RCNX: RESIDUAL CAPSULE NEXT

by

Arjun Narukkanchira Anilkumar

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Indianapolis, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Mohamed El-Sharkawy, Chair

Department of Electrical and Computer Engineering

Dr. Brian King

Department of Electrical and Computer Engineering

Dr. Maher Rizkalla

Department of Electrical and Computer Engineering

Approved by:

Dr. Brian King

Dedicated to

My Parents: Anilkumar N C and Swapna K A,

My grandparents, teachers, friends, family, colleagues and all well wishers.

ACKNOWLEDGMENTS

I would like to begin by thanking my parents and my teachers for everything. My thesis advisor, Dr. Mohamed El-Sharkawy, whose guidance and wisdom has been crucial throughout my thesis. I am grateful to have had Dr. Brian King and Dr. Maher Rizkalla for serving on my graduate committee and lending their expertise to my research. Additionally, I would like to thank Ms. Sherrie Tucker, who patiently assisted me throughout my graduate studies.

I am grateful to Purdue School of Engineering & Technology for help fund my Thesis development. I am grateful to the IoT Collaboratory at IUPUI for accepting me as a member. The lab provided me with most of the hardware and software support for this research. I would like to thank my colleagues for their insights. I would like to acknowledge Lilly Endowment Inc. for their support to Indiana University Pervasive Technology Institute.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional Neural Networks
RCN	Residual Capsule Network
RCN2	Residual Capsule Network V2
RCNX	Residual Capsule NeXt
NN	Neural Networks
ANN	Artificial Neural Networks
M	million
NNI	Neural Network Intelligence
ELU	Exponential Linear Unit
ReLU	Rectified Linear Unit
MNIST	Modified National Institute of Standards and Technology database

ABSTRACT

Machine learning models are rising every day. Most of the Computer Vision oriented machine learning models arise from Convolutional Neural Network's(CNN) basic structure. Machine learning developers use CNNs extensively in Image classification, Object Recognition, and Image segmentation. Although CNN produces highly compatible models with superior accuracy, they have their disadvantages. Estimating pose and transformation for computer vision applications is a difficult task for CNN. The CNN's functions are capable of learning only shift-invariant features of an image. These limitations give machine learning developers motivation towards generating more complex algorithms.

Search for new machine learning models led to Capsule Networks. This Capsule Network was able to estimate objects' pose in an image and recognize transformations to these objects. Handwritten digit classification is the task for which capsule networks are to solve at the initial stages. Capsule Networks outperforms all models for the MNIST dataset for handwritten digits, but to use Capsule networks for image classification is not a straightforward multiplication of parameters. By replacing the Capsule Network's initial layer, a simple Convolutional Layer, with complex architectures in CNNs, authors of Residual Capsule Network achieved a tremendous change in capsule network applications without a high number of parameters.

This thesis focuses on improving this recent Residual Capsule Network (RCN) to an extent where accuracy and model size is optimal for the Image classification task with a benchmark of the CIFAR-10 dataset. Our search for an exemplary capsule network led to the invention of RCN2: Residual Capsule Network 2 and RCNX: Residual Capsule NeXt. RCNX, as the next generation of RCN. They outperform existing architectures in the domain of Capsule networks, focusing on image classification such as 3-level RCN, DCNet, DC Net++, Capsule Network, and even outperforms compact CNNs like MobileNet V3.

RCN2 achieved an accuracy of 85.12% with 1.95 Million parameters, and RCNX achieved 89.31% accuracy with 1.58 Million parameters on the CIFAR-10 benchmark.

1. INTRODUCTION

1.1 Context

Machine learning took various structures in the previous decade. CNN achieved what once humans thought impossible for machines to do. The rise in computational efficiency and changes in the back propagation algorithms created the platform for CNNs to grow[1]. Requiring various levels of depth and numerous convolutions, CNN outperformed expectations of the human mind. From the beginning of neural networks, machine learning developers take inspiration from the millions of years worth of the human visual system's evolution[2]. Machine learning is an integral part of modern society. From security to games to complex machine drawings, it is now unstoppable. Although the use of CNN exceeds human limitations, it is not ideally similar to the human brain[3]. To merge this gap and produce more similarity to a neural network Capsule Network(CapsNet) is introduced[4]. Capsule Networks focus on neural connections developed with routing by agreement algorithm[4]. This thesis focuses on improving one such network, Residual Capsule Network, where capsule network is merged with ResNet architecture to produce an image classification model.

1.2 Motivation

Copious amounts of neural network architectures helped us understand various levels of the model's complexity[5]. In part, replicating the brain cells, neural networks came into existence. Although many unknown factors remain, we build superficially similar neurons matched with activation functions similar to what can be expected in the brain cell. With growing, neural network architectures came to show limitations of CNN[6]. CNN, also knows as shift-invariant artificial neural networks are, as the name suggests, a convolution function that learns concerning the invariance to a special-shift of the window in an input[4]. These CNNs are helpful in the context of Image recognition, object recognition, image classification, and image segmentation.

This thesis focuses on the Image classification task with exemplary new models RCN2: Residual Capsule Network v2 and RCNX: Residual Capsule Next, outperforming CNN establishing their dominance. Although good at image classification tasks, CNN's is weak, which is easily visible since they are insensitive towards pose and image transformations[7]. This insensitivity also extends to a stage where the spatial correlation between sub-features is discarded. Capsule Networks are developed to target these flaws on CNN and a closer look at the human visual system. In human vision, we observe a robust spatial correlation and strong pose effect where we characterize an object by the pose of its sub-sections.

Capsule Networks work on the principle of routing by agreement. Capsules Neural Networks is a system of artificial neural networks capable of handling better model hierarchical relationships[4]. They are much closer to the biological neural organization compared to CNNs. Having various advantages over CNNs, Capsule Networks outperforms CNNs in understanding the object and its characteristics in an image. Since the routing by agreement creates connections with neurons that agree, parameter number is lesser in caparison to CNNs.

The model Residual Capsule Network is a combination of ResNet CNN architecture and Capsule Network architecture[7]. The residual convolutional blocks are used for the initial layers of the Capsule Network. Using such blocks helped the model to avoid vanishing gradient and improve the application of the Capsule Network.

1.3 Challenges

This thesis aims to modify the Residual Capsule Network to a model comparable to the CNN model with high accuracy and fewer parameters. Challenges in achieving this include the following.

- Creating a new neural network structure from RCN (baseline architecture)[7].
- Reducing model size.
- Training using CIFAR-10 dataset[8].
- Testing the model.
- Tuning hyper-parameters.
- Increasing accuracy of the model.

- Deploying model into an Embedded System.

1.4 Methodology

In this thesis, the following procedures are followed:

- Analyzing the architecture of RCN.
- Finding weaknesses of the structure.
- Finding improvement to the weaknesses of RCN.
- Estimating increase or decrease in the number of parameters.
- Implementing established ideas of improving accuracy
- Confirming architecture's learning capability.
- Check for over-fitting or under-fitting.
- Optimizing parameters with hyper-parameter tuning using NNI.
- Reducing the size of the model with affecting accuracy.
- Optimizing training environment.
- Deploying on embedded hardware.

1.5 Contributions

Contribution towards the completion of the thesis is listed as follows.

- Design space exploration of RCN Architecture.
- Proposed RCN2: Residual Capsule Network V2
- Proposed RCNX: Residual Capsule NeXt.
- Image classification capability of the proposed models are verified.
- Deploying the efficient compressed RCNX into i.MX RT1060.
- Two papers Accepted for IEEE conferences.

2. LITERATURE REVIEW

This section is a synopsis for theoretical details of Neural Networks, Convolutional Neural Networks(CNN), Capsule Neural Networks(CapsNet), Residual Network(ResNets), Residual Capsule Networks(RCN), 3-Level Residual Capsule Networks, and other Capsule Networks[9]. It will discern differences between CNNs and CapsNets. Deep Learning architectures that lead to computer vision. These sections also speak about the missing elements of RCN and 3-level RCN, and these disadvantages are removed with design space exploration further in the thesis.

2.1 Neural Networks

Neural networks (NNs) consist of rules and algorithms that try to imitate the human brain in learning relationships between massive data[10]. NNs are a collection of neurons, simulate a network comparable to a human brain[11]. The NNs can adjust to changing input, so the model creates the leading conceivable result without overhauling the yield criteria[12]. A ”neuron” in an NN is mapping a mathematical function that collects and classifies data concurring to relation. The NNs bear strategies such as regression analysis and curve fitting. Applications of NN are extensive and include Natural language processing, Computer Vision, Stock Market predictions, Astronomical data analysis, etc. We focus on the Computer vision section of Neural Networks[5].

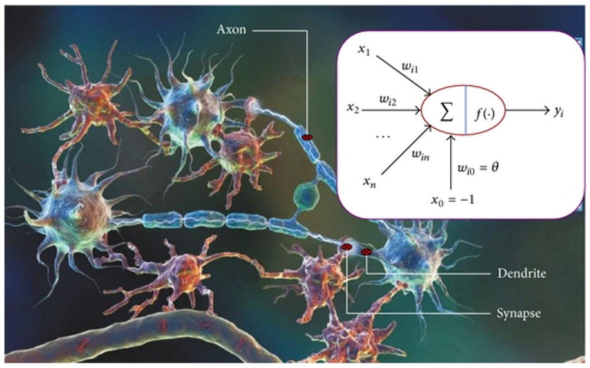


Figure 2.1. Comparison of Biological Neural Network to Artificial Neural Network[13]

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN), also called shift-invariant artificial neural networks, are, as the name suggests, a convolution function that learns concerning the invariance to a special-shift of the window in an input[14], [9]. These help CNN understand image recognition, object recognition, image classification, and image segmentation. The application of CNN extends to videos and lidar point clouds[5]. A typical CNN as depicted in Figure 2.2 contains Convolution Layers, Pool Layer, Activation Layer, and a Fully connected layer. The initial Convolutional layers are intended for extracting features, and further pooling

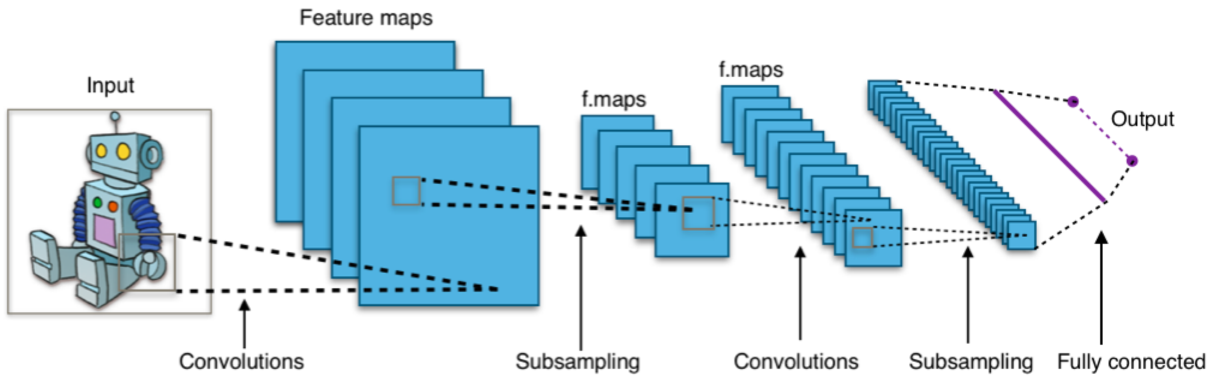


Figure 2.2. A typical CNN[15]

layers reduce the number of connections to improve computation with a lesser number of parameters as visualised in Figure 2.3. Pooling is mainly done in two formats, maximum value pooling, and average value pooling[12]. The pooling layers discount the information to only the average of a kernel square or maximum of a kernel square. These pooling layers cause loss of information, as mentioned by Capsule Network authors. Capsule Network creators see the first and foremost disadvantage in CNN as the pooling layer's loss of information[16].

Pooling layers are embedded extensively in CNN to reduce the size of models. Thereby achieving accuracy with less model size. As the main aim of Neural Networks is to create models comparable to a human brain, these pooling layers lead CNN to be less related to human vision[18].

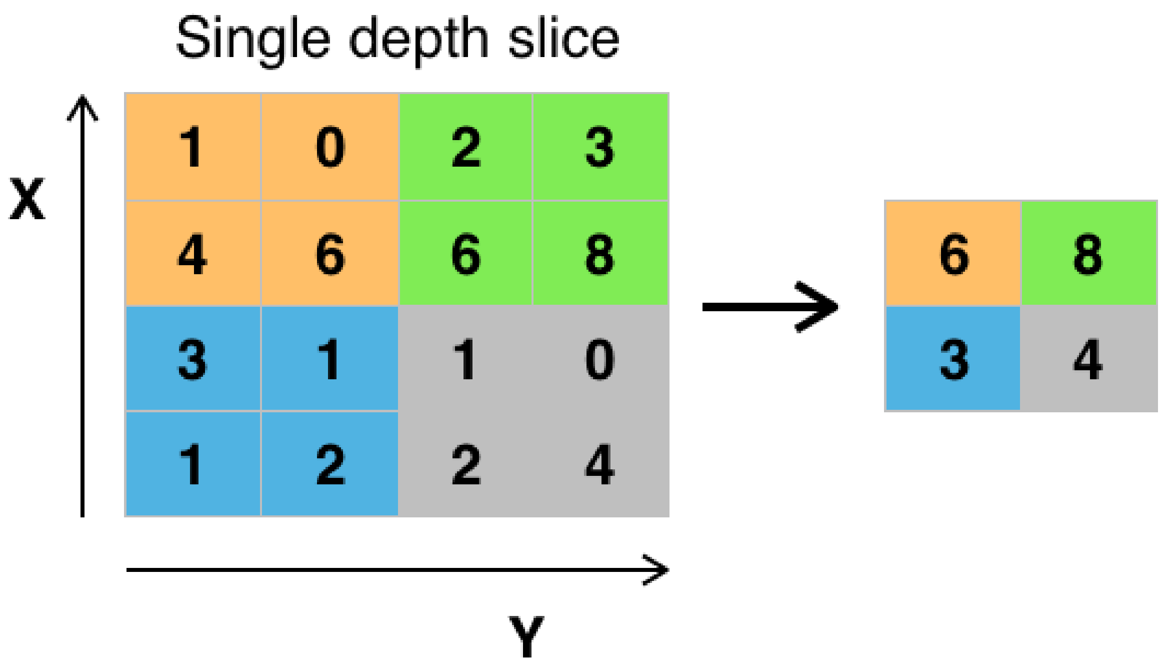


Figure 2.3. Max pooling[17]

Convolutional layers help the network extract features and help in learning features. Convolutional layers irrespective of the method of implementation approximates in feature learning. These features extracted by the CNN pass through the network as scalar features. These scalar features are the second limitation of CNN. As we proceed to the following section about capsule networks, we can see the methods deployed to overcome these flaws.

2.3 Capsule Neural Network

In this section, extensive insight into CapsNet is described. Human vision disregards unimportant subtle elements by employing a carefully decided arrangement of fixation points to guarantee that as it were, a modest division of the optic cluster is ever handled at the most noteworthy determination[4]. Contemplation may be a destitute direct to understanding how much of our information of a scene comes from the arrangement of obsessions and how much we gather from a single obsession. In CapsNet, the idea extends to assume a single focus in the image produces more than a simple recognition and a list of its properties[7]. CapsNet is developed on the idea that the human visual system produces a tree with various analyses of each fixation. Thus, the multilayer model for CapsNet includes the tree-like structures with each layer containing groups of neurons, and these groups are called capsules. A typical CapsNet is depicted in the Figure 2.4.

