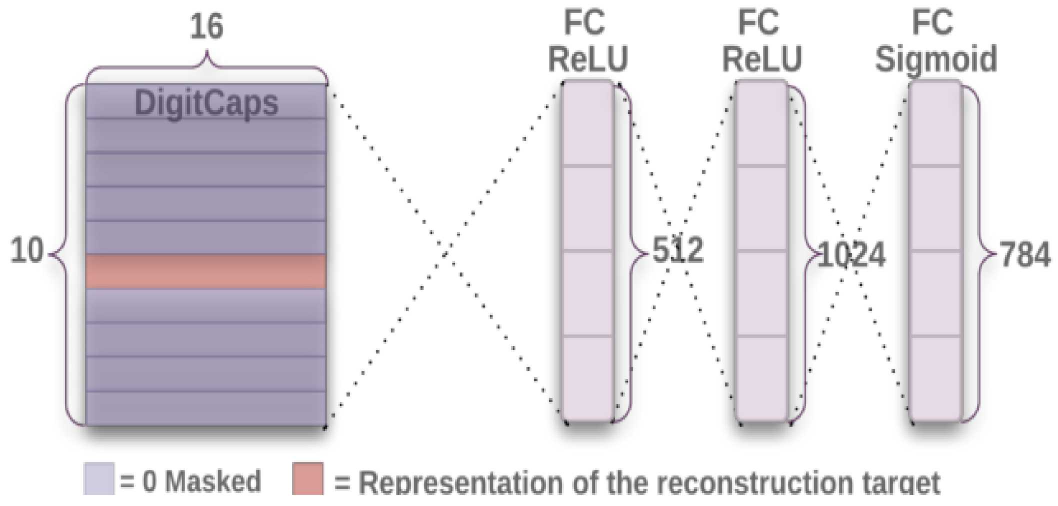


Fig. 2.15. Decoder Architecture [1]

Decoder takes the 16D vector input and learns the capsule features used for reconstructing the original image. All activity vectors are masked except the correct one which is then used to reconstruct the input image.

2.4 YOLOv3

You only look once, or YOLO [10], is one of the faster algorithms for object detection in both static images and live videos. YOLO uses a training set which consists of images and their corresponding bounding boxes of the target images. The newer YOLOv3 architecture boasts about detecting images at three different scales as shown in Fig. 2.16

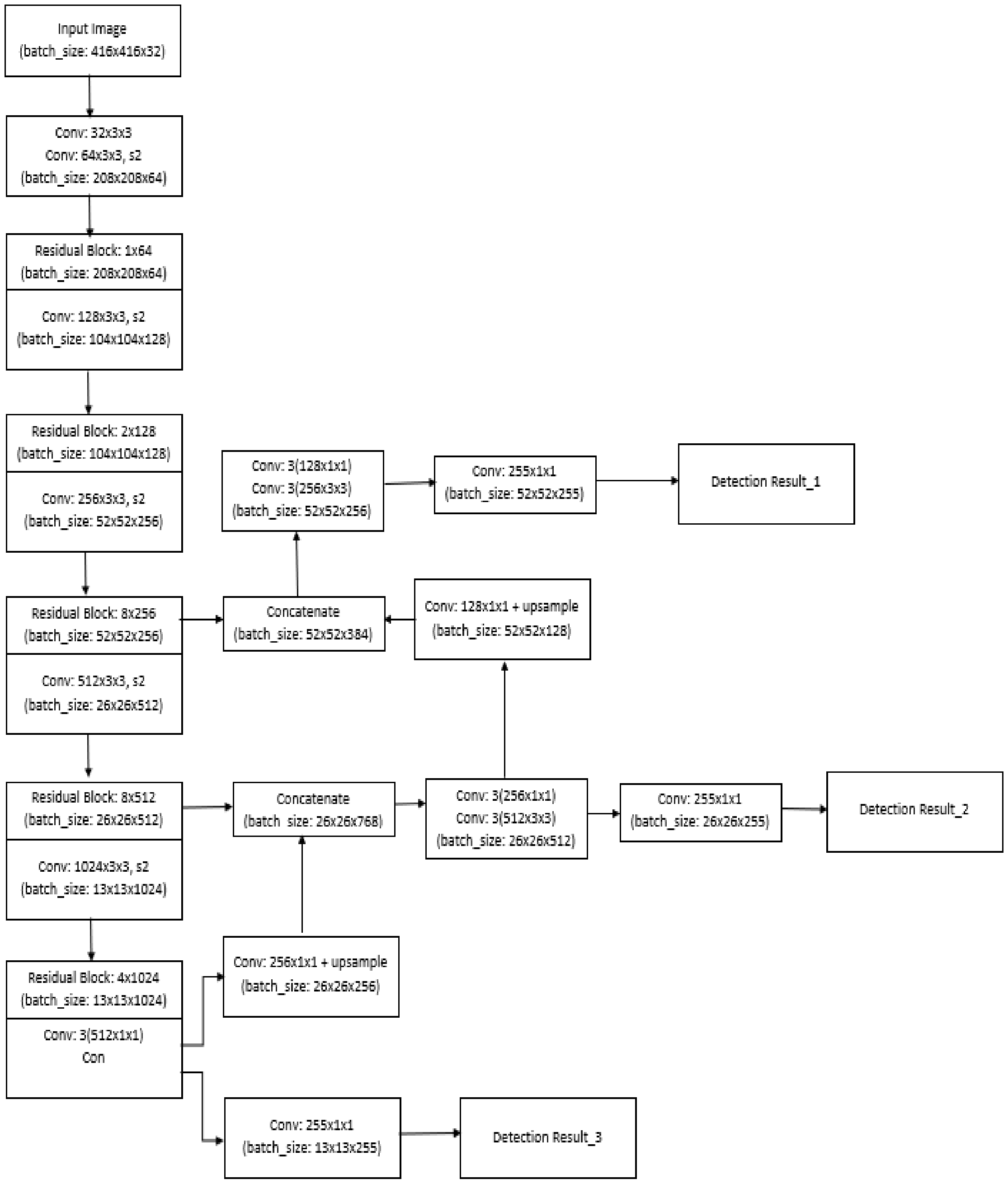


Fig. 2.16. YOLOv3 Network Architecture

Fig. 2.16 shows the architecture of YOLOv3. It is a feature-based learning network that implements 75 convolutional layers. The architecture does not have fully connected layers, any of the pooling layers or the softmax function. This makes it possible for the structure to detect images at various scales. It uses a convolutional layer with stride 2 to first downsample the feature map.

As the network goes deeper, the size of the image keeps decreasing and hence, any feature might disappear at some point in a deep network. The features may not be relevant at different depth. That is, with increase in depth, the features change from low level to high level features. Hence, the features need to be predicted at a certain depth, making feature detection at various scales even more important to maintain a good accuracy.

The shape of detection kernel is [10]:

$$1 \text{ x } 1 \text{ (B x } (5 + \text{C))}$$

Where, B = number of bounding boxes

C = number of classes

5 = stands for 4 bounding boxes and one object confidence

The feature map generated by this kernel has the height and width same as the previous feature map. The stride of each layer is defined by the ratio by which the input is downsampled. The network is given an input to predict tensors at three scales. The three scales are designed to detect objects with various sizes. The predictions precisely happen at 82nd, 94th and 106th layers. The detection at the 82nd layer gives a feature map of 13 x13 x 255.

The feature map is then subjected to few convolutions before being upsampled by twice the dimensions. This feature map is then concatenated depth-wise and the combined feature maps is again subjected to convolution to merge with the earlier layers. A similar procedure is followed to get the detected result at the 94th and 106th layers yielding feature maps of sizes 26 x 26 x 255 and 52 x 52 x 255 respectively.

3. METHODOLOGY

As discussed in the previous chapters, simple networks have performed remarkably well in classifying small datasets as MNIST. Multiple layers give the networks a compelling advantage in learning to solve complex problems. Adding depth to CNN in different combinations can improve the performance. However, increasing the depth has led to vanishing gradient problem which was addressed by ResNet [3] by adding skip connections. Adding these skip connections has reduced the number of parameters compared to the conventional CNN. Another advantage of having these skip connections is, it allows a better gradient flow through the network. Though CNN is good at detecting features for Image recognition, Image classification, Object detection and many other CV related applications, it is less effective at exploring the spatial relationships among features such as perceptive, size, colour, orientation, etc. Conceptually, a CNN model uses many layers and neurons in it to capture the different feature variants. On the other hand, Capsule network shares the same capsule across the network to detect different variants. This is the main difference between using invariance and equivariance. Pooling just concentrates on the presence the entity and ignores the position in order to decrease the number of parameters.

To overcome the drawback of ignoring the position of the entity, Sabor et. al [1] has proposed the Capsule Network which stores the information of entity at vector level rather than scalar level using dynamic routing (or routing-by-agreement). However powerful Capsule Network be, there is always a room for improvement when it comes to Neural Networks. Residual Network [3] on the other hand, has the capability to decrease the number of parameters in a complex network by adding skip connections. We bring together the concepts of skip connections from Residual Network to decrease the complexity of Capsule Network. We try decreasing the complexity by proposing two frameworks and discussing them in the following sections.