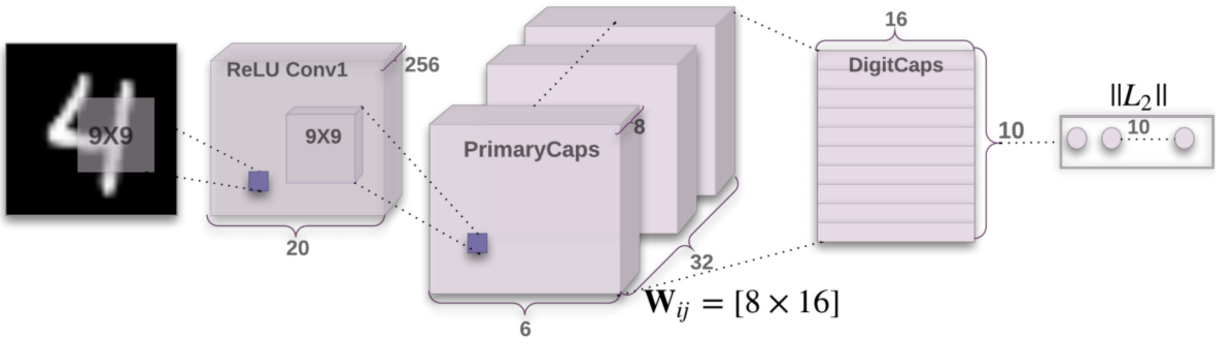


Figure 2.4. CapsNet[4]

CapsNet, being different from CNN, is still a field yet of unfolding its true potential. A typical CapsNet includes a convolution layer, primary capsules, digit capsules, and a dynamic routing algorithm. The salient feature of CapsNet being that features are extracted and represented with vectors of N-dimension based on capsules[4]. Capsules are groups of neurons that produce an N-dimensional vector consist the actualization details of an entity like an object or sub-part of an object. This vector's length denotes the object or sub-part of objects existence with a probability, and the direction of the same represents the entity's pose.

Dynamic capsules make expectations utilizing the change in the lattices, which changes upper hierarchical capsules instantiation parameters[4]. When numerous expectations concur, the next level capsules get activated. With expectations produced by low-level capsules, a route to the high-level capsule is so that low-level capsules' expectation provides an agreement to the high-level capsule. This agreement-based routing is called routing by agreement algorithm, and it is the dynamic routing algorithm used by CapsNet. To obtain the routing-by-agreement scalar product of probability vectors of the low-level capsule and high-level capsules is calculated, and the product with a more significant value is preferred to be the route[19].

2.3.1 Capsule vector input and output computation

There are numerous ways to actualize the common idea of capsules. CapsNet capsules aim to produce a likelihood vector that an entity is present in the input. To accomplish this, CapsNet use the squashing function as described below[4].

$$v_{\mathrm{j}} = \frac{\|s_{\mathrm{j}}\|^2}{1 + \|s_{\mathrm{j}}\|^2} \frac{s_{\mathrm{j}}}{\|s_{\mathrm{j}}\|} \tag{2.1}$$

While training the model, it will use the non-linearity in the squashing, and the above function also provides normalization to the vector outputs. From layer two, all layers consider input $\mathbf{s_j}$ with a weighted sum operation with a weighted matrix $\mathbf{W_{ij}}$ over all the inputs with the following equation[4].

$$s_{\mathrm{j}} = \sum_{\mathrm{i}} c_{\mathrm{ij}} \mathbf{\hat{u}_{j|i}}, \quad where \;\; \mathbf{\hat{u}_{j|i}} = W_{\mathrm{ij}} u_{\mathrm{i}} \tag{2.2}$$

In the above equation $\mathbf{c_{ij}}$ is the coupling coefficient learned from training via dynamic routing algorithm. The loss function of CapsNet is give in Figure 2.5.

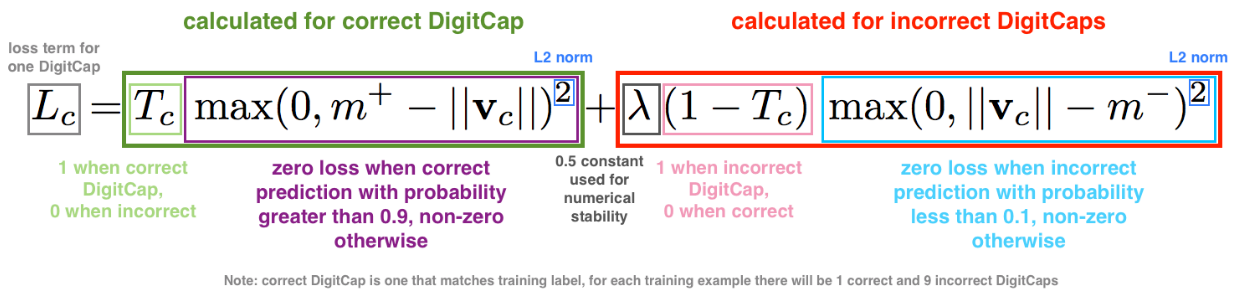

calculated for correct DigitCap
calculated for incorrect DigitCaps
loss term for one DigitCap
L2 norm
L2 norm
$L_c = T_c \max(0, m^+ - ||\mathbf{v}_c||)^2 + \lambda (1 - T_c) \max(0, ||\mathbf{v}_c|| - m^-)^2$
1 when correct DigitCap, 0 when incorrect
zero loss when correct prediction with probability 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

Figure 2.5. CapsNet loss function

2.3.2 Routing-by-agreement algorithm

In general, routing algorithms are of two types adaptive routing algorithms and non-adaptive routing algorithms. Capsule networks take an adaptive routing algorithm, which is established with routing by agreement mechanism. Routing by agreement is a dynamic routing algorithm[4], [10]. To obtain the routing-by-agreement scalar product of probability vectors of the low-level capsule and high-level capsules is calculated, and the product with a more significant value is preferred to be the route. The dynamic routing algorithm used by the CapsNet is given in Algorithm 1[4].

Algorithm 1 Dynamic Routing Algorithm

```
procedure: ROUTING($\hat{\mathbf{u}}_{j|i}, r, l$)
    Initialisation :
    for all capsule i in layer $l$ and capsule j in layer $(l+1)$ : $b_{ij} \leftarrow 0$
    for $r$ iterations do
        for all capsule i in layer $l$ : $c_i \leftarrow \text{softmax}(b_i)$
        for all capsule j in layer $(l+1)$ : $s_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
        for all capsule j in layer $(l+1)$ : $v_j \leftarrow \text{squash}(s_j)$
        for all capsule i in layer $l$ and capsule j in layer $(l+1)$ : $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.v_j$
    end for
    return $\mathbf{v}_j$
```

2.3.3 CapsNet Architecture

The architecture of CapsNet, given in Figure 2.4, is not deep since only two convolution layers, and a fully connected layer are present. In detail, the first convolutional layer with 256 channels, 9×9 kernel, the stride of one, and ReLU activation effectively transforms pixel information into functions producing local feature detectors. These detector features are then fed forward to primary capsules. The lowest possible degree of the capsule layers starts at the primary capsule[7]. As discussed above, CapsNet focuses on two processes, the rendering, and inverse rendering, out of which the primary capsules, when triggered, resembles inverse rendering. This distinctive sort of computation, then patching instantiated parts collectively to create commonplace wholes, is what capsules are planned to be great at.

The primary capsules used by CapsNet are with 32 channels of eight-dimensional capsules. This means that there will be eight convolution units with the kernel of size 9×9 and two as stride. Since the layer before primary capsules generates 256×81 convolutional units with overlapping fields in the center of the input, each capsule in the primary capsule layer has all that it needs to see from the outputs of the Convolution layer. With block non-linearity of equation 2.1, across the CapsNet primary layers with a grid size of $[32 \times 6 \times 6]$[4]. Each of them has 8D vector output, and primary capsules can be seen as a Convolutional layer. DigitCaps is the final layer of the CapsNet, and it contains 16-dimensional vectors per class. Since it is used for handwritten digit classification, it is digit classes and hence the name DigitCaps[4]. Routing is only necessary between 2 consecutive capsules.

Reconstruction network, given in Figure 2.6, is a crucial part of the CapsNet. A prediction of estimated vectors in DigitCaps is assumed to be a collection of the image features and then reconstructed. This reconstruction loss amounts to the support of DigitCaps to get better instantiation of the input image. The reconstruction network used by CapsNet is three fully connected layers that finally give the reconstruction to an output same as the input image. Reconstruction loss is scaled to a minimal value to avoid the dominance of this loss alone.

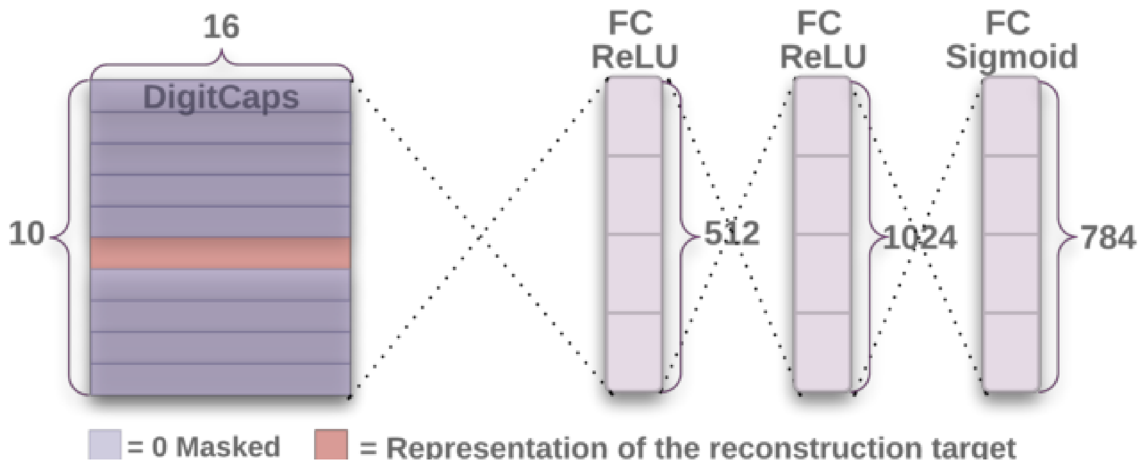


Figure 2.6. CapsNet Reconstruction Network[4]

2.3.4 Performance of CapsNet

CapsNet is trained to a dataset MNIST, which contains images of handwritten digits with a size of 28×28. The CapsNet performed very well and produced an error of only 0.25% for the test. This accuracy is achieved with only a three-level depth model[4].

CapsNet is not intended for complex image datasets like the CIFAR-10. Thus a version that fits the CIFAR-10 like dataset is the seven ensemble model of the CapsNet. This seven ensemble model's accuracy is 89.40% on tests with a size of 101.5 Million parameters[7].

2.4 Residual Network

Deeper neural networks are better for accuracy. Increasing the number of parameters and depth of the network is considered the easy, straightforward way to improve the model's accuracy. From the quotes, it can be noted such an improvement in the model size has a limit[20]. In machine learning, this limit is produced due to vanishing gradient and problems regarding dimensionality[6]. Deep networks often face these problems. Vanishing gradients have been existent from the initial days of Deep Neural networks. To overcome this hamper in convergence first method to existing was a normalized initialization with embedded normalizing layers. These techniques allowed proper back-propagation with reducing the vanishing gradient problem[20]. Improving from these basic techniques, which take time and effort, newer solutions started to surface. Deep residual learning framework addressed

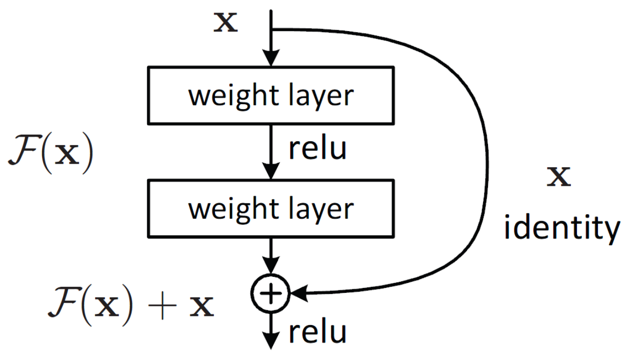

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.7. Residual Network[20]

these issues in a different architecture. This solution is elegant and faster convergence is achieved. Rather than trusting every few stacked layers fit a desired elemental mapping, deep residual learning unequivocally lets these layers fit a leftover mapping. Let an underlying architecture be $F(x)$, and residual learning provides an additional overlap of the input to the model's subsection by making the mapping function $F(x) + x$[20]. These feed-forward networks are called skip connections or shortcut connections. These networks were named Residual Networks (Figure 2.7).

Residual Networks are conceivably the foremost imaginative work within the Computer Vision community within the past decade. The additional is an identity function of the input to the model's subsection, and this can also be called identity shortcut connections. ResNets were capable of training numerous layers without affecting the accuracy or saturating of the model's accuracy[20]. With this technique alone, many image classification and object detection techniques have improved tremendously. By experiments, it was established that models without shortcuts tend to learn slower and reach saturation sooner than the models with the same architecture but with the above-mentioned 'identity shortcut connections.'

Simplicity in concept, Residual Networks truly makes implementations easy. In a regular CNN, each layer's output is considered as the only input to the following layer. In ResNets, a layer's input in a high-level feature extractor is the prior layer's output and a feed-forward skip connection from 2 to 3 layers before it. ResNets contains skip-connections to layers where a vanishing gradient problem might occur. Even with hundreds of layers, Residual connections achieved complete removal of vanishing gradient. Pre-activation residual connections are found to be better to help in the passage of information. Residual Networks have accomplished a striking execution on Image classification errands by presenting skip associations utilized as bypassing ways[21]. It can be found that skipping has viably rearranged the network and expanded the learning speed by diminishing the effect of vanishing gradients as there are fewer layers to proliferate through.

2.5 Baseline: Residual Capsule Network (RCN)

A deeper network is effective than shallow systems but, consequently, more cumbersome in the training process. ResNets ease the preparation and have appeared to prove that they can give great precision with significant profundity. By utilizing Residual Network with the Capsule Network, the RCN model came into existence[7]. RCN Architecture is as shown in Figure 2.8. Skip routes improve the customary Convolution layer in the initial layer of CapsNet, just like the ResNets, to diminish the network's complexity and seven-ensemble CapsNet. This model was trained and tested on MNIST and CIFAR-10 datasets and has seen a significant decrease in the number of parameters compared to the seven-ensemble residual models[7].

The first convolutional layer in CapsNet identifies the natural highlights within the input 2D image. However, this might not be enough for complex datasets to prepare for advanced features in the capsules. Thus, RCN extends the profundity by including more convolutional layers. Basically, stacking the layers may not lead to any change. This issue can be where ResNet comes to protect. Within the RCN, the authors modified the convolutional layer with the shortcut connections mentioned in ResNet architecture[7]. These skip connections thus provide the necessary paths for avoiding vanishing gradient.

The initial ResNet convolutional layers brought depth to the network. In RCN, they utilized eight consecutive ResNet convolution layers stacked back to back. All residual connections were combined in the feedforward layers. ResNet convolution layers take the input image and transform the three channeled image into a 32 channel with eight layered feature detectors. These feature extractors generated layers of 256 feature maps.

Furthermore, the extracted 256 channeled features are fed into the CapsNet capsules. As discussed, the CapsNet capsules use 9×9 convolutional layers with Rectified Linear Unit activation. On the existing model of CapsNet, variations in the primary capsules were also created for RCN. The CapsNet, being an equivariant model, is necessary to maintain the feature extractors to follow the same model nevertheless of any modifications. This was sustained in RCN as there are no pooling layers embedded in it. Adding pooling layers with the convolution layers will lead to a functionality change. For this reason, RCN excluded any embedded pooling layers.

As discussed in the CapsNet section, the primary capsules produce the extracted features into output vectors of eight dimensions. These are then fed into the DigitCaps layer. The DigitCaps layer can extract other complex functions from the eight-dimensional feature

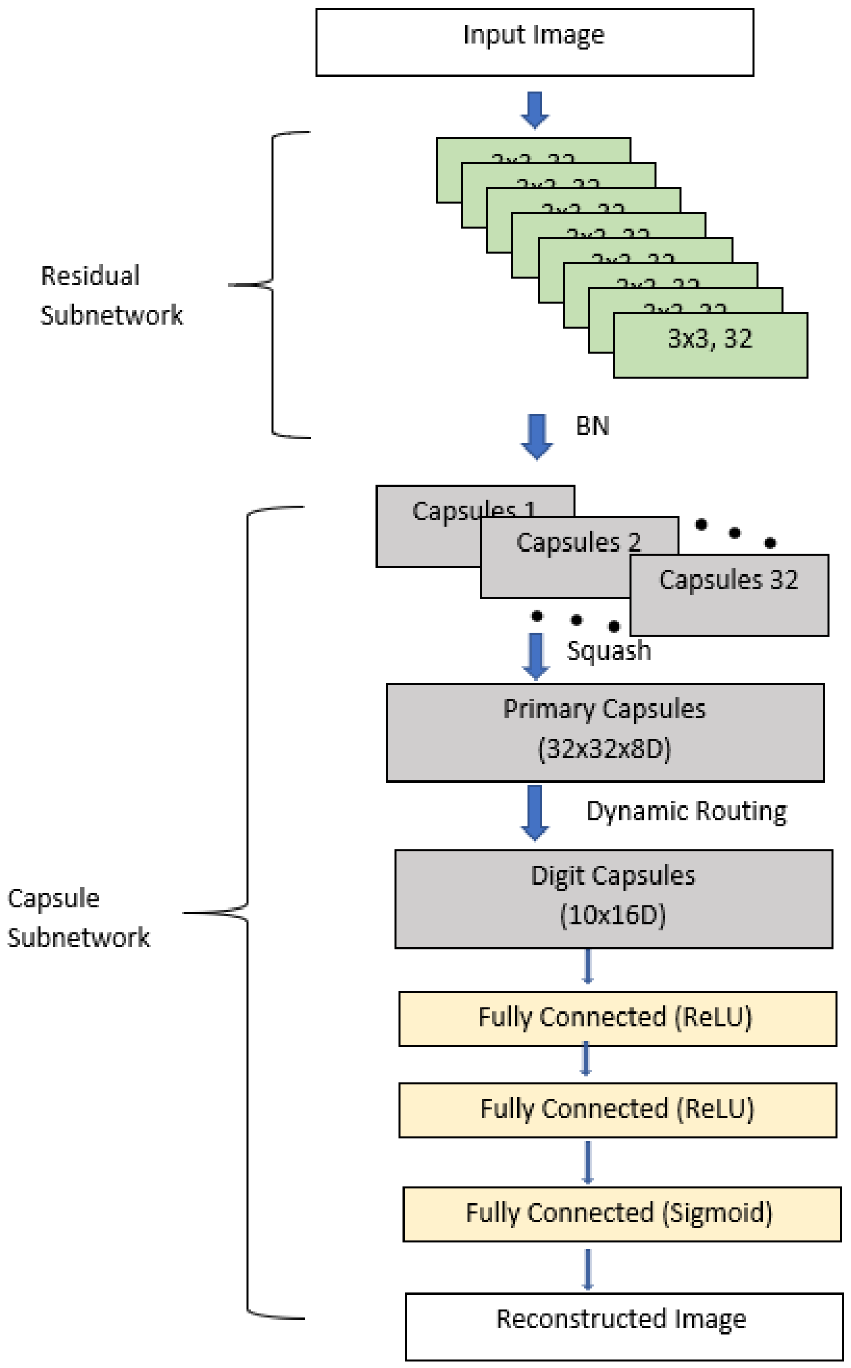


Figure 2.8. Residual Capsule Network(RCN)[7]

vectors by mapping these into a 16-dimensional space with weighted coefficients[7]. As with every classification tasked Neural networks, the RCN converges to a final layer of the number of classes, i.e., ten classes for the CIFAR-10 dataset. Thus the DigitCaps output of 16-dimensional vectors is then mapped to ten neurons to handle the training output as a one-hot encoding method.

We can note RCN has a vast amount of parameters generated solely from the thick redundant layers of ResNet convolution. Like any CapsNet based architecture, the RCN had a reconstruction network. The reconstruction networks are generally connected with the network's output as input and the network's input as output. On a detailed study, we find the RCN lacking the required complexity in the reconstruction network. We assume that this is due to the network's size, and adding a whole complex reconstruction network will add many more parameters to the existing RCN. With this in mind, we fixed our aim to not focus on the reconstruction network as it is easily removed after the training. It is crucial to understand the use of a reconstruction network is to provide an extra gradient flow during training and hence o not pose any actual use while deploying or testing the model.

RCN model produced an accuracy of 84.16% with 11.86 M(million) parameters for the CIFAR-10 dataset compared to the accuracy of 89.40% with seven ensemble Capsule Network model with 101.5 million parameters[7]. This model was a good achievement, but we believed focusing on the flaws mentioned above can bring the best out of RCN.

2.6 YOLO-v3

You Only Look Once(YOLO) is an extensive model for object recognition in an image[22]. YOLO-v3 became one of the popular models used by developers to deploy and execute at a fast pace quickly. Currently, YOLO-v4 has come to light and is the fastest algorithm for object recognition. The YOLO model recognizes the object and draws a bounding box within which the object is more likely to be present. YOLO employments a preparing set that consists of pictures and their comparing bounding boxes of the target pictures[22]. Thus, an exemplary model, YOLOv3, was chosen to inspire the RCN authors to bring another network 3-level Residual Capsule Network model.

The idea that was inspiring in YOLOv3 is a three-staged network where the network views different input picture scales through each stage. These stages are mainly due to the necessity of YOLO as they were focusing on different scaled images. The structure of YOLO-v3 is shown in Figure 2.9[23].

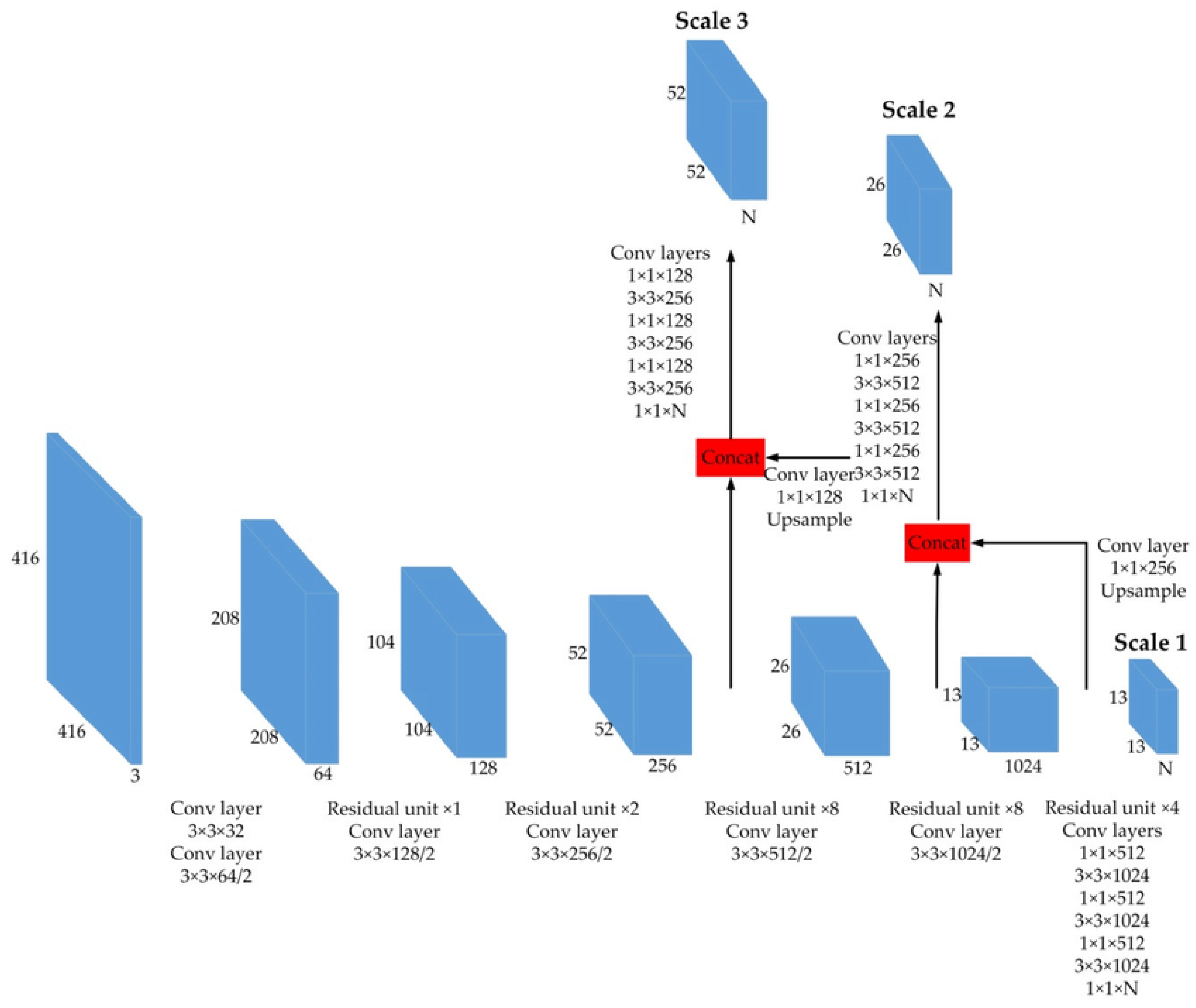


Figure 2.9. YOLO-v3 Structure[23]

The YOLO implements a feature-dependent trainable model consisting of 75 convolution layers. YOLO altogether avoided fully connected layers by deploying the convolutions of specific sizes, which produced a fully connected layer-like structure and at the same time avoided pooling layers as this also can be replaced with convolutions which can be seen as a weight-based pooling layer[22]. They also avoided using SoftMax activation. This activation in itself limits the network, although it is necessary to get convergence.

2.7 3-level Residual Capsule Network

The authors of RCN brought the salient features of YOLO v3 and combined them with RCN[9]. These features brought tremendous change into the field of RCN. The authors implemented a three-staged network of 8 layered ResNets, finally connections that interface from every eight layers of ResNets to Primary capsules. A straightforward layer of Residual Network did not unravel the issue for complex datasets like CIFAR-10[9]. This lack of performance is possible since the straightforward essential capsules may not be sufficient to compute all the picture highlights.

In the 3-Level RCN network, the primary capsules were modified to retrieve images at three stages of the ResNet layers. The seven-ensemble CapsNet produced a test result of 89.4%. With this high accuracy, it could have been just enough if the size of the model of seven-ensemble CapsNet was not 100 Million parameters[9]. This number of parameters is very high compared to any machine learning model. Although a 3-level residual capsule network aimed at reducing these parameters, which they were able to achieve, we believe there is room for more improvements.

The structure of the 3-level residual capsule network, as shown in Figure 2.10, was inspiring that we brought these with the new architectures. A hierarchical structure like a pyramid was the core of YOLOv3 and is as well the structure of 3-level RCN. Each level in this pyramid-like layer contained a layer of repetitive eight cumbersome RCN layers. In 3-level RCN, the primary capsules were fed in with squashed 12 capsule inputs, and then these capsules output were provided to the digit caps. The DigitCaps used a dynamic routing algorithm like the CapsNet. They then were fed into the reconstruction layers of Fully Connected neurons and with ReLU activation for two layers and the final layer with 10 unit output with a sigmoid activation function[9]. In 3 level RCN, the authors added one extra layer of DigitCap, leading to 4 DigitCaps in the DigitCap layer. This final addition of the

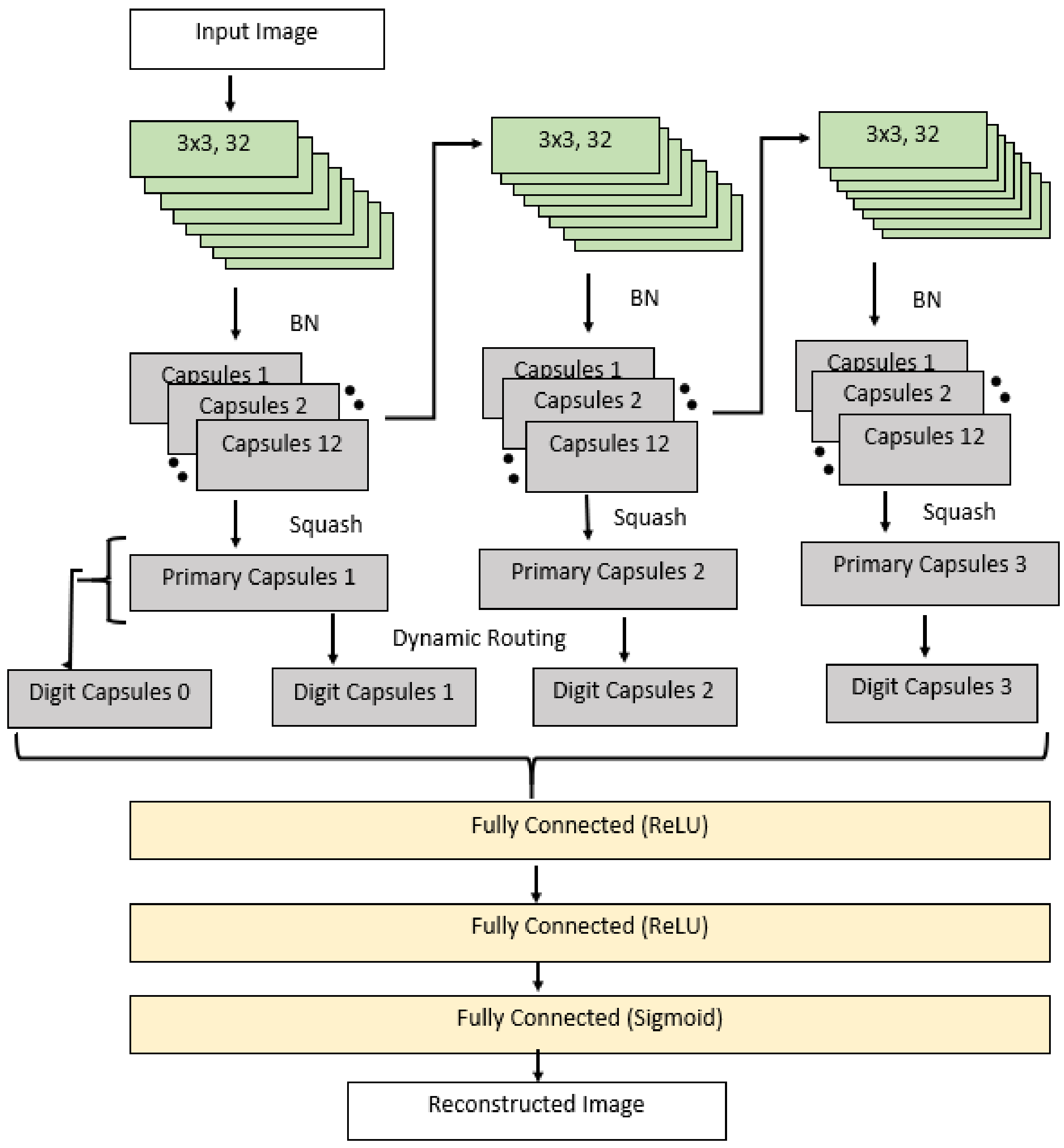


Figure 2.10. 3-Level Residual Capsule Network

DigitCaps led to a merger of all primary capsules fed into it. This new merger DigitCaps was another exciting and innovative structure brought forth by the 3-level architecture.