3.1 Proposed Residual Capsule Network

The first architecture we propose is the Residual Capsule Network. The first convolutional layer in Capsule Network detects the basic features in the input 2D image. But, for complex datasets this might not be enough to process further in the capsules. Hence, we try to increase the depth by adding more convolutional layers. As discussed in chapters above, simply stacking the layers may not lead to any improvement. This is where Residual Network comes to rescue. In the proposed Residual Capsule Network architecture, we modified the convolutional layers based on the skip connections. This way, there would be a better gradient flow compared to just stacking the convolutional layers.

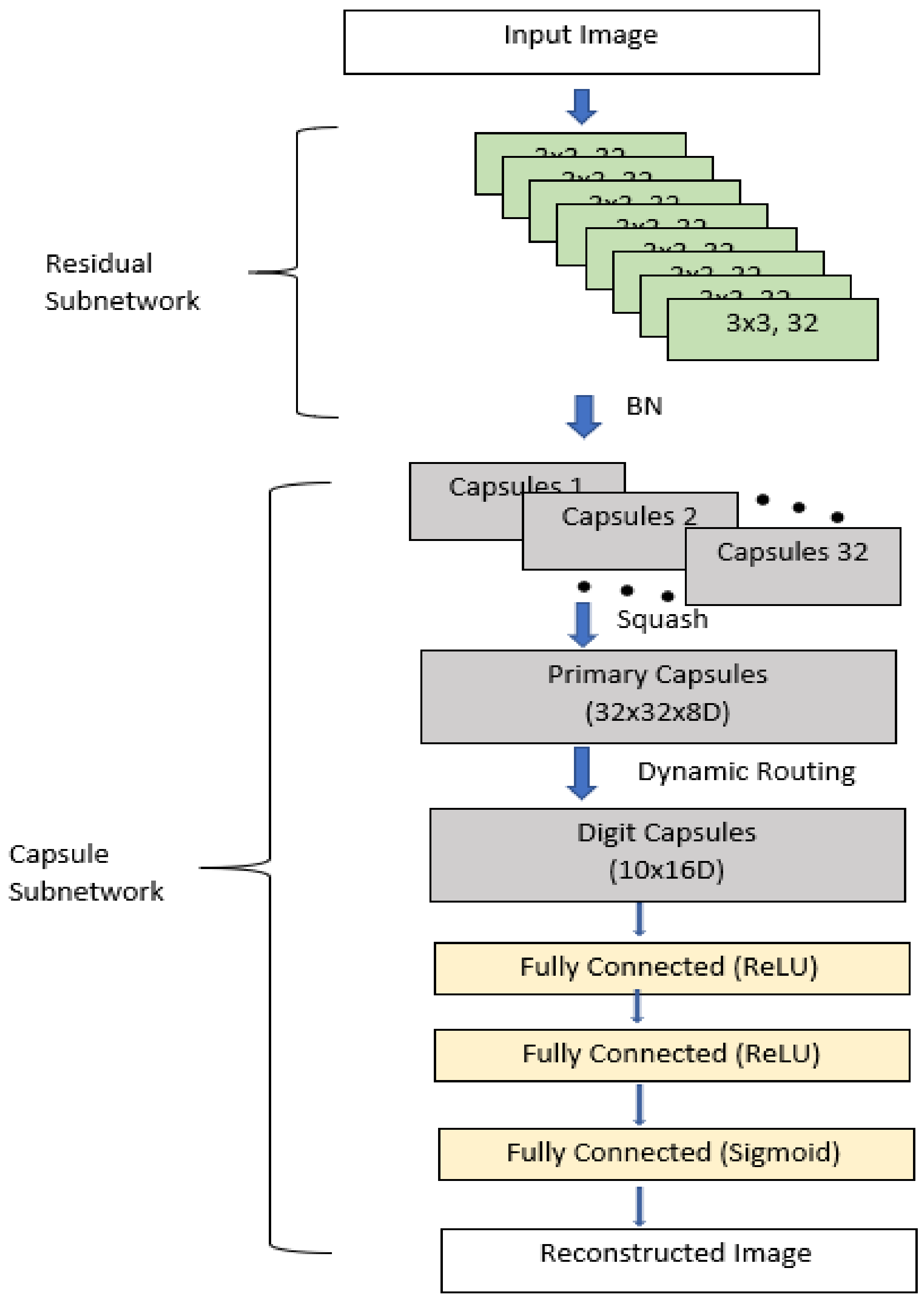


Fig. 3.1. Proposed Residual Capsule Network

The convolutional layers were modified to form a deeper architecture. An eight-layered residual convolutional subnetwork based on skip connections has replaced the first convolutional layer in the Capsule Network. Every layer is concatenated to the next layer in feed-forward manner, adding up to the final convolutional layer. Fig. 3.1 shows the detailed architecture of the proposed Residual Capsule Network for small dataset.

The input image goes into 8 layers of convolutions and each of the convolution layer generates 32 feature maps followed by concatenation of the feature maps from the previous layers adding up to 257 feature maps including the input image. These variety of feature maps act as input to the capsule layer where a convolution of 9x9 with a stride 2 is applied followed by the ReLU activation function. The convolutional layer in the baseline Capsule Network has 256 kernels with size 9x9 with stride 1. We have changed the stride from 1 to 2 and the number of kernels from 256 to 257 which in fact showed better results. The capsules attained here act as Primary capsules in the next layer of Capsule Network. The primary capsules are created by combining all the feature maps obtained from different levels of complexity.

The primary capsule layer has 32 primary capsules which take the basic features detected by the convolutional layers above and produce combinations of the features. The main intention behind Sabor et. al [1] works is to use equivariance instead of invariance by avoiding the pooling layers: max pooling and average pooling. For the same reason, we did not have any pooling layers included in the proposed architecture. The primary capsule layer is followed by the Digit Capsule layer where each of the 8D input vector gets a weight matrix and the 8D input space is converted to 16D capsule output space. The matrices for each capsule and the coefficients are used in the dynamic routing. This is repeated and a 16D output vector is generated for each of the 10 classes which further generates a one-hot 10D output vector like the baseline CapsNet.

The loss function by the baseline CapsNet [1] is given by:

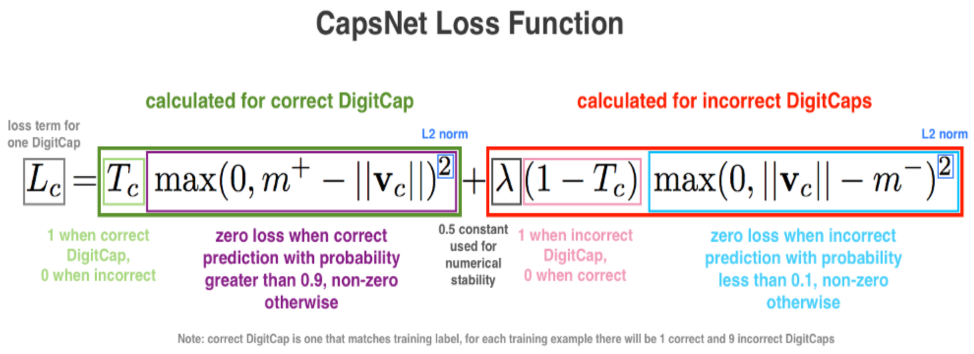

CapsNet Loss Function
calculated for correct DigitCap
calculated for incorrect DigitCaps
loss term for one DigitCap
L2 norm
L2 norm
$L_c = T_c \max(0, m^+ - ||\mathbf{v}_c||)^2 + \lambda(1 - T_c)\max(0, ||\mathbf{v}_c|| - m^-)^2$
1 when correct DigitCap, 0 when incorrect
zero loss when correct prediction with probability greater than 0.9, non-zero otherwise
0.5 constant used for numerical stability
1 when incorrect DigitCap, 0 when correct
zero loss when incorrect prediction with probability less than 0.1, non-zero otherwise
Note: correct DigitCap is one that matches training label, for each training example there will be 1 correct and 9 incorrect DigitCaps

Fig. 3.2. Loss Function of Capsule Network

The output of the Digit Capsule layer is a 16D vector. During training, each of the training example will have one loss value calculated for 10 vectors and all the values will be added to get the final loss (Fig. 3.2) for each. In other words, each label will have a ten-dimensional vector with 9 zeros for the incorrect value calculated and 1 one for the correct position calculated. For every correct label, 1 is assigned and for every incorrect label, 0 is assigned. The loss will be zero for the right prediction of the DigiCap with a probability of value greater than 0.9 and for probability any less than 0.9, the value assigned is non-zero.

The decoder learns to decode the correct 16D vector received from DigitCap to an image. It uses the vector during training and masks all other outputs except the correct DigitCap with the help of loss function and reconstructs the input image. The fully connected layer gets the weights of each neuron from the lower level. The neurons are then directed to the other fully two fully connected layers. During training, all the activity vectors are masked except the right one. The input image is reconstructed using this activity vector. The output from the digit capsule layer is fed into decoder with 3 fully connected layers that is patterned into pixel intensities for the (input) image.

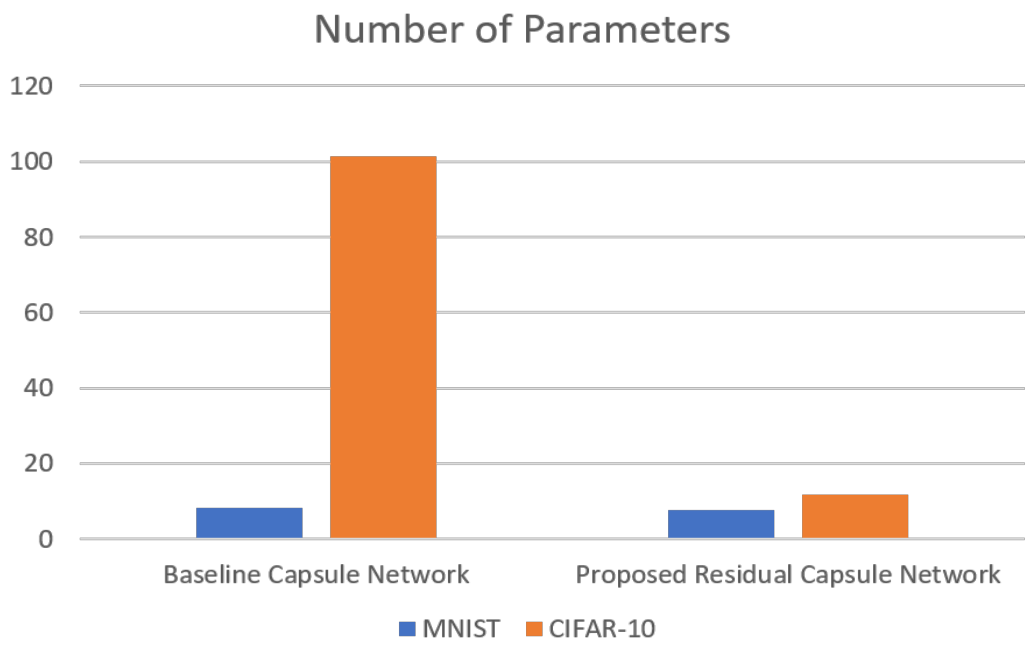


Fig. 3.3. Parameter Comparison of Models

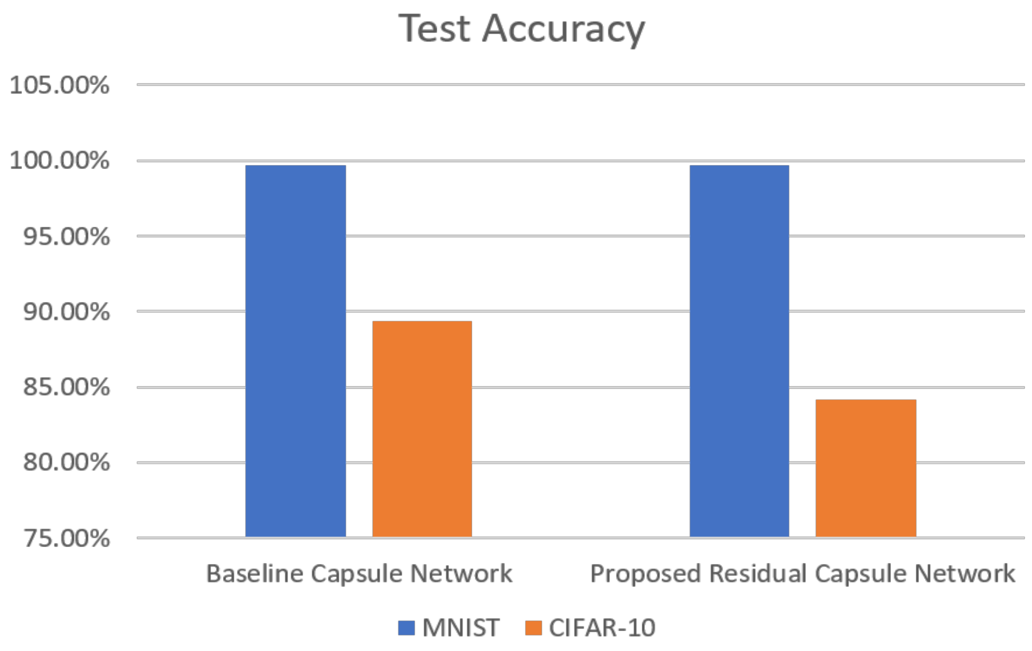


Fig. 3.4. Accuracy Comparison of Models

Residual Capsule Network performed well with both the accuracy and the number of parameters on small dataset of images such as MNIST. But, when it came to complex datasets like CIFAR-10, it did not outmatch the performance of seven ensemble Capsule Network [3]. Since the CIFAR-10 images are more complex, a single layer of primary capsules is not enough to calculate all the features and relation between part-whole of the image. This problem has been tackled by 3-Level Residual Capsule Network.

3.2 Proposed 3-Level Residual Capsule Network

A simple layer of Residual Network [2] did not solve the problem for complex dataset like CIFAR-10. This maybe because of the simple primary capsules which may not be enough to compute all the features of the image and the relation between part-whole of each entity. To overcome this, we propose another architecture: 3-Level Residual Capsule Network for complex datasets. This architecture has been inspired by the paper published on YOLOv3 by Joseph Redmon et. al [10]. The primary capsules are created to carry information at 3 different scales of the image. Each level of the 3-Level Residual Capsule Network is similar to Residual Capsule Network.

The YOLOv3 [10] architecture has residual skip connections. The most prominent feature of this network is that it makes detections at three different scales. This network is capable of detecting objects of different sizes. As the network goes deeper, the feature map keeps getting smaller making it difficult to detect. In order to detect all the entities of the image, the feature maps need to be detected at different scales. Since YOLOv3 has similar structure to ResNets it is a better way to implement the idea into Residual Capsule Network and improvise it to grasp features at different scales.

Inspired by the idea of detecting images at three different scales, we have implemented our proposed Residual Capsule Network to carry out information of various scales of images, hence diversifying the capsules. If we cone one level of the proposed

3-Level Residual Capsule Network, the image activations will be same as the earlier proposed Residual Capsule Network and baseline Capsule Network.

The ensemble baseline Capsule Network model by Sabor et. al [3] gives an accuracy of 89.4%. Though the ensemble model is achieving a high accuracy, it leads to a large increase in the number of parameters, which will be discussed further in the next chapter. Our architecture aims at reducing the number of parameters. Fig. 3.3 shows the queuing of the multiple Residual Capsule Network into layers forming 3-Level Residual Capsule Network. Each of the single layered Residual CapsNet comprises of 12 capsules. The stride at each level is defined to downsample the input image before feeding into the next level of the network.