The 3-Level RCN performed better than RCN when tested on the CIFAR-10 dataset. Although initial RCN produced an accuracy of only 84.16% with 11.86 M parameters, the later 3-level RCN achieved 86.42% accuracy with 10.8 M parameters[9]. However, an impressive reduction in size accuracy came with a drop of 4% accuracy from the baseline CapsNet. We believe the full potential of RCN is yet to be realized.

2.8 ResNeXt and Cardinality

ResNext NN is the model that brings improvements to the structure and performance of ResNet[24]. These changes are mainly revolving around the convolutions. The convolutional layers of ResNet are changed with an additional dimension. ResNeXt is a neural network that brought advancements to ResNet. The ability to squeeze the conventional convolutions in the ResNet with the addition of Cardinality leads to higher performance with a reduction in ResNet size[24]. The Cardinality is an additional dimension after the number of filters helped improve the network to a large extend. Since the Cardinality brings a certain complexity to the model's convolutions, it can be easily assumed any replacement of ResNet convolutions with the ResNeXt convolutions will bring improvements, as demonstrated by the authors of ResNeXt. This network is shown in Figure 2.11.

Cardinality portrays the degree of changes. The usage of Cardinality in ResNet design leads to ResNeXt. ResNeXt was a leading classification machine learning model for the COCO dataset. Although the dataset variation is to be accounted for, it can be noted that even YOLOv3 was trained with the COCO dataset. Since the YOLO architecture inspires the existing 3-level residual capsule network model, it can be assumed that ResNeXt and the 3-layered structure will have no reason to be not compatible with each other[22]. The idea of Cardinality arises from the insights provided by various models of convolutions, i.e., Multi-branch convolution network, Grouped convolutions, Compressing convolution networks, and Ensembling[25].

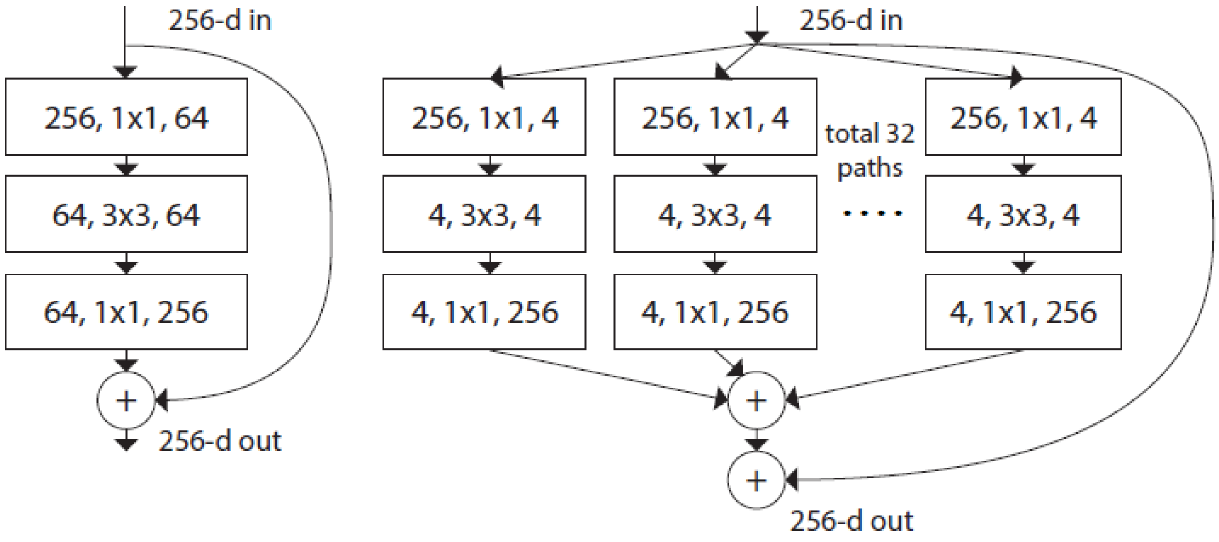


Figure 2.11. Left: ResNet; Right: ResNeXt with cardinality 32[24]

ResNeXt proved its dominance over many modern machine learning algorithms, like ResNet, Inception, Inception-ResNet, etc. ResNeXts followed the idea of Network in Neuron that a Network in Network model. This idea expanded to the creation of a new dimension. This new method was stated as aggregated transformation by the ResNeXt authors[24]. With experiments, the paper of ResNeXt shows in detail how the extra dimension is essential to the model's capabilities and that it is more convenient and successful than the depth and width dimensions.

It is crucial to note that the ResNeXt capability comes with no extra parameters and is just an architectural strengthening tool. We discuss more on how we utilized this capability to improve RCN in the later sections.

2.9 DeepCaps

We discussed various changes in the convolutions leading to improvement in the CapsNet as it is clear that CapsNet size highly depends on the dynamic routing algorithm as well. Therefore, various dynamic routing algorithms were considered, and then on analysis of each, we found that the DeepCaps brought better routing algorithm. DeepCaps aimed at the Capsule Network's depth. DeepCaps focused on intuition as we go deeper in a layer like a CNN, the performance improves. This 'going deeper' was achieved by improving the rouging algorithms[26].

The DeepCaps is designed and developed with estimations of applying to the similar classification tasks on CIFAR 10. The highlight of DeepCaps is the new routing algorithm, a 3D-convolution based dynamic-routing algorithm[26]. Along with the improvements in the dynamic routing algorithm, the DeepCaps also focuses on other factors such as the decoder network. The decoder network was improved to incorporate the class independent decoder. The 3D-convolution based dynamic routing algorithm is given in Algorithm 2[26].

In detail, the new dynamic routing algorithm is giving vast importance in avoiding unnecessary routes. This change is achieved by considering the neighboring neurons activate similarly and provides similar instantiation parameters in higher-level capsules. DeepCaps removes this redundancy by involving a convolution in the dynamic algorithm. Typically a 3×3 filter with one channel is used inside the older dynamic routing, thus the name 3D convolution-based dynamic routing algorithm. DeepCaps understands that the depth of CapsNet matters for complex networks, and thus by removing these redundant routings enables the CapsNet to handle better depths.

Consider an N channel input to the new dynamic routing, and it can be seen that DeepCaps achieved to create a 3D voting-like system, where the winning mode is detected with a weighted summation. Using this technique number of parameters involved in CapsNet were reduced by a factor of $c * (w^L w^{L+1})^2$ parameters in each capsule, where c represents channels, and w^L represents the width of layer L[26].

3. PROPOSED ARCHITECTURES

This section details on the two architectures proposed and implemented toward the completion of this thesis.

3.1 RCN2: Residual Capsule Network V2

As we discussed, CNNs can only work with the shift-invariance in an image, due to the parameter sharing of convolutional layers and a partial effect from pooling layers. This is the limitation that we try to overcome with CapsNet. CapsNet, being different from CNN, is still a field yet of unfolding its true potential. A typical CapsNet includes a convolution layer, primary capsules, digit capsules, and a dynamic routing algorithm[26]. The salient feature of CapsNet is that features are extracted and represented with vectors of N-dimension based on capsules[4]. CapsNet focuses on two processes, the rendering and inverse rendering, out of which the primary capsules, when triggered, resembles inverse rendering. CapsNet are great at this distinctive sort of computation to achieve inverse rendering.

The inspiration for the proposed RCN2 architecture is from RCN (Figure 2.8). From RCN, Residual Networks are being used along with the CapsNet[7]. A deeper network is effective than shallow ones but, consequently, more cumbersome in the training process. ResNets ease the preparation and have proven that they can give great precision. By utilizing Residual Network with the Capsule Network, the RCN model came into existence. The combination of two has proven extensively helpful[7].

RCN2 is a compressed and improved version of RCN. We implement many functionalities which helped improve the RCN to become a better Network, and we named it RCN2. RCN2 architecture is depicted in Figure 3.1. The improvements that were included are Bottleneck architecture, 3D convolution based dynamic routing, Mish activation function, and changes in the reconstruction network. These are detailed in the following subsections.

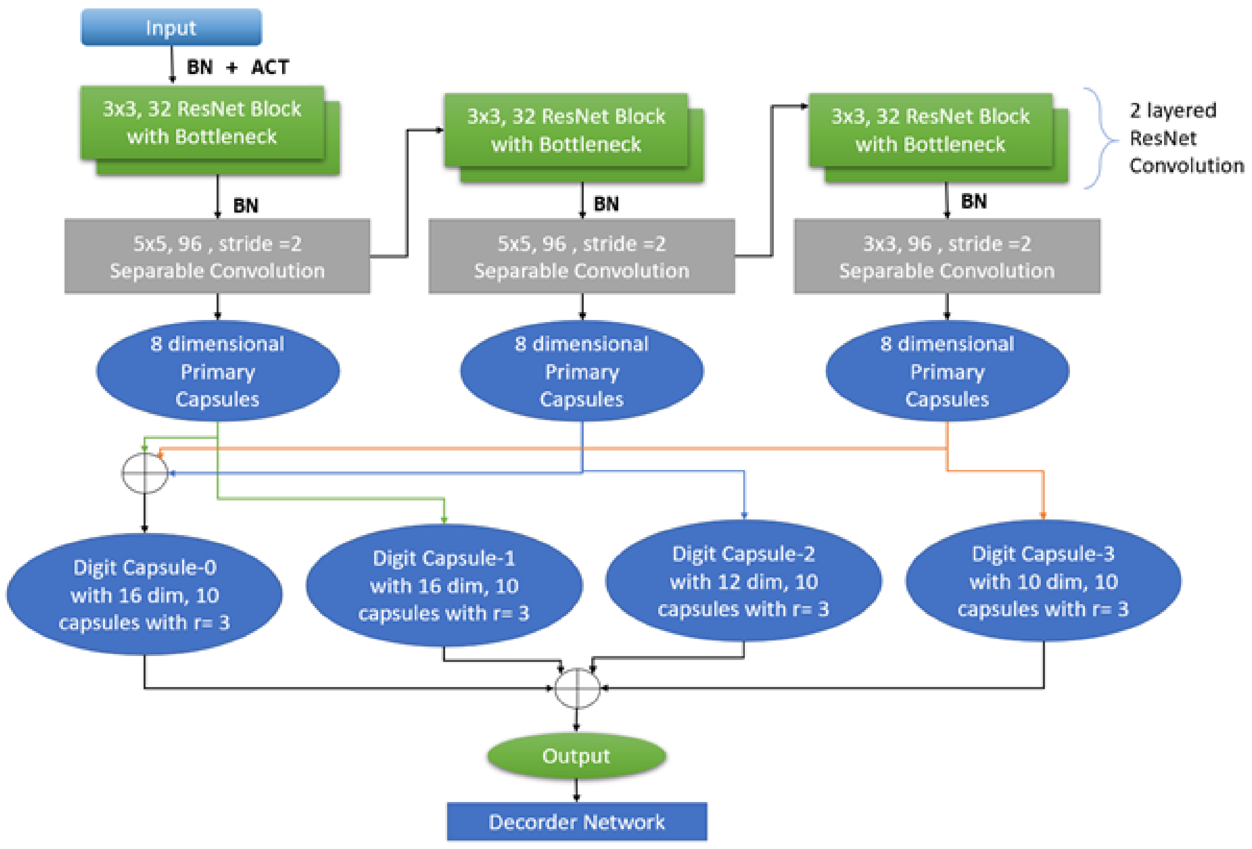

Input
BN + ACT
3x3, 32 ResNet Block with Bottleneck
BN
5x5, 96 , stride =2 Separable Convolution
8 dimensional Primary Capsules
3x3, 32 ResNet Block with Bottleneck
BN
5x5, 96 , stride =2 Separable Convolution
8 dimensional Primary Capsules
3x3, 32 ResNet Block with Bottleneck
2 layered ResNet Convolution
BN
3x3, 96 , stride =2 Separable Convolution
8 dimensional Primary Capsules
Digit Capsule-0 with 16 dim, 10 capsules with r= 3
Digit Capsule-1 with 16 dim, 10 capsules with r= 3
Digit Capsule-2 with 12 dim, 10 capsules with r= 3
Digit Capsule-3 with 10 dim, 10 capsules with r= 3
Output
Decorder Network

Figure 3.1. Architecture of proposed RCN2: Residual Capsule Network V2

3.1.1 Improving Residual Convolution layers with Bottleneck

As we understand from the literature review, the initial layer of the CapsNet is a convolutional layer. It is also important to note that this convolution layer is solely responsible for extracting the images' features. Multiple layers of the convolution will provide the model with the suitable complexity to learn features better. RCN authors improved this with the inclusion of 8 repetitive ResNet layers. These layers are lacking the ability to learn better as there is no bottleneck structure embedded in them.

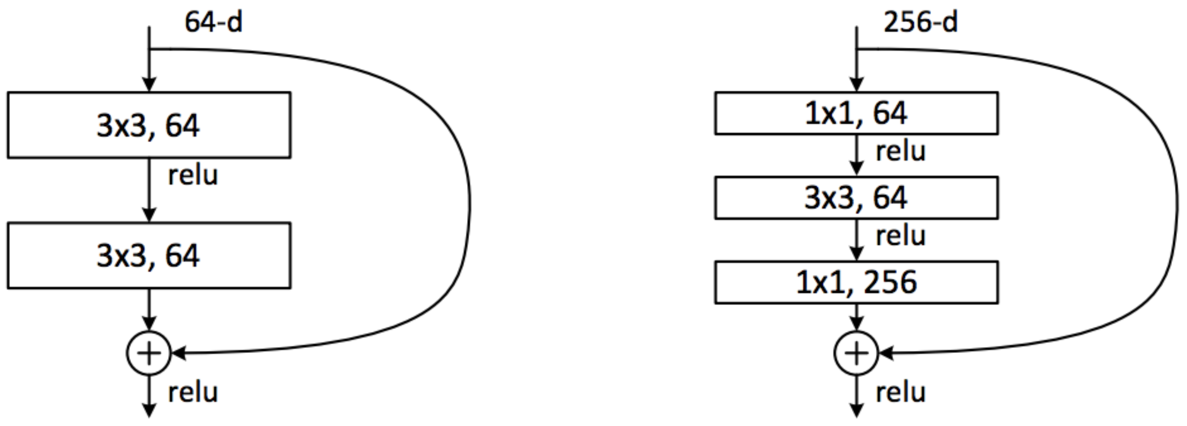


Figure 3.2. Left: ResNet, Right: ResNet with Bottleneck[20]

We included Bottleneck to the ResNet blocks, like in Figure 3.2. Bottleneck design is supposed to reduce the number of parameters and matrix multiplications[20]. Here we expanded the channel size to include more complexity by the supplement bottleneck layers. Thus reaching a balance of complexity with slightly reduced size. Furthermore, we removed redundant layers of RCN. In RCN, there are eight levels of redundant ResNet layers. This redundancy is removed by adding the new ResNet layers with Bottleneck, removing six layers, and keeping two ResNet levels. Additionally, only one of the two layers is provided with the bottleneck architecture.