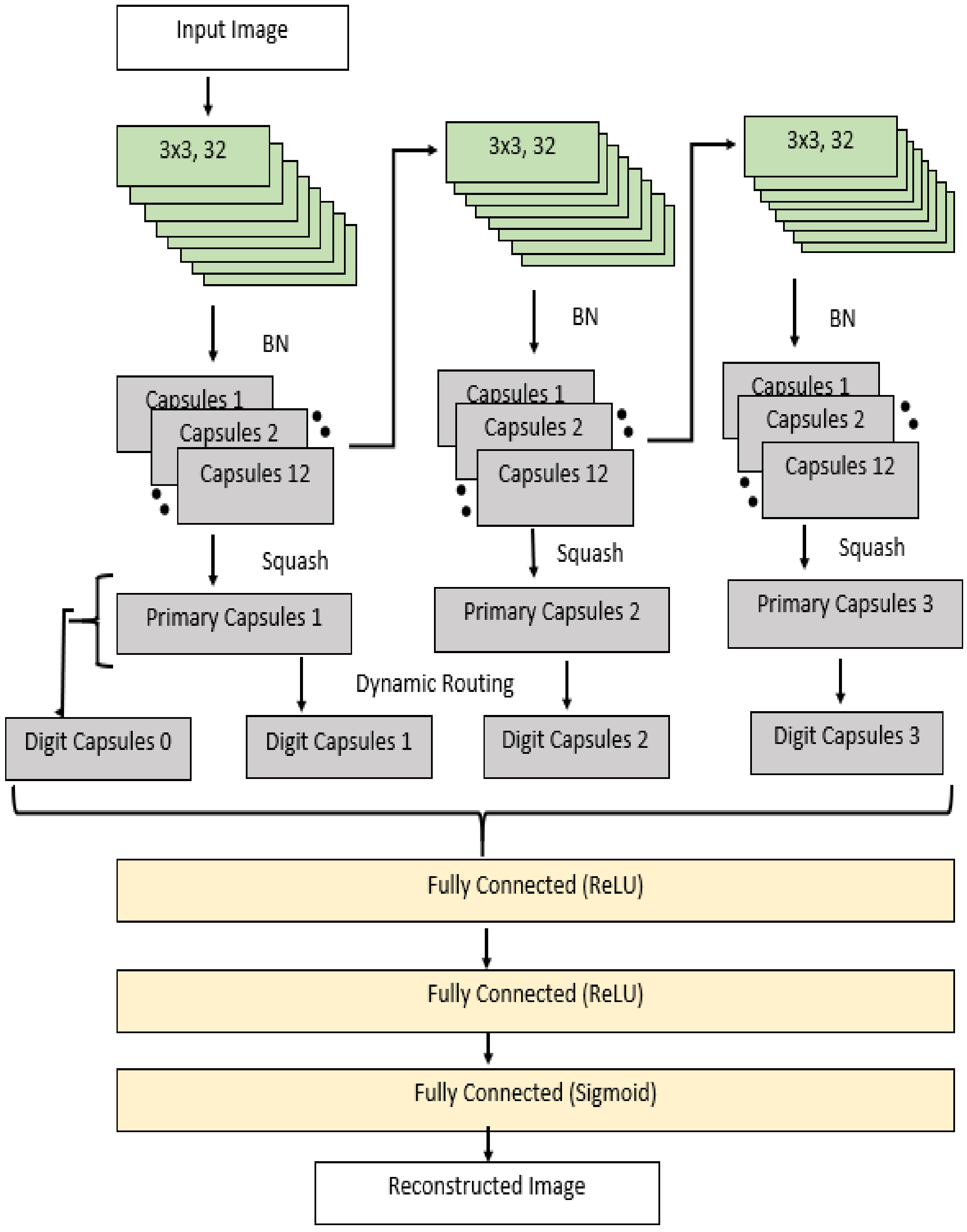


Fig. 3.5. Proposed 3-Level Residual Capsule Network

Fig. 3.5 shows the architecture of proposed 3-Level Residual Capsule Network. This network has been particularly designed to train complex datasets like CIFAR-10. It is a hierarchical model similar to the structure of a pyramid as shown in Fig. 3.6 [17], where each level represent the proposed Residual Capsule Network model and it is used an input to the next level of Residual Capsule Network which in turn generates the feed to the next layer. There are twelve capsules in each of the Residual Capsule layers and each of the layers have convolution of size 9x9 with stride 2. The convolution reduces the size of the image which is fed into the next level. This is exactly how the brain functions where the information is separated into channels i.e., separate channels for colour, frequency, etc.

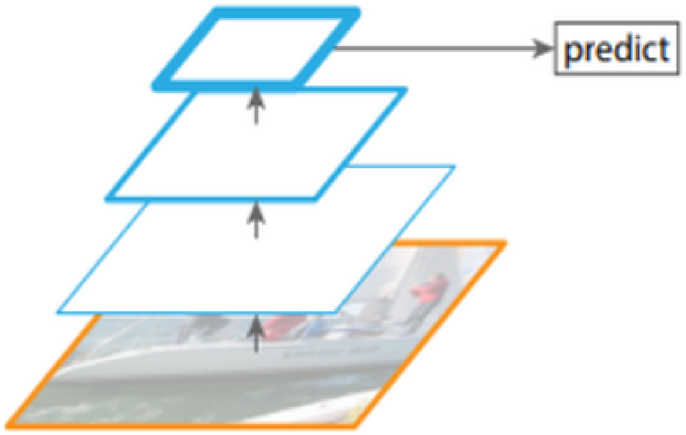


Fig. 3.6. Pyramid Structure of Multi-scale Learning

In addition to the three DigitCaps layers, we created one more DigitCaps output layer by routing the concatenation of the three Primary capsule layers [8]. Adding the extra layer would help the model learn better from the merged features assembled from various levels. While running the entire model, the four DigitCaps layers are concatenated to avoid any imbalance learning by the model. DigitCaps generated from various Primary capsule layers played a major role in affecting the reconstruction, which is a combined effect of the different layers of capsules.

The 3-Level Residual Capsule Network has been tested on CIFAR-10 and has proven to overcome the problem from Residual Capsule Network. The test results will be discussed in detail in the next chapter.

4. DATASETS

The datasets are mainly divided into three categories — Audio/ Speech Processing, Image Processing and Natural Language Processing. The usage of the datasets depends on the structure of the problem and uniqueness of the case. It may be difficult to find a specific dataset for variety of machine learning problems. There are few pointers which need to be noted while looking for a right dataset: the dataset chosen should be of high-quality but not messy, the dataset should have fewer rows or columns for the ease of working with them, cleaning the dataset might take abundant time hence, a clean dataset is a must and lastly, the dataset should meet the requirements of the problem that needs to be solved. For this thesis work, the two datasets chosen are MNIST and CIFAR-10.

4.1 MNIST

The MNIST database (Modified National Institute of Standards and Technology database) [12] is generally used for training various image processing applications [19] [20]. It is used to benchmark classification algorithms. Every time a new machine learning technique emerges, MNIST has remained to be a consistent source for testing. The MNIST database is a set of handwritten digits. It has a training set of 60,000 images and a testing dataset of 10,000 samples. MNIST is a subset of a larger set NIST [18]. The set contains digits 0 through 9 size-normalized and centered in an image of fixed size. It is a basic database for learning techniques and pattern recognition problems. The database is easier to use with less preprocessing and formatting.

4.2 CIFAR-10

The CIFAR-10 dataset (Canadian Institute for Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms in research. It is a subset of 80 million tiny images dataset [23]. The CIFAR-10 dataset [21] consists of 60,000 colour images of size 32 x 32 in 10 classes with 6,000 images per class. There are 50,000 training images and 10,000 testing images. The 10 different classes are [22]: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks. All the images in the classes are mutually exclusive i.e., there is no overlap between any two different kinds of objects.

5. EVALUATION

We have evaluated our proposed models on the two basic datasets: MNIST, CIFAR-10 and compared the results with the Baseline Capsule Network by Sabor et. al [3] and DCNET and DCNET++ by Phaye [8]. We ran all our evaluations on Aorus GeForce RTX 2080Ti GPU. We have run our models for 50, 120 epochs. Our implementation is in Keras and we use Adam optimizer with parameters set to 0.001 as initial learning rate and a decay rate of 0.9. We have used publicly available code [10] and made changes to suit our requirements for the proposed network. We have kept most of the parameters of the proposed Residual Capsule Network similar to the traditional Capsule Network, for fair comparisons.

5.1 MNIST Handwritten Digits Database

The MNIST database of handwritten digits [11], has a training set of 60,000 images and the testing set has a set of 10,000 images of each 28x28 in size.

Table 5.1.
Performance of various Capsule Network models on MNIST dataset

Model	***Parameters***	***Test Accuracy****
Baseline Capsule Network	8.2M	99.67%
DCNet	11.8M	99.75%
Proposed Residual Capsule Network	7.7M	99.66%
DC++Net	10.5M	99.69%
Proposed 3-Level Residual Capsule Network	8.5M	99.59%