We understand that supplementing a bottleneck design to the given layer will diminish the preparation time. Allowing training to run smoother than the baseline RCN. In each ResNet layer where the RCN authors utilized 2-layers, we changed to a 4-layer structure containing

$1 \times 1, 3 \times 3, 1 \times 1$, and 3×3 convolutions. Inserted with bottleneck strategy, which, together with Identity routes, gives less time complexity and less neural network model size. We configured this network to a 3-level structure to improve the performance. Thus at each stage, instead of 8 redundant layers, we reduced them to 2 ResNet with bottleneck layer.

3.1.2 3D convolution-based dynamic routing

As examined earlier in the literature review, the initial layers of the CapsNet are convolutional. In CapsNet the output of these convolution-based layers is fed into primary capsules. The primary capsules transform the input vectors from the local feature detectors to eight-dimensional capsules as output. These are then fed into Digit capsules. Going deeper with a model is always what is done to increase performance in many CNN models. CapsNet's dynamic routing is such that they flatten the primary capsules' output vectors, and then they are routed with the dynamic routing algorithm as given in Algorithm 1[4], [26]. This structure gives rise to a structure similar to a multi-layer perceptron model. This structure is a highly time-consuming one[26]. Stacking these to reach better performance will drastically pull the network's speed and efficiency, may even limit improvements after certain depth.

To accumulate multiple layers of CapsNet, it is needed to establish a dynamic routing with some effect of a convolution-like process. This effect in process is achieved by the 3D-convolution based dynamic routing algorithm proposed by the DeepCaps. This algorithm is given in Algorithm 2[26]. Within the Digit capsule layer, vector feature maps are squashed and steered through an dynamic routing calculation. The new dynamic routing algorithm from DeepCaps massively dropped the redundancy by directing squares of capsule s, from layer L to layer $L+1$, rather than directing all capsules in layer L separately[26]. This change was brought with the idea that the neighboring capsules generate similar predictions.

As we know, the primary capsules create an abstract estimation of parts of an object. Further, they create a prediction of the presence of these parts and even with the transformation parameters of it. This inverse rendering effect is crucial in the CapsNets. The next most important step is to create a routing by agreement from the primary capsules to the next level capsules, which is usually termed as DigitCaps. The DigitCaps as one can imagine

Algorithm 2 3D convolution based Dynamic Routing Algorithm[26]

```
procedure: ROUTING($\mathbf{\hat{u}_{j|i}}, r, l$)
    Require: $\Phi^l \in \mathbb{R}^{(w^l,w^l,c^l,n^l)}$, $r$ and $c^{l+1}, n^{l+1}$
    $\tilde{\Phi}^l \leftarrow Reshape(\Phi^l) \in \mathbb{R}^{(w^l,w^l,c^l \times n^l,1)}$
    $V \leftarrow Conv3D(\tilde{\Phi}^l) \in \mathbb{R}^{(w^{l+1},w^{l+1},c^l,c^{l+1} \times n^{l+1})}$
    $\tilde{V} \leftarrow Reshape(V) \in \mathbb{R}^{(w^{l+1},w^{l+1},n^{l+1},c^{l+1},c^l)}$
    $B \leftarrow 0 \in \mathbb{R}^{(w^{l+1},w^{l+1},c^{l+1},c^l)}$
    Let $p \in w^{l+1}, q \in w^{l+1}, r \in c^{l+1}$ ans $s \in c^l$
    for i iterations do
        for all $p, q, r, k_{pqrs} \leftarrow softmax\_3D(b_{pqrs})$
        for all $s, S_{pqr} \leftarrow \sum_s k_{pqrs} \cdot \tilde{V}_{pqrs}$
        for all $s, \hat{S}_{pqr} \leftarrow squash_3D(S_{pqr})$
        for all $s, b_{pqrs} \leftarrow b_{pqrs} + \hat{S}_{pqr} \cdot \tilde{V}_{pqrs}$
    end for
    return $\mathbf{\Phi^{l+1}} = \mathbf{\hat{S}}$
```

is similar to the primary capsule with the fact that these take more complex object detection in the image[4]. DigitCaps also generate the prediction of presence of the capsules objects and also the instantiation parameters of the particular object in each location.

3.1.3 Mish activation

The activation function gives the NN flexibility to incorporate the required nonlinearity to learn mapped functions. This flexibility provides a fundamental part of the implementation and gaining accuracy. The baseline model, RCN, uses the ReLU (Rectified Linear Unit) activation function after every ResNet layer. The function of ReLU is as follows:[27]

$$f(x) = x^+ = max(0, x) \tag{3.1}$$

In RCN2, we utilized the very flexible Mish Activation. Via experiments, we tested and trained various activation functions, and the Mish activation function provided a boost to the network's accuracy. Mish activation function is as follows (Figure 3.3)[28].

$$f(x) = x\, tanh(softplus(x)) \tag{3.2}$$

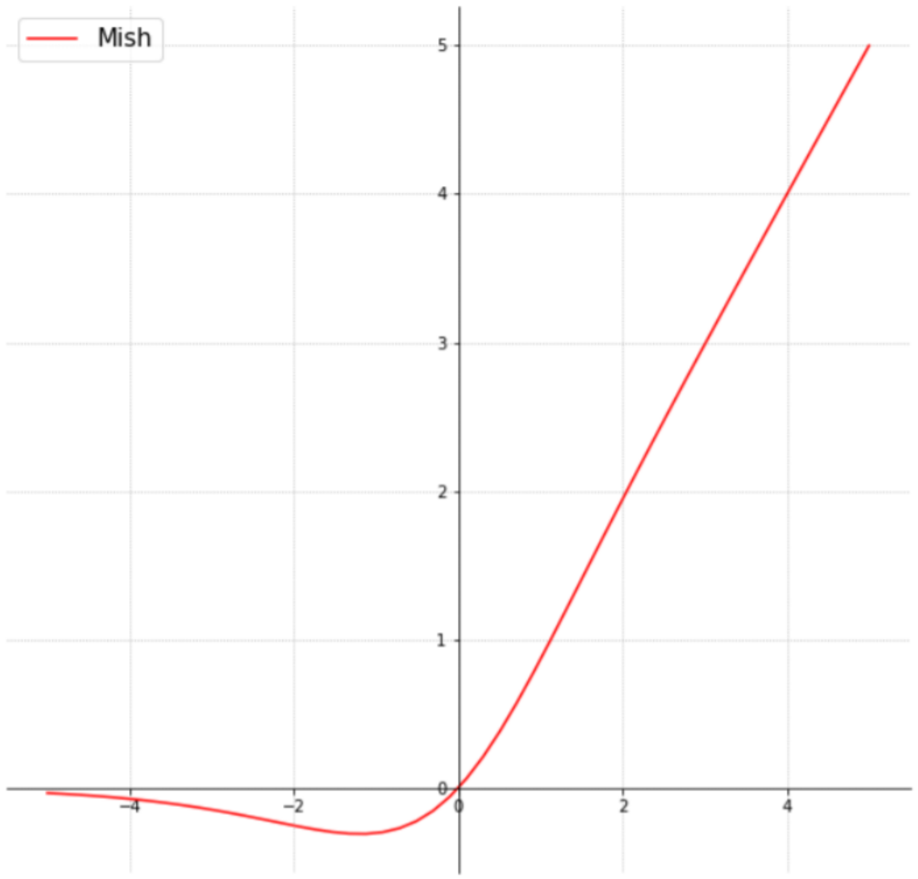


Figure 3.3. Mish activation function[28]

Mish function has certain properties, which led to our conclusion that Mish activation was the right choice for RCN2. Mish activation prevents over-fitting and provides the necessary intricacy for self-regularization. Unlike ReLU, Mish activation does not get overwhelmed by a near-zero gradient. These properties allow the Mish activation to achieve better generalization[28].

3.1.4 3D reconstruction by decoder network

The Reconstruction networks are a crucial part of the CapsNet. In RCN, the authors tried to reduce the computation cost by deducting complexity from the reconstruction network. We believe that a proper reconstruction network is always suitable for a CapsNet based machine learning model. It is understood from the literature review that the reconstruction network is only used for training and is discarded or unnecessary during the process of testing or inference. Thereby allowing a fair comparison to the existing CNNs to the CapsNets. Here we employ the 3-dimensional reconstructions, and the reconstruction is based on the class independent decoder network for the RCN2. The DeepCaps inspired us to implement the decoder network with the instantiation parameters extracted from the model with deconvolutional layers to reproduce the input[26], [29].

3.1.5 Summary of the proposed architecture RCN2

In summary, as the image moves across the initial ResNet units, the primary capsules obtain the output from them at three different stages permitting the network to evaluate the input at different layers of feature mining. These primary capsules produce 8D vectors, for each being a highlight of the object. These 8D vectors go across a DigitCaps layer with three reiterations of routing to deliver 16-dimensional vectors, following which it is combined to deliver a classification. Moreover, the decoder network decodes each output of the network and tries to match it with the input received.

3.2 RCNX: Residual Capsule NeXt

Unlike the familiar Convolutional Neural Network (CNN) that depends on the shift-invariance in the image, Capsule Networks utilizes hierarchical model relations in depth. This uniqueness keeps them in the realm even with their vast volume with only equivalent precision to the CNNs. Capsules develop a sophisticated system to produce a routing by agreement, prominent to their volume and makes them exceptional. Previous improvements in RCN to RCN2 have aided in alleviating the complexity of size of the RCN.

We focus on revising RCN, to improve the accuracy while still reducing the size of the model and thus reiterating Capsule Network's prominence. In this section, the proposed Residual Capsule NeXt (RCNX) is projected as an active and improved architecture of RCN with a scale of 1.5 M parameters. By including modification in the initial layers of the RCN, with tremendous ResNeXt layers and modifying altogether structure. The final structure on the whole can be seen as an bottleneck version of RCN. The model produced an extraordinary progress in the system's accuracy on the CIFAR-10 dataset. This accuracy and size exceed the prominent embedded CNN model MobileNetV3. The architecture of RCNX is depicted in Figure 3.4.

3.2.1 ResNeXt Convolution Layers

To achieve excellent accuracy, convolutional layers of RCNX should become efficient in composite feature abstraction. Although Capsule Network functions are established on routed capsules, the primary levels are convolution reliant[26]. Repetitive convolutional layers in the opening stages of Capsule Network provide us superior feature mining, and this offers the neural network an excellent opening advantage.

The RCN ponders on eight repetitive ResNet layers with no presence of bottleneck layout or any cardinality to enhance the intricacy and lessen parameters. We bring substantial compression and development to the RCN architecture. As cited heretofore, this thesis intends revisions in such elements, which are deficient in the RCN system. RCNX is unique with the insertion of the latest dimension cardinality to RCN. As clarified in the literature

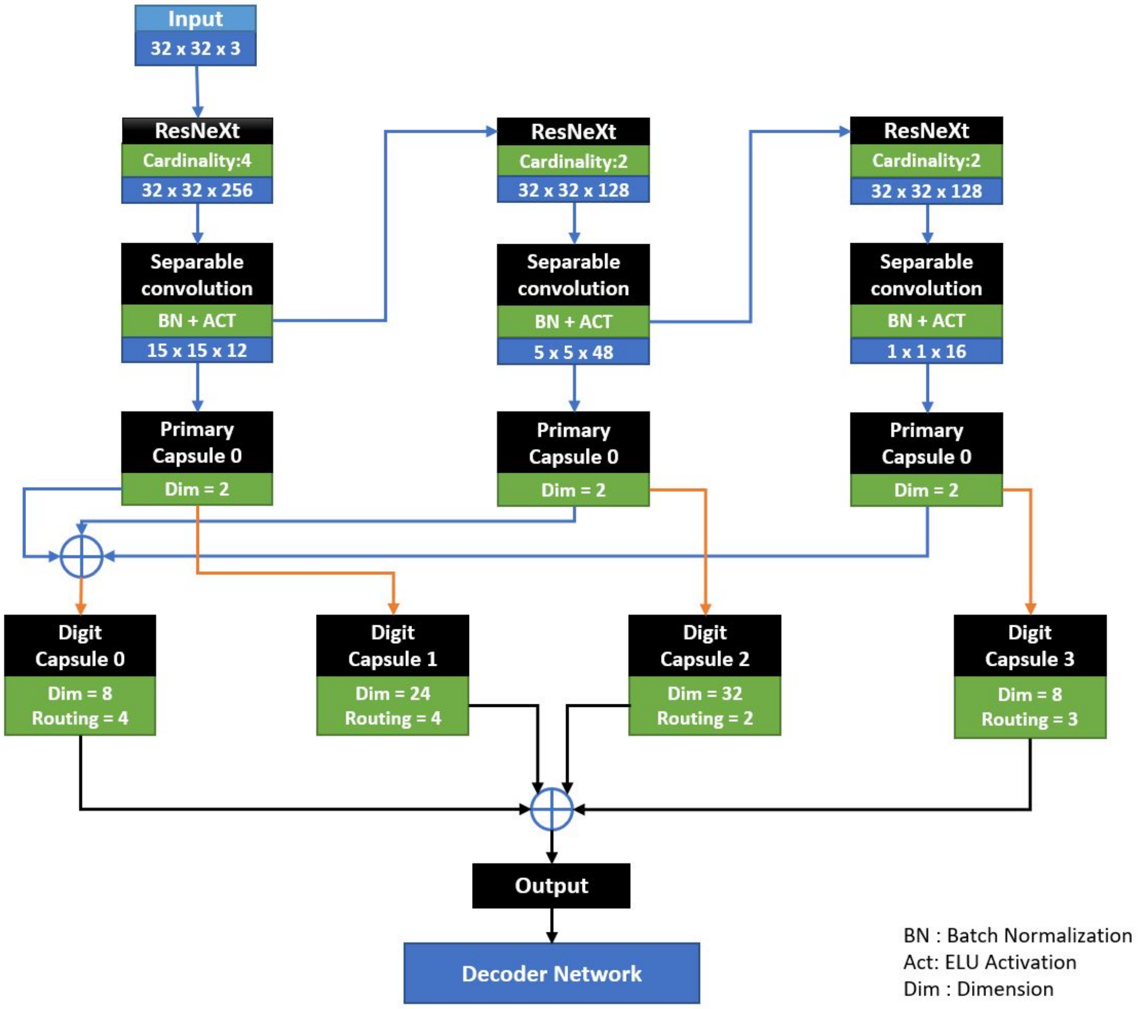


Figure 3.4. Architecture of proposed RCNX: Residual Capsule NeXt

review, the ResNeXt design (Figure 3.5) maintain the number of parameters but enhanced the accuracy of the RCNX's image classification capabilities.

With the absent structural improvements, RCN requires the intricacy to find a richer grasp of the inputs, which can be the justification for not attaining above average accuracy even with eight monotonous ResNets layers. From the time when we involve the requisite convolution applying ResNeXt, we excluded the unnecessary layers. ResNeXt structure inserted in the RCNX is with varying Cardinality prior to applying to the capsules. We also propose the three-staged architecture and find the system expands learning of the input for various capsules to understand from different image viewpoints.

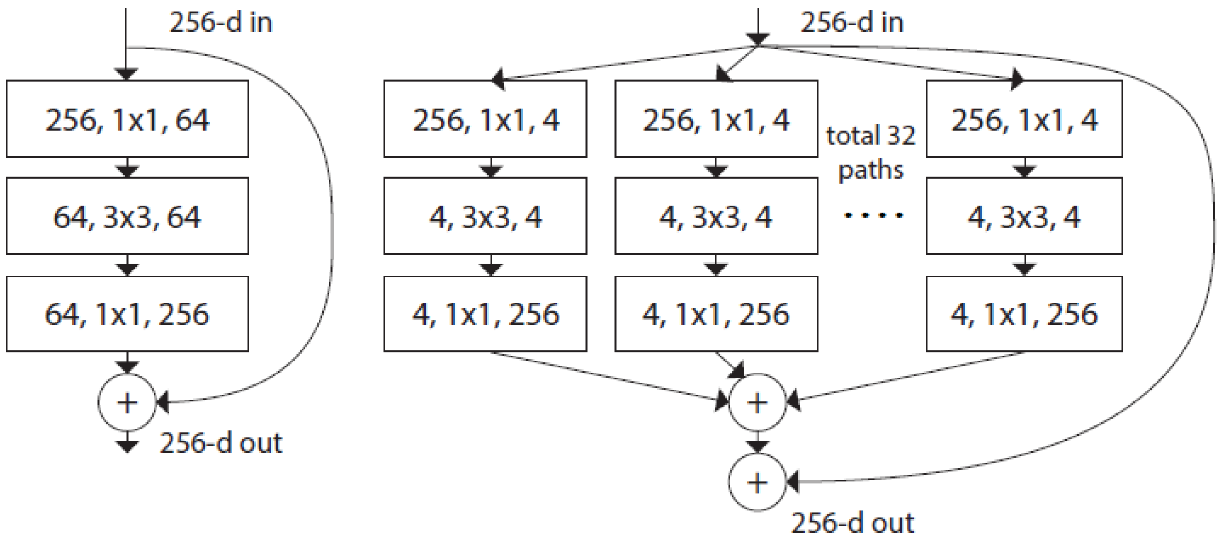


Figure 3.5. Left: ResNet; Right: ResNeXt with cardinality 32

3.2.2 3-Level ResNeXt structures in RCNX

The proposed RCNX can understand sophisticated elements and know to do it rapidly with the integration of different viewpoints. Letting the model go across each stage repeatedly because of the elaborate composition makes the best of RCNX with minimal effort.

In the 3-staged structure, the ResNeXt network with four and two cardinalities is incorporated, with identical size filters, i.e., 32. This differing Cardinality gives variable total filter sizes for individual capsules. Initial capsules getting three separate viewpoints to the input were configured to deliver likelihood vectors of variable dimensions, with plasticity in creating different features with various dimensions. We configured this network to a 3-level structure to improve the performance. We use the dimensions of $8, 24, 32$, and 8 across four primary capsules.

3.2.3 Effective Capsules with 3D convolution-based dynamic routing.

As examined earlier, the initial layers of the CapsNet are convolutional. The output of these convolution-based layers is fed into primary capsules. Allowing the training network to go through each level repetitively due to the intricate structure brings the best of RCNX with minimal effort. Utilizing 3D convolution-based dynamic routing, DeepCaps creators altered and made strides in routing by agreement calculation by changing including convolution for each capsule in the capsule networks[26].

The primary capsules transform the input vectors from the local feature detectors to eight-dimensional capsules as output. These are then fed into Digit capsules. Going deeper with a model is always what is done to increase performance in many CNN models. CapsNet's dynamic routing is such that they flatten the primary capsules' output vectors, and then they are routed with the dynamic routing algorithm. This structure gives rise to a structure similar to MLP models, which is a highly time-consuming structure. Stacking these to reach better performance will drastically pull the network's speed and efficiency[26].

This 3D convolution makes a difference in trimming the network in size. The inclusion of convolution is considering that neighboring neurons create a comparable instantiation output, and this can be clustered[26]. To accumulate multiple layers of CapsNet, it is necessary to establish a dynamic routing with some effect of a convolution-like process. This convolution-like process is achieved by the 3D-convolution based dynamic routing algorithm proposed by the DeepCaps. Within the Digit capsule layer, vector feature maps are squashed and steered through an energetic routing calculation. The new dynamic routing algorithm from DeepCaps massively dropped the redundancy by directing squares of capsule s from layer L to layer $L + 1$, rather than directing individual capsules in layer L separately[29].

3.2.4 3D reconstruction by decoder network

At the end of RCN, the decoder network does not include full inverse rendering but is reduced to a 2-dimensional restoration. The Reconstruction networks are a critical part of the CapsNet. In RCN, the authors tried to cut the computation cost by deducting complexity from the reconstruction network. We believe that a proper reconstruction network is always suitable for a CapsNet based machine learning model. It is understood from the literature review that the reconstruction network is only used for training and is discarded or unnecessary during the process of testing/inference. Thereby allowing a fair comparison to the existing CNNs to the CapsNets. Here we employ the 3-dimensional reconstructions, and the reconstruction is based on the class independent decoder network for the RCNX. The DeepCaps inspired us to implement the decoder network with the instantiation parameters extracted from the model with deconvolutional layers to reproduce the input[29].

3.2.5 ELU activation function

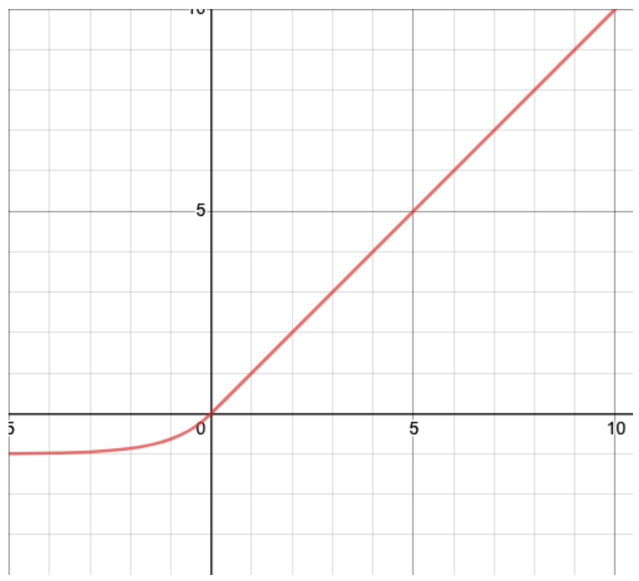


Figure 3.6. ELU Activation function[30]

Non-linearity is a significant part of the neural network, and without it, the NN will fail to achieve anything significant. The activation functions mainly contribute to this nonlinearity for the NN involved in the neurons. Over the past several years, activation functions alone have become a scope for further research. After each convolutional layer, speaking in a layer-like manner, we add the activation layer on it, which is usually followed by a batch normalization. In the baseline RCN, they use the ReLU activation function. In the RCN2, we used the Mish activation function, but for the proposed RCNX, we utilize the ELU (Exponential Linear Unit) activation function (Figure 3.6)[30]. ELU activation function avoids the dead ReLU problem. It also brings the better optimization of biases and weights and provides a negative output to prevent saturation near-zero gradient. Function representation of ELU is as follows[30].

$$f(x) = \left\{ \begin{matrix} x & x > 0 \\ \alpha.(\mathrm{e}^{x} - 1) & x <= 0 \end{matrix} \right\} \tag{3.3}$$

where α is also a trainable parameter.

3.2.6 Summary of the proposed architecture RCNX

In conclusion, when a $32 \times 32 \times 3$ input image is fed through the proposed RCNX: Residual Capsule NeXt the RGB channels of the images move through the convolutional layers and bring the channels of various output width and height sizes, giving the primary capsules a chance to see different sizes of the input image. The obtained in-depth features with channels of volume $c \times f$, where c is the Cardinality and f being the filter width, pass through a separable convolution before proceeding to primary capsules. These networks generate channels of 256 and 128. The separable convolution adjustments the image height and filter sizes to 3 different sizes, they are $15 \times 15 \times 12$, $5 \times 5 \times 48$, and $1 \times 1 \times 16$.

The digit capsules are of variable sizes yet lengthier dimensions than Primary Capsules with enhanced routing numbers and finally combine to form an output. Further, we decode output to form an inverse rendering effect by the decoder network to the equivalent input. This decoder network only activates during training and is removed while testing the proposed RCNX.

4. TRAINING SETUP

4.1 Hardware Setup

For the training and inference of the algorithms I utilized Lenovo think system compute node provided by the IU compute cluster[11]. These systems have the following hardware configurations.

Table 4.1. Hardware specifications.

CPU	Intel Xenon Gold
RAM	128 GB
GPU	NVIDIA Tesla V100

4.2 Dataset: CIFAR-10

CIFAR-10(Canadian Institute for Advanced Research) dataset is used to gain inference of the model[8]. The CIFAR-10 is a benchmark dataset comprised of 60,000 RGB images of 10 classes with 6000 pictures per class. Ten thousand images in these are for testing and the remainder for training the neural network models[8].

4.3 Hyper-parameter Tuning: Neural Network Intelligence

When challenged with the design parameters for a neural network, it is optimal to utilize a hyper parameter tuner. Earlier on the common tool used for hyper parameter optimization was TensorFlow hyper-parameter library, along with visualization tool TensorBoard[31]. We utilized TensorBoard and TensorFlow hyper-parameter tuners for the RCN2 development. Recent developments in the field of machine learning brought forth the amazing tool Neural Network Intelligence (NNI)[21]. Many parameters like optimizer, number of filters in the layers, activation functions, kernel sizes and routing numbers are experimentally optimized with the help of NNI.

5. RESULTS

Utilizing above mentioned training setup, RCN2 and RCNX were trained and tested. Using NNI, hyperparameter tuning was conducted, and a sample graph output of the hyperparameter tuning for RCNX is given in Figure 5.1.

We evaluated the proposed models RCN2 and RCNX with the CIFAR-10 dataset and compared the observed performance to the baseline RCN. We also brought in the accuracy comparisons of DCNet++, DCNet, 3-level RCN, seven ensemble CapsNet since these are leading Capsule Networks[9]. Furthermore, we compare our proposed RCN2 and RCNX to the embedded model MobileNetV3 to prove that the aim of creating an embedded machine learning model by compressing RCN without losing performance is achieved. The comparison is listed in Table 5.1.

Table 5.1. Model performance on CIFAR-10[9], [32]

Model	**No. of Parameters**	**Test Accuracy**
Proposed **RCNX**	1.58 M	89.31%
Proposed **RCN2**	1.95 M	85.12%
Baseline RCN	11.86 M	84.16%
3-Level RCN	10.8 M	86.42%
Seven-ensemble CapsNet	101.5M	89.4%
DCNet	11.8 M	82.63%
DCNet++	13.4 M	89.71%
MobileNetV3	1.8 M	88.93%

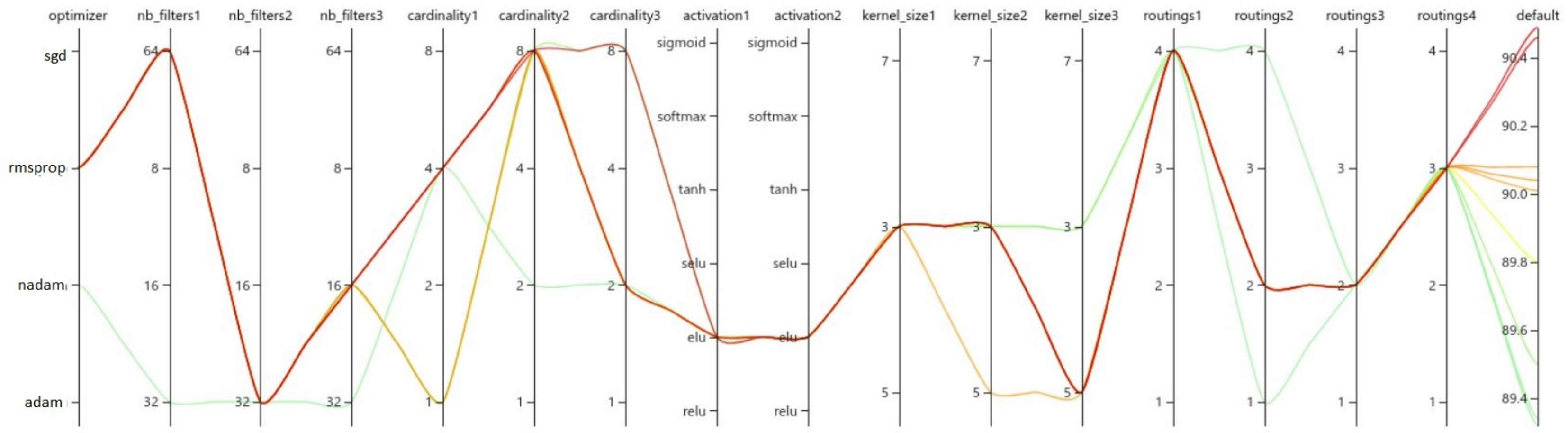


Figure 5.1. Hyperparameter Tuning achieved using NNI

5.1 RCN2 Results

With compression in the RCN, we achieved a balance between reductions in parameters while maintaining accuracy. The Figure 5.2 shows the train and test accuracy for each epoch for 30 epochs.

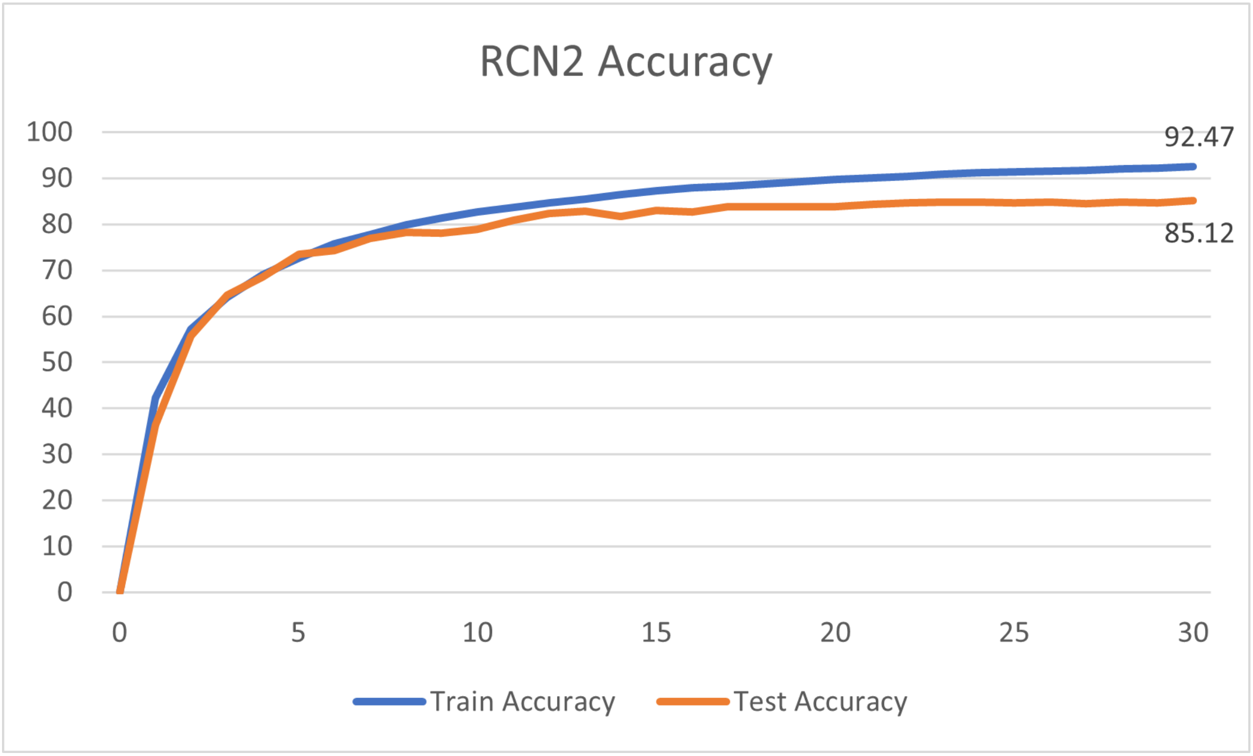


Figure 5.2. Accuracy plot while training the proposed RCNX

5.2 RCNX Results

With further improvements to the RCN2, we achieve the RCNX architecture, which is efficient and compressed. This RCNX outperformed the baseline model by 5.15% while reducing parameters by 86.67%. The train and test accuracy, while the model is trained for 30 epochs, is shown in Figure 5.3. The test loss curve is also depicted in Figure 5.4.