$*Models are run for 50 Epochs each.$

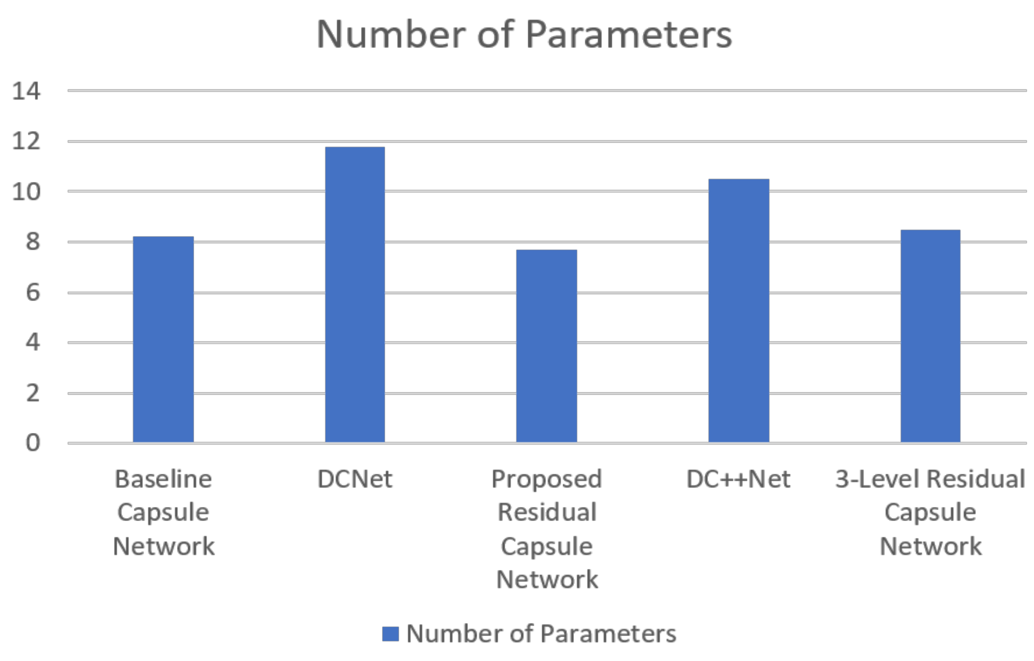


Fig. 5.1. Parameters comparison of various models on MNIST Dataset

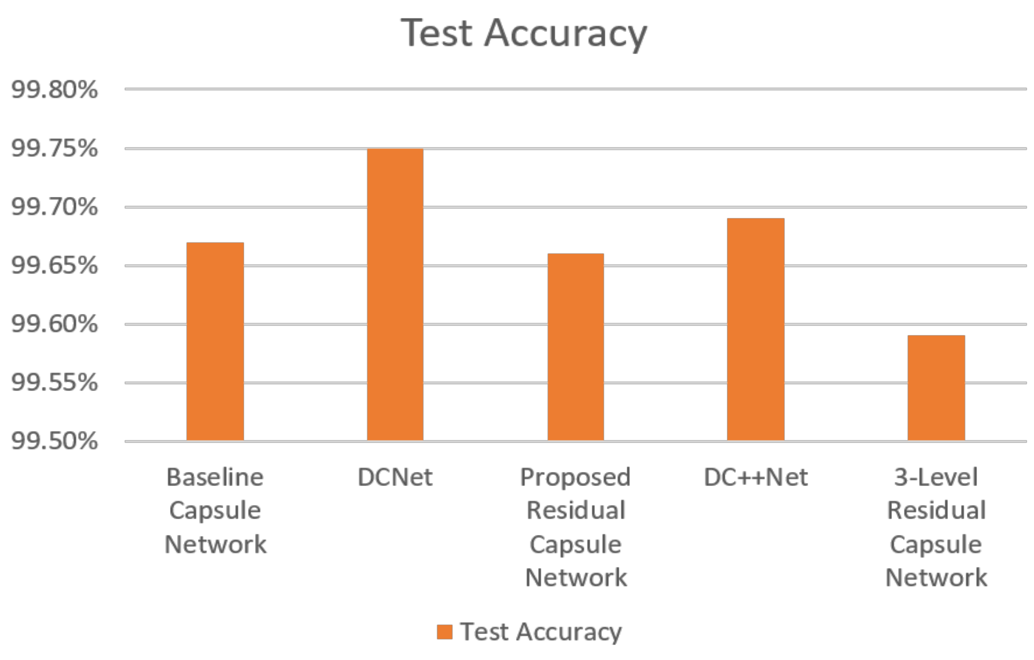


Fig. 5.2. Accuracy comparison of various models on MNIST Dataset

The performance of various Capsule network models has been compared with the proposed models in Table 1. It can be clearly seen that the accuracy of the proposed Residual Capsule Network is 99.66%. Since our aim was to reduce the size of the model by decreasing the number of parameters, our proposed model has decreased the number of parameters by 6.09% compared to the Baseline Capsule Network [3] at a cost of 0.01% reduction in accuracy. When compared with architecture built with Dense Convolutional Network [8], the size of proposed model is 34.74% less and a decrease of 0.09% in accuracy has been observed.

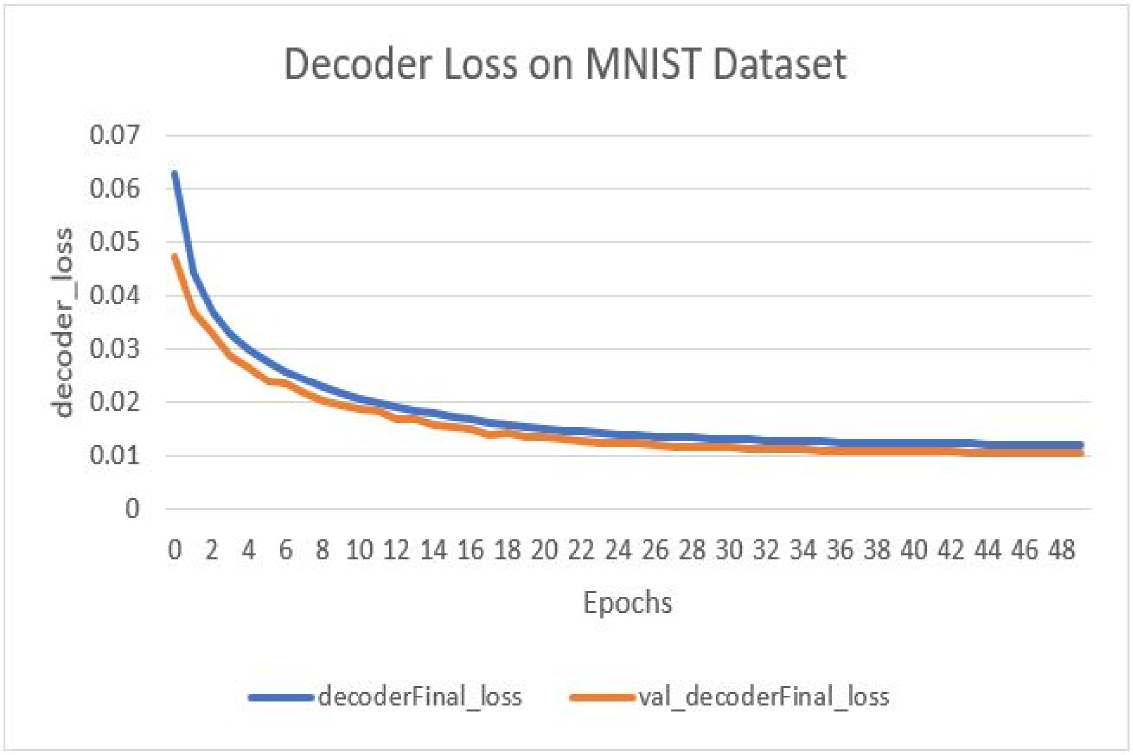


Fig. 5.3. Decoder Loss on Residual Capsule Network (MNIST Dataset)

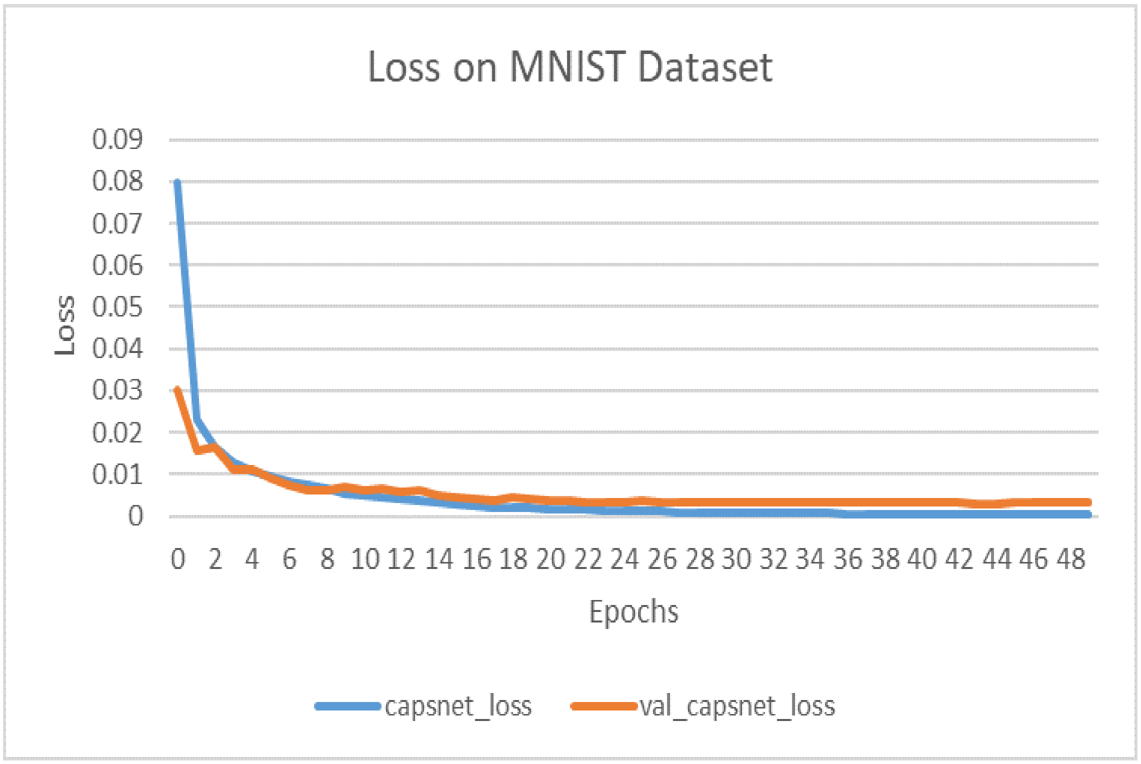


Fig. 5.4. Final Loss on Residual Capsule Network (MNIST Dataset)

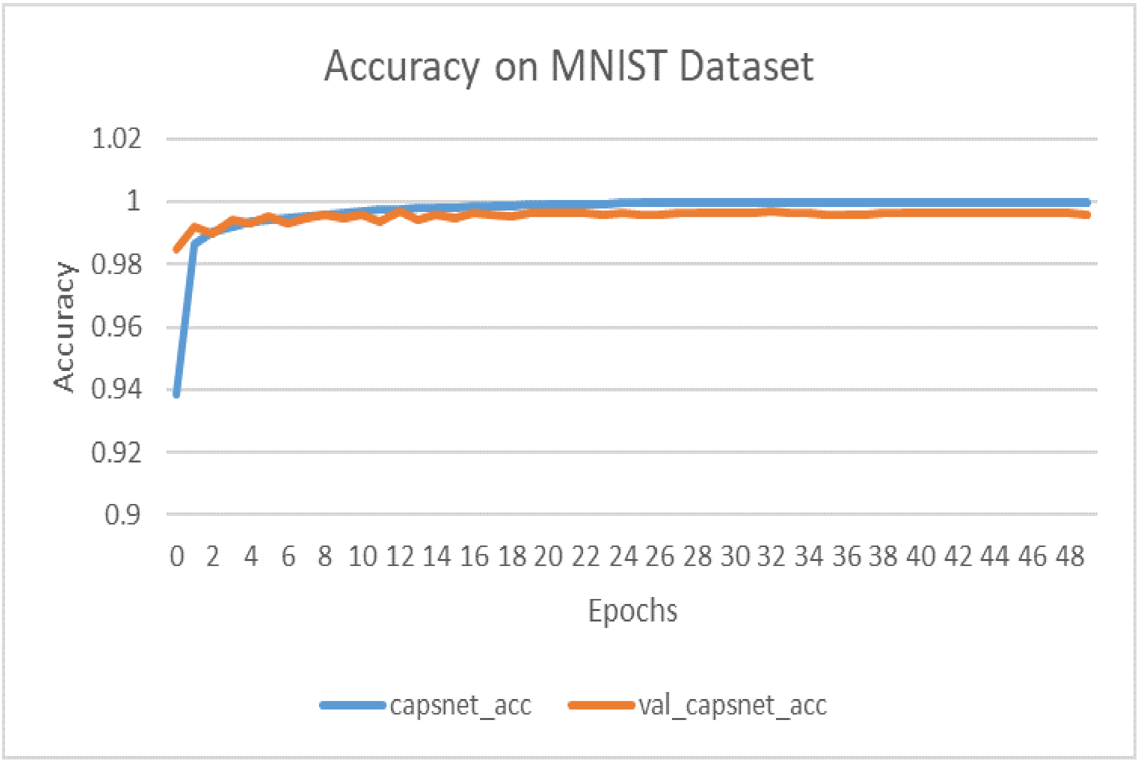


Fig. 5.5. Test Accuracy on Residual Capsule Network (MNIST Dataset)

Though the proposed 3-Level Residual Capsule Network did well compare to DC-Net as well as DC++Net in terms of reducing the number of parameters, it failed to perform at par with the Baseline Capsule Network with respect to number of parameters and accuracy. Residual Capsule Network did a fine work in decreasing the number of parameters when trained on MNIST dataset.

5.2 CIFAR-10 Dataset

CIFAR-10 [21] dataset consists of 60,000 images of 32x32 size each in 10 classes, with 6000 images per class. The images are divided into 50,000 training images and 10,000 test images. We compare our proposed 3-Level Residual Capsule Network with seven ensemble Capsule Network [3] and DC++Net [8].

Table 5.2.
Performance of various Capsule Network models on CIFAR-10 dataset

Model	***Parameters***	***Test Accuracy***
Baseline Capsule Network	7x14.5M = 101.5M	89.40% (7 model ensemble)
DCNet	11.88M	82.63% (120E)
Proposed Residual Capsule Network	11.86M	84.16% (120E)
DC++Net	13.4M	89.71% (120E)
Proposed 3-Level Residual Capsule Network	10.8M	86.42% (120E)

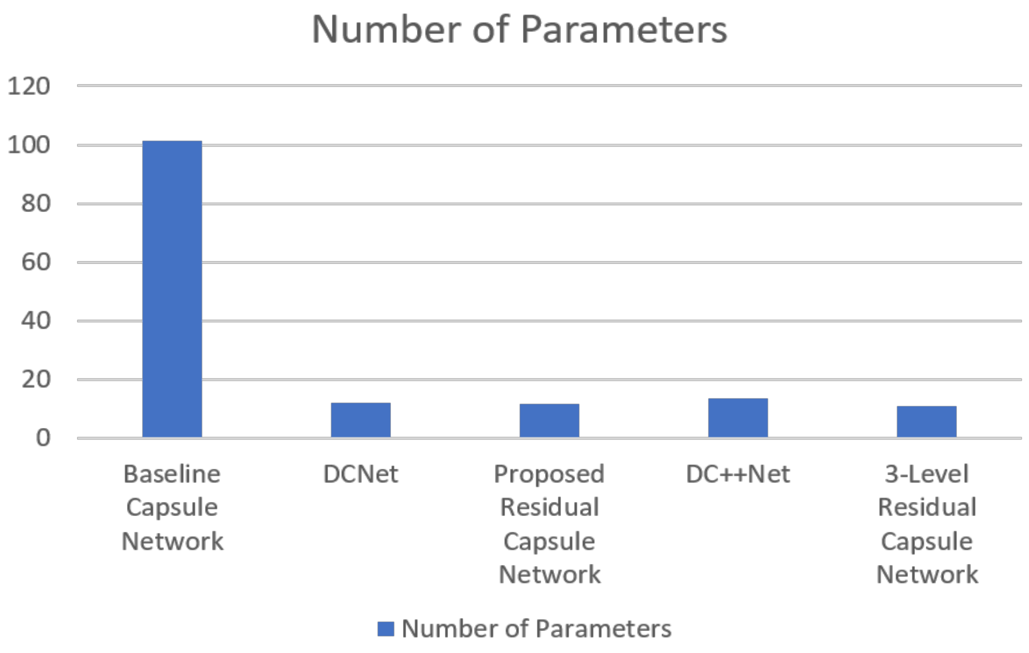


Fig. 5.6. Parameters comparison of various models on CIFAR-10 Dataset

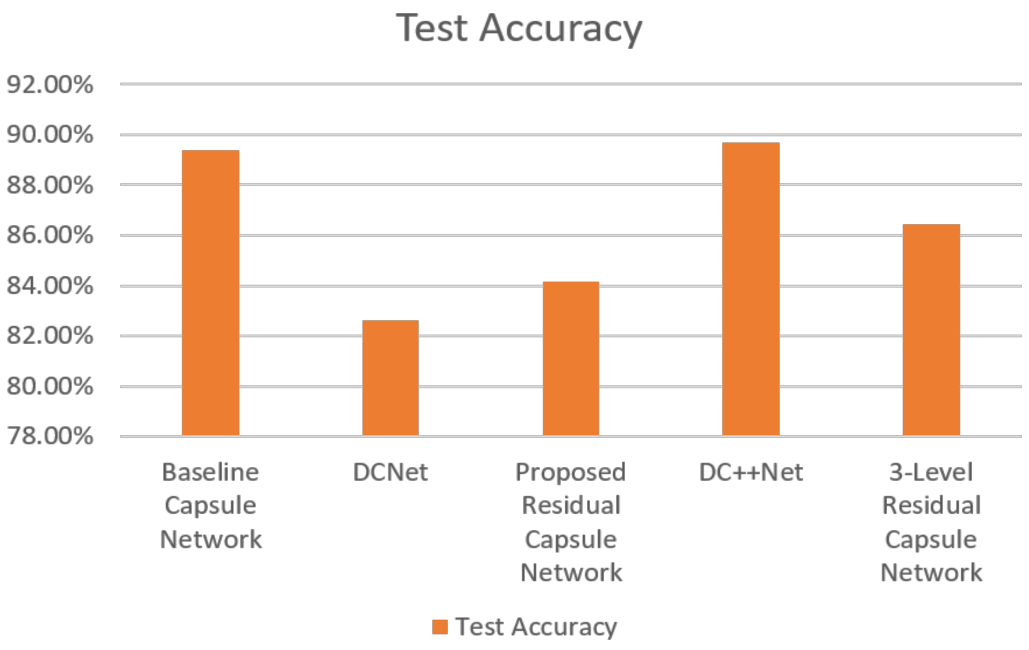


Fig. 5.7. Accuracy comparison of various models on CIFAR-10 Dataset

The proposed 3-Level Residual Capsule Network has parameters similar to Residual Capsule Network. The proposed 3-Level Residual Capsule Network when compared to the seven ensemble Capsule Network reduced the number of parameters by 89.35%, compromising on the accuracy by 2.98%. When compared against DC++Net, the size of proposed model is 19.40% less at the cost of 3.29% decrease in accuracy. The proposed 3-Level Residual Capsule Network achieved an accuracy of 86.42% with 10.8M parameters which has drastically dropped from 101.5M (7x14.5M) proposed by Sabor et. al [3].