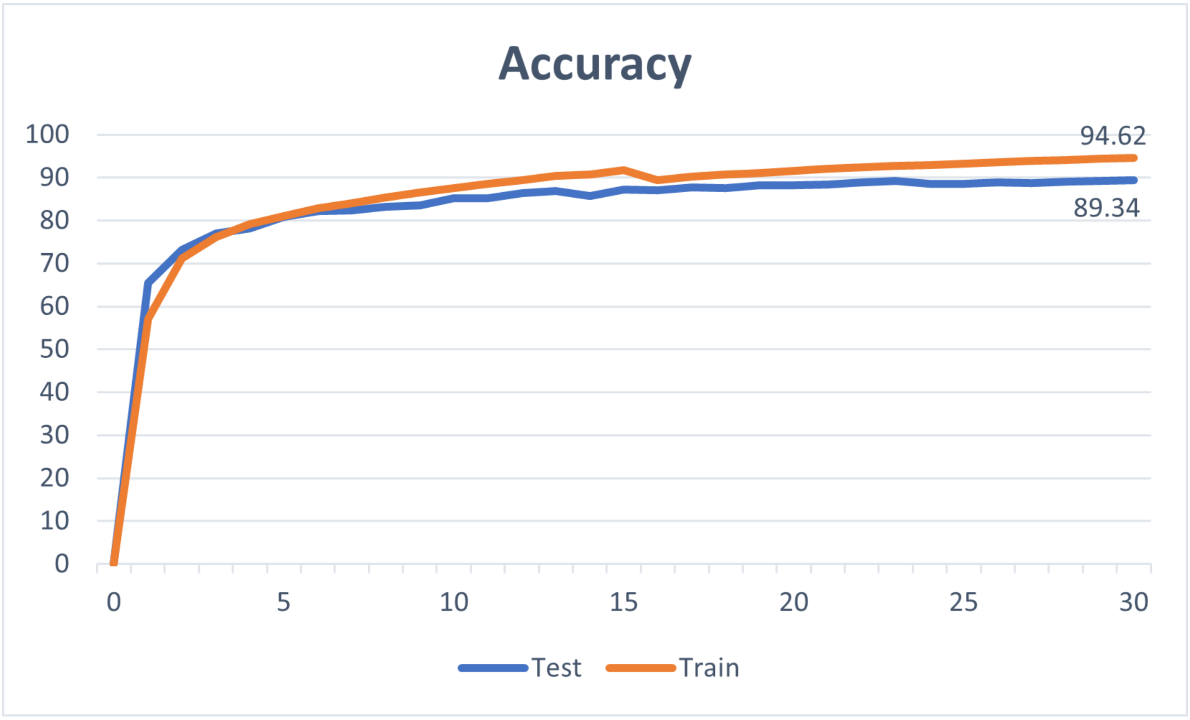


Figure 5.3. Accuracy plot while training the proposed RCN2

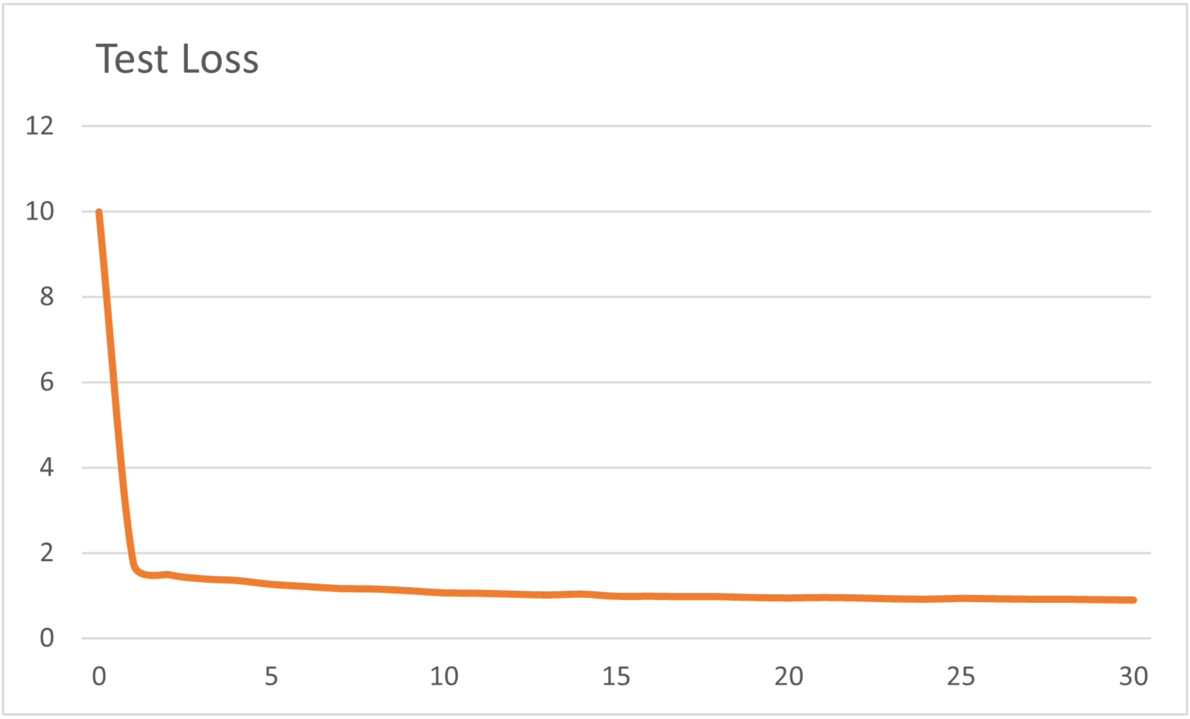


Figure 5.4. Loss plot while training the proposed RCN2

5.3 Performance Comparison

From the above results and comparisons, a performance comparison chart is generated for visualizing the impact of our proposed RCN2 and RCNX in Figure 5.5.

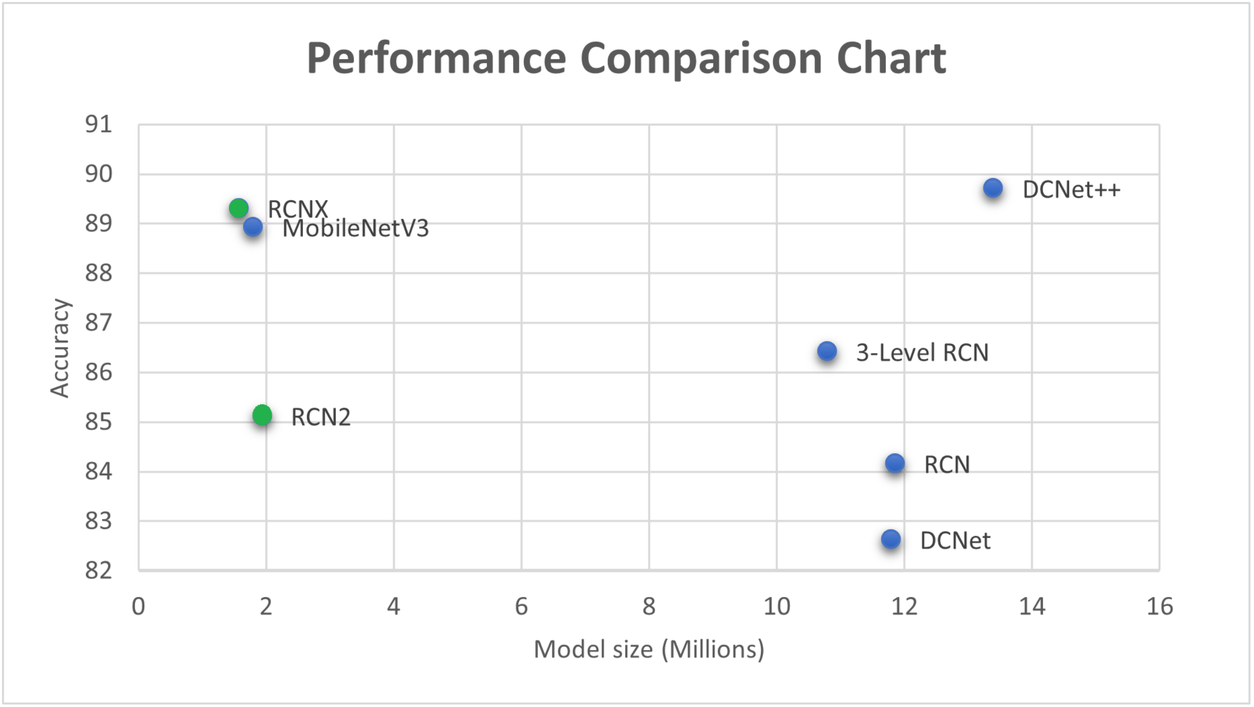


Figure 5.5. Performance Comparison Chart

It is clear from the above Figure 5.5 that RCN2 and RCNX have achieved performance better than the existing MobileNetV3, which is a comparison to CNN and outperforms all other CapsNets given above. Hence, we clearly reinstate the importance of the Capsule Networks in the field of Machine Learning.

6. IMPLEMENTATION ON I.MX RT1060

This section we discuss the implementation of RCNX architecture on the NXP i.MX RT1060 for image classification

6.1 Hardware Setup

The RCNX model achieved good accuracy of 89.31% with a model size of 1.58 Million parameters. This model has a deploy-able size for embedded hardware. We used i.MX RT 1060 Evaluation board which is developed by NXP semiconductors for image classification application (Figure 6.1)[33].

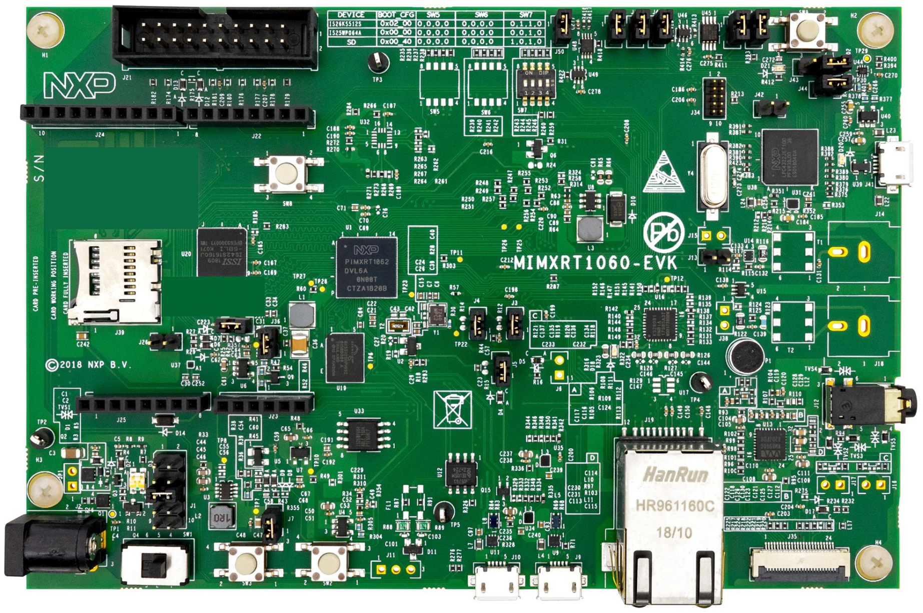


Figure 6.1. i.MX RT1060[33]

Additional hardware required for this process include a Camera Module MT9M114 (Figure 6.2) and an LCD screen[33]. Finally, the camera's input is processed frame by frame, and the detection result is sent via UART communication.

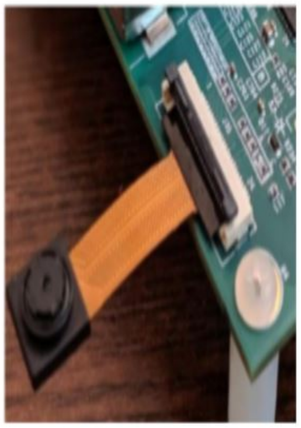


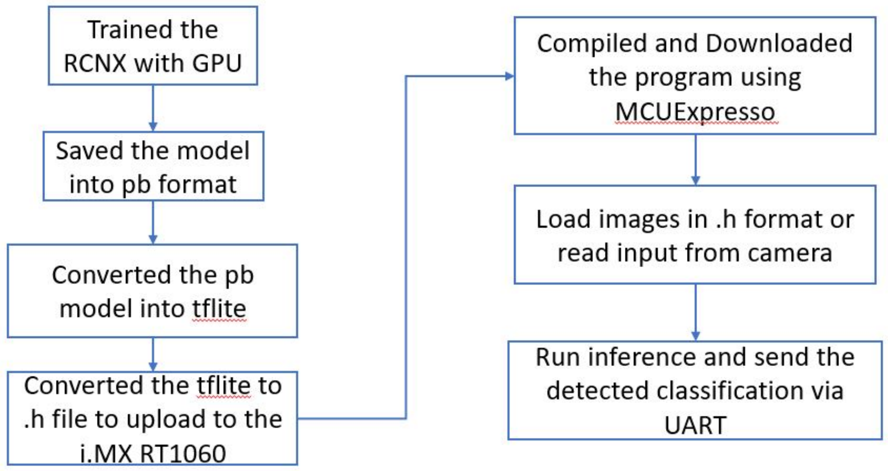
Figure 6.2. Camera module attached on IMX RT1060[33]

6.2 Softare Setup

MCUExpresso is used as the IDE for the deployment. NXP provides an eIQ software library intended for the software development for neural network-based applications and contains various optimized libraries for the compilation, back propagation, and inference generation. This library utilizes the Tensorflow-lite model of the intended architecture[34]. The eIQ software is given as middleware in the MCUXpresso SDK for the NXPi.MX RT1060 board[33]. It contains the latest eIQ SDK and demos. The tflite has to be converted into a C array structure to be downloaded to the board.

6.3 Model Preparation

After successfully training the RCNX with the CIFAR-10 dataset, we saved this model into PB format, the protocol buffer(protobuf) format. This PB format is commonly used in TensorFlow models. Since the development of RCNX is based on Keras, TensorFlow, and custom functions, it was easily convertible to the PB model[35], [36]. This PB model is further converted into a tflite(TensorFlow lite) model, a common standard used from converting models cross-platform[34]. The steps followed are as per the following Figure 6.3.



Trained the RCNX with GPU
Saved the model into pb format
Converted the pb model into tflite
Converted the tflite to .h file to upload to the i.MX RT1060
Compiled and Downloaded the program using MCUExpresso
Load images in .h format or read input from camera
Run inference and send the detected classification via UART

Figure 6.3. Course of action for deploying RCNX on IMX RT1060

6.4 Implementation Results

The RCNX was successfully deployed into the i.MX RT1060, proving that the CapsNet for image classification can be downloaded into embedded systems. A demonstration of the model running on i.MX RT1060 is provided in Figure 6.4. Although the model ran successfully and the accuracy of prediction was good, the time taken for inference is 3.8 seconds, which is a long time. This delay in processing could be due to the complex routing algorithms and current libraries' inefficiency to help optimize CapsNet based networks. We believe this prototype presented in the thesis is only an initial step for the CapsNet to be considered for embedded application.