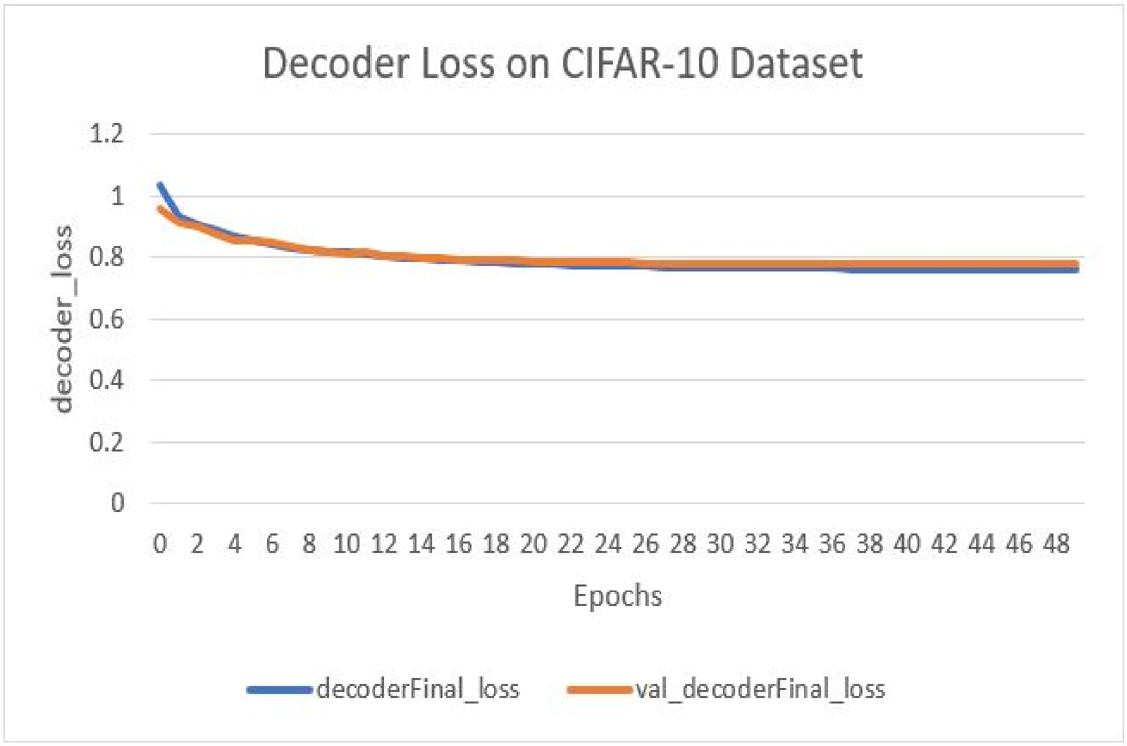


Fig. 5.8. Decoder Loss on 3-Level Residual Capsule Network (CIFAR-10 Dataset)

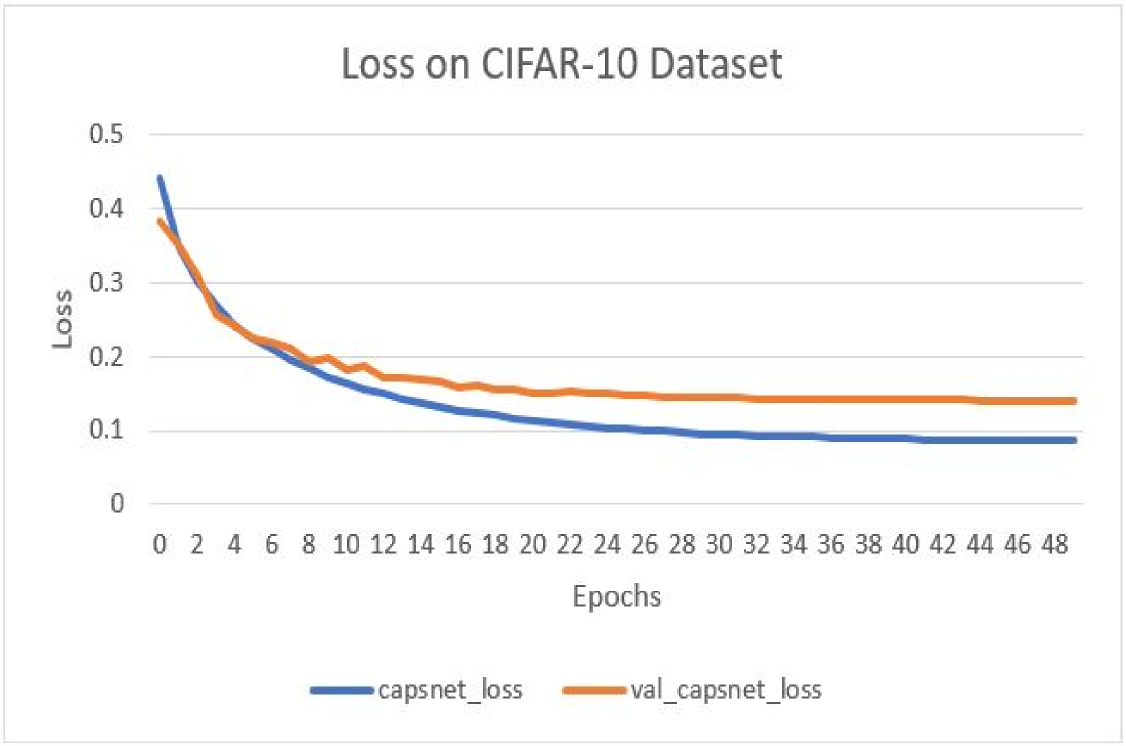


Fig. 5.9. Final Loss on 3-Level Residual Capsule Network (CIFAR-10 Dataset)

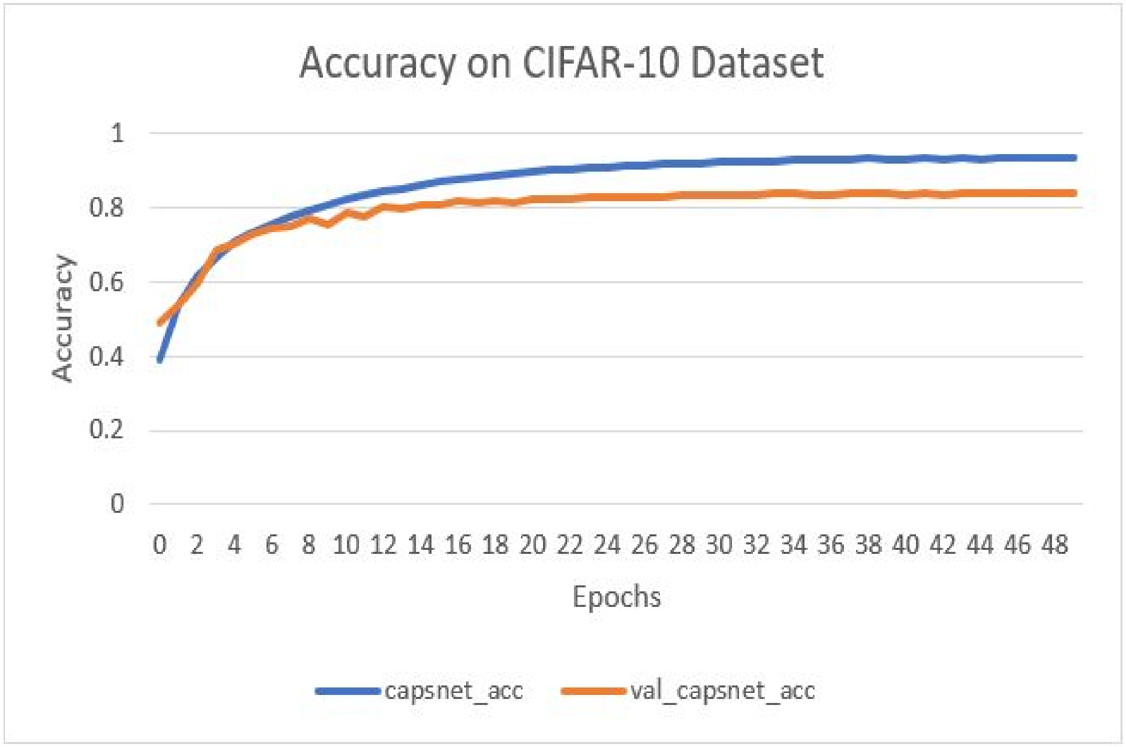


Fig. 5.10. Test Accuracy on 3-Level Residual Capsule Network (CIFAR-10 Dataset)

6. SUMMARY

Convolutional Neural Networks are fascinating and a powerful tool. They are one of the reasons why Deep learning is so popular [2]. Making a network deeper networks did improve the performance of the neural network models, but simply stacking of layers brought in a new problem of vanishing gradients. This problem has been alleviated with the introduction of a new network Residual Network. On the other hand, though CNNs are very appealing and have contributed a lot towards machine learning, they are not flawless. Firstly, backpropagation is a method to evaluate the weight in the error after each batch is preprocessed. It is good but not good enough since it needs large datasets for learning. As pointed out by Hinton et. al, pooling is a huge mistake since most of the valuable information about the relation between a part and whole is lost. Convolutional Neural Networks try to make the neutral activities invariant to small changes in viewpoint by combining the activities of the pool. But it is better to aim at equivariance where the changes in viewpoint correspond to neural activities. Sabor et. al [1] has tossed the idea of replacing the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement to achieve the state-of-the-art accuracy on the MNIST dataset.

This thesis uses the best of features of both Residual Network and Capsule Network and proposes two frameworks: Residual Capsule Network and 3-Level Residual Capsule Network to make the Capsule Network deeper yet efficient. To achieve this we decrease the complexity of the baseline Capsule Network by decreasing the number of parameters. We have replaced traditional Convolutional Layer in the Capsule Network has been replaced by Residual (block-2) Network to help decrease the model size proposed by Sabor et. al [3]. The proposed Residual Capsule Network model is potent of reducing the number of parameters by 6.09% by compromising on the accuracy by 0.01% when compared to Baseline Capsule Network and 34.74% reduction on architecture built by Dense Convolutional Network at the cost of 0.09% accuracy on MNIST dataset. But this proposed model did not do well on complex dataset CIFAR-10.

A simple layer of Residual Network [2] did not solve the problem for complex dataset like CIFAR-10. This maybe because of the simple primary capsules which may not be enough to compute all the features of the image and the relation between part-whole of each entity. 3-Level Residual Capsule Network has addressed the problem with complex dataset such as CIFAR-10 by queuing the number of levels of Residual Capsule Network. The proposed 3-Level Network outperformed seven ensemble Capsule Network in reducing the number of parameters by 89.35% and affecting the accuracy by 2.98%. On the other hand, the proposed 3-Level Network was successful in reducing the number of parameters by 19.40% when compared to DC++Net with a compromise of 3.29% in accuracy.

REFERENCES

REFERENCES

[1] Sara Sabour, Nicholas Frosst, Geoffrey E Hinton. *“Dynamic Routing between Capsules.”* In Advances in Neural Information Processing Systems, 2017.

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton. *“ImageNet Classification with Deep Convolutional Neural Network.”* In advances in Neural Information Processing Systems, 2012.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *“Deep Residual Learning for Image Recognition.”* In Proceedings of the IEEE Conference on Computer Vision and Patter Recognition. arXiv preprint arXiv: 1512.03385, 2015. [online] [https://arxiv.org/pdf/1512.03385.pdf] [Accessed on July 01, 2019].

[4] Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner. *“Gradient-Based Learning Applied to Document Recognition.”* Proceedings of the IEEE, vol. 86, no. 11, 1998, pp. 2278-2324., doi:10.1109/5.726791. [online] [https://ieeexplore.ieee.org/document/726791] [Accessed on July 01, 2019].

[5] Rupesh Kumar Srivastava, Klaus Greff, Jurgen Schmidhuber. *“Highway Networks.”* CoRR, abs/1505.00387, 2015.

[6] Geoffrey E Hinton, Sara Sabour, Nicholas Frosst. *“Matrix Capsules with EM Routing.”* 2018. [online] [https://openreview.net/pdf?id=HJWLfGWRb] [Accessed on July 01, 2019].

[7] Edgar Xi, Selina Bing, Yang Jin. *“Capsule Network Performance on Complex Data.”* arXiv preprint arXiv:1712.03480, 2017. [online] [https://arxiv.org/pdf/1712.03480 .pdf] [Accessed on July 01, 2019].

[8] Geoffrey E Hinton, Alex Krizhevsky, Sida D Wang. *“Transforming Auto-Encoders.”* In International Conference on Artificial Neural Networks. Springer, 2011.

[9] Sai Samarth R Phaye, Apoorva Sikka, Abhinav Dhall, Deepti Bathula. *“DCNET and DCNET++: Making the Capsules Learn Better.”* arXiv:1805.04001, 2018. [online] [https://arxiv.org/pdf/1805.04001.pdf] [Accessed on July 01, 2019].

[10] Joseph Redmon, Ali Farhadi. *“YOLO v3: An Incremental Improvement.”* arXiv preprint arXiv:1804.02767, 2018. [online] [https://pjreddie.com/media/files/papers/YOLOv3.pdf] [Accessed on July 01, 2019].

[11] SSRP, *Multi-level-DCNet.*[online] [https://github.com/ssrp/Multi-level-DCNet]. [Accessed on 07/01/2019].

[12] Yann LeCun, Corinna Cortes, Christopher JC Burges. *“The MNIST Database of Handwritten Digits,”* 1998. [online] [http://yann.lecun.com/exdb/mnist/] [Accessed on July 01, 2019].

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. *"Going Deeper with Convolutions."* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[14] Balazs Csanad Csaji, 2001. *"Approximation with Artificial Neural Networks"* (Mas ter's Thesis); Faculty of Sciences; Eotvos Lorand University, Hungary. [online] [http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.2647rep=rep1ty pe=pdf] [Accessed on July 01, 2019].

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *"Identity Mappings in Deep Residual Networks."* arXiv preprint arXiv:1603.05027v3, 2016. [online] [https://arxiv.org/pdf/1603.05027.pdf] [Accessed on July 01, 2019].

[16] Naturomics, *CapsNet-Tensorflow.* [online] [https://github.com/naturomics/Caps-Net-Tensorflow/] [Accessed on July 01, 2019].

[17] *"A Closer Look at YOLOv3."* [online] [https://cyberailib.com/home/a-closer-look-at-yolov3] [Accessed on July 01, 2019].

[18] Vincent K. Shen, William P Krekelberg, Daniel Siderius, Raymond D Mountain, Harold Wickes Hatch, NIST Standard Reference Simulation Website, NIST Standard Reference Database Number 173, National Institute of Standards and Technology, Gaithersburg MD, 20899, September 2017.

[19] Hyeran Byun, Seong-Whan Lee. *"Applications of Support Vector Machines for Pattern Recognition: A Survey,"* Vision Systems Design. [online] [http://citeseerx.i st.psu.edu/viewdoc/download?doi=10.1.1.723.5893rep=rep1type=pdf] [Accessed on July 01, 2019].

[20] Sachin Gangaputra. *"Handwritten Digit Database."* [online] [http://cis.jhu.edu/ sa chin/digit/digit.html] [Accessed on July 01, 2019].

[21] Tom Hope, Yehezkel S. Resheff, Itay Lieder (2017-08-09). *"Learning TensorFlow: A Guide to Building Deep Learning Systems".* "O'Reilly Media, Inc. pp. 64. ISBN 9781491978481.

[22] Plamen Angelov, Alexander Gegov, Chrisina Jayne, Qiang Shen (2016-09-06). *"Advances in Computational Intelligence Systems: Contributions Presented at the 16th UK Workshop on Computational Intelligence,"* Lancaster, UK. Springer International Publishing. pp. 441. ISBN 9783319465623. [Accessed on July 01, 2019].

[23] Alex Krizhevsky, *"Learning Multiple Layers of Features from Tiny Images."* 2009. [online] [https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf] [Accessed on July 01, 2019].

[24] Shubham Panchal. *"Artificial Neural Networks - Mapping the Human Brain."* [online] [https://medium.com/predict/artificial-neural-networks-mapping-the-human-brain-2e0bd4a93160] [Accessed on July 01, 2019].

[25] Mukul Malik, *"Basics of Neural Network."* [online] [https://becominghuman.ai/basics-of-neural-network-bef2ba97d2cf] [Accessed on July 01, 2019].

[26] Prabhu, *"Understanding of Convolutional Neural Network (CNN) Deep Learning."* [online] [https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148] [Accessed on July 01, 2019].

[27] Ribhu Lahiri, in The School of AI (official). *"Developing an Intuition for Better Understanding of Convolutional Neural Networks."* [online] [https://medium.com/the-school-of-ai-official/developing-an-intuition-for-better-understanding-of-convolutional-neural-networks-17812fe4722a] [Accessed on July 01, 2019].

[28] *"Vanishing Gradient Problem."* [online] [https://en.wikipedia.org/wiki/Vanishing_gradient_problem] [Accessed on July 01, 2019].

[29] Sik-Ho Tsang, *"Review: Highway Networks Gating Function to Highway (Image Classification)."* [online] [https://towardsdatascience.com/review-highway-networks-gating-function-to-highway-image-classification-5a33833797b5] [Accessed on July 01, 2019].

[30] Max Pechyonkin, *"Understanding Hinton's Capsule Networks. Part I: Intuition."* [online] [https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b] [Accessed on July 01, 2019].