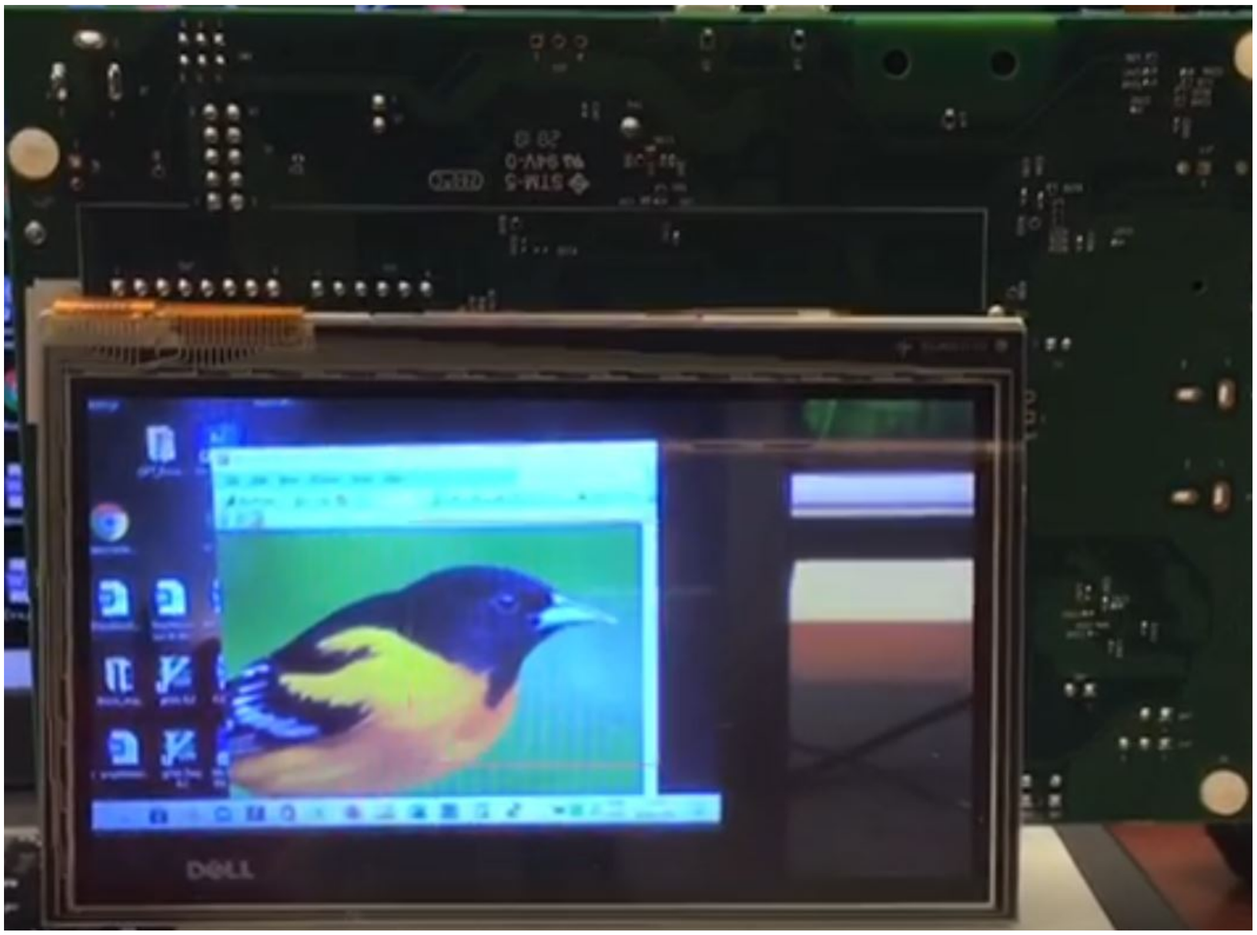
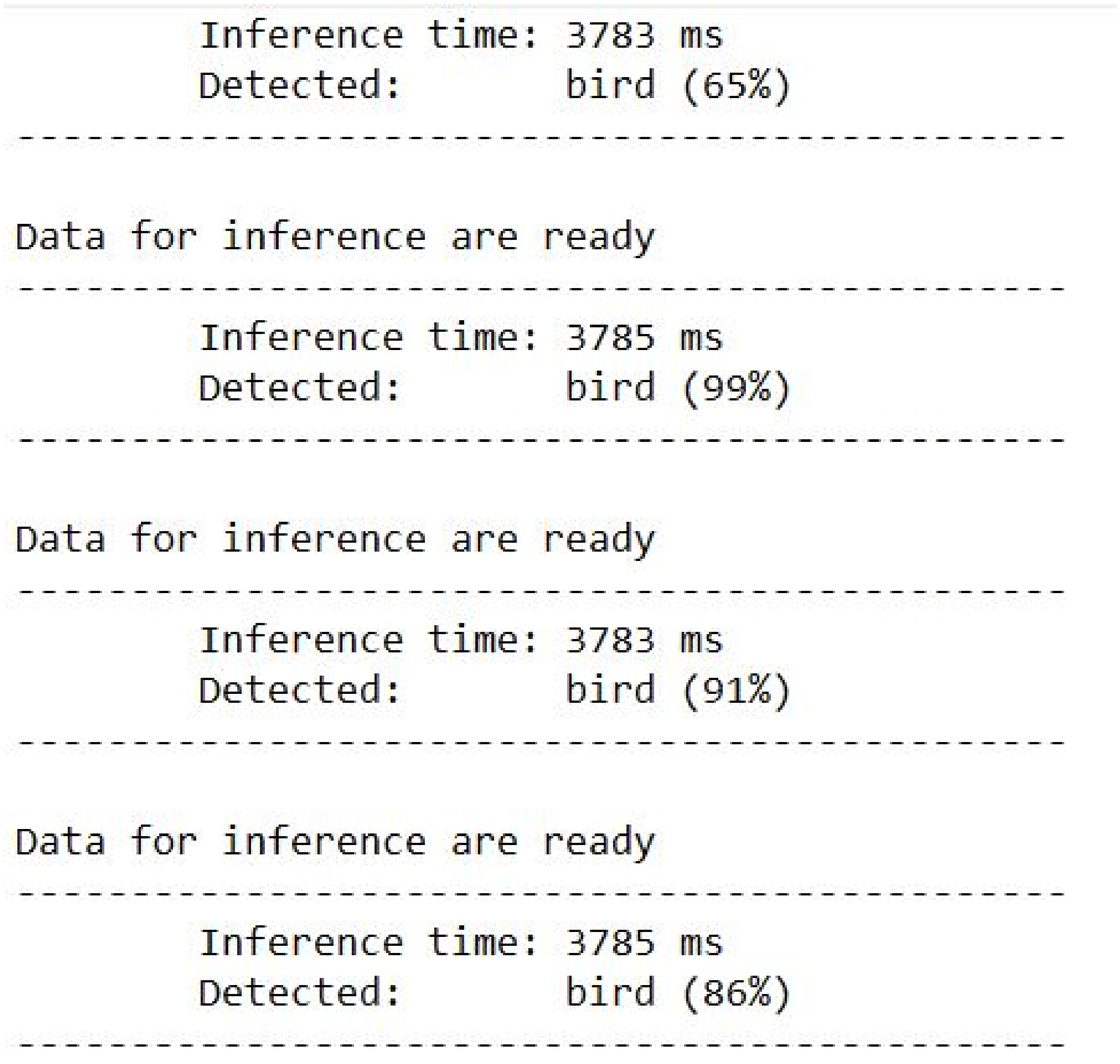


Figure 6.4. Execution after download



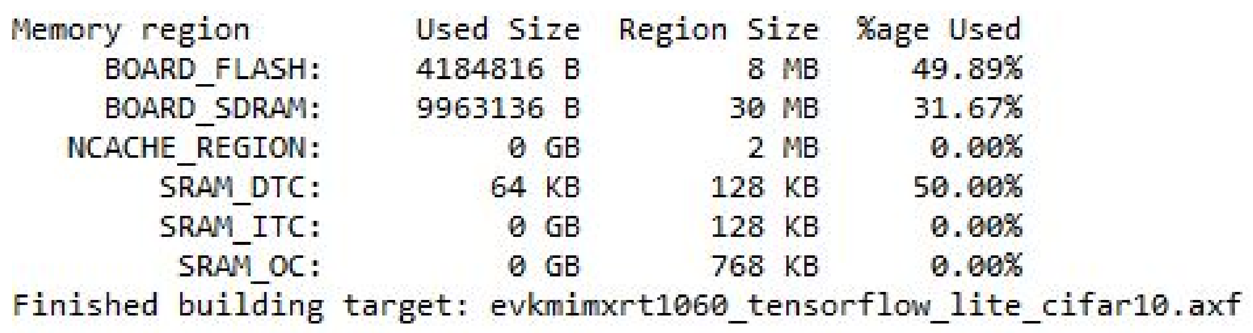
```
        Inference time: 3783 ms
        Detected:       bird (65%)
----------------------------------------------

Data for inference are ready
----------------------------------------------
        Inference time: 3785 ms
        Detected:       bird (99%)
----------------------------------------------

Data for inference are ready
----------------------------------------------
        Inference time: 3783 ms
        Detected:       bird (91%)
----------------------------------------------

Data for inference are ready
----------------------------------------------
        Inference time: 3785 ms
        Detected:       bird (86%)
----------------------------------------------
```

Figure 6.5. UART output of above image



```
Memory region         Used Size  Region Size  %age Used
     BOARD_FLASH:     4184816 B         8 MB     49.89%
     BOARD_SDRAM:     9963136 B        30 MB     31.67%
   NCACHE_REGION:          0 GB         2 MB      0.00%
        SRAM_DTC:         64 KB       128 KB     50.00%
        SRAM_ITC:          0 GB       128 KB      0.00%
         SRAM_OC:          0 GB       768 KB      0.00%
Finished building target: evkmimxrt1060_tensorflow_lite_cifar10.axf
```

Figure 6.6. Software size after build

7. CONCLUSION

Convolutional Neural Networks is one of the causes why Deep Learning is so prevalent. Constructing a deeper network always improves performance, but the vanishing gradient problem does not allow us to stack layers. This problem has been mitigated with the entrance Residual Network. Although CNN's are impressive and have provided many advancements towards machine learning, they are not impeccable. Problems such as the requirement of large datasets, pooling layers that cause loss of valuable information, and lack of hierarchical perception of objects contribute to the creation of CapsNet. CapsNet utilizes vector output capsules to enhance the primitive scalar feature detectors.

Residual Capsule Networks brought the best features from both worlds together. Though inspiring, this initial attempt is ineffective. This thesis uses the RCN baseline framework and improves the parts where we found lacking structural accuracies. Starting with the redundant layers, which do not get the required nonlinearity, to the inaccuracy in the implementation of reconstruction network is attended. We replace many crucial elements like Capsules Routing-by-Agreement algorithms, Redundant ResNet layers, and lack of complexity in the initial convolutions.

This thesis proposes RCN2: Residual Capsule Network V2 and RCNX: Residual Capsule NeXt to achieve superior accuracy with fewer parameters. This attempt was to lead to an architecture similar to the CNNs and thus improve Capsule Networks' field. The proposed models compressed and combined features from the RCN and the 3-Level RCN.

The proposed architectures were trained and tested with CIFAR-10 dataset. RCN2 achieved an accuracy of 85.12 % with only parameters of 1.95 M. Furthermore, the RCNX demonstrated superior accuracy of 89.31% with only parameters of 1.58 M. Compared to the previous models RCN and 3 Level RCN, the proposed models have achieved higher level of compression. We conclude that the RCN2 and RCNX are the models that shows the capability of Capsule Networks to be implemented to a size, which makes it easier to deploy in embedded systems.

In the near future, we believe these models will inspire more effective neural networks and also deliver accuracy superior to CNN models.

REFERENCES

[1] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, L. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *arXiv:1512.07108 [cs]*, Apr. 5, 2021. arXiv: 1512.07108. [Online]. Available: http://arxiv.org/abs/1512.07108.

[2] V. Dragoi. (Oct. 7, 2020). "Visual processing: Cortical pathways (Section 2, Chapter 15) Neuroscience Online: An electronic textbook for the neurosciences | Department of Neurobiology and Anatomy - The University of Texas Medical School at Houston," [Online]. Available: https://nba.uth.tmc.edu/neuroscience/m/s2/chapter15.html. (accessed: 04.05.2021).

[3] P. F. Sharp and R. Philips, "Physiological optics," in *The Perception of Visual Information*, W. R. Hendee and P. N. T. Wells, Eds., New York, NY: Springer, Jan. 11, 2021, pp. 1–32, ISBN: 978-1-4612-1836-4. DOI: 10.1007/978-1-4612-1836-4_1. [Online]. Available: https://doi.org/10.1007/978-1-4612-1836-4_1.

[4] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, CA, USA., p. 11. [Online]. Available: https://papers.nips.cc/paper/2017/file/2cad8fa47bbef282badbb8de5374b894-Paper.pdf.

[5] K. Bai. (Feb. 11, 2019). "A comprehensive introduction to different types of convolutions in deep learning," Medium, [Online]. Available: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215. (accessed: 04.05.2021).

[6] H. H. Tan and K. H. Lim, "Vanishing gradient mitigation with deep learning neural network optimization," in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, Sarawak, Malaysia, Malaysia: IEEE, Jun. 2019, pp. 1–4, ISBN: 978-1-72811-557-3. DOI: 10.1109/ICSCC.2019.8843652. [Online]. Available: https://ieeexplore.ieee.org/document/8843652/.

[7] S. B. S. Bhamidi and M. El-Sharkawy, "Residual capsule network," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, Apr. 22, 2021, pp. 0557–0560. DOI: 10.1109/UEMCON47517.2019.8993019.

[8] R. C. Calik and M. F. Demirci, "Cifar-10 image classification with convolutional neural networks for embedded systems," in *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, Aqaba: IEEE, Oct. 2018, pp. 1–2, ISBN: 978-1-5386-9120-5. DOI: 10.1109/AICCSA.2018.8612873. [Online]. Available: https://ieeexplore.ieee.org/document/8612873/.

[9] S. B. S. Bhamidi and M. El-Sharkawy, “3-level residual capsule network for complex datasets,” in *2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS)*, ISSN: 2473-4667, Apr. 22, 2021, pp. 1–4. DOI: 10.1109/LASCAS45839.2020.9068990.

[10] G. E. Hinton, S. Sabour, and N. Frosst, “Matrix capsules with EM routing,” presented at the International Conference on Learning Representations, Feb. 15, 2018. [Online]. Available: https://openreview.net/forum?id=HJWLfGWRb.

[11] C. A. Stewart, V. Welch, B. Plale, G. Fox, M. Pierce, and T. Sterling, “Indiana university pervasive technology institute,” Sep. 1, 2017. DOI: 10.5967/K8G44NGB. [Online]. Available: https://scholarworks.iu.edu/dspace/handle/2022/21675.

[12] J. Talukdar, A. Biswas, and S. Gupta, “Data augmentation on synthetic images for transfer learning using deep CNNs,” in *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida: IEEE, Jan. 27, 2021, pp. 215–219, ISBN: 978-1-5386-3045-7. DOI: 10.1109/SPIN.2018.8474209. [Online]. Available: https://ieeexplore.ieee.org/document/8474209/.

[13] J. Q. Fan, *English: Biological neural networks.* Dec. 24, 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Biological-neural-networks.jpg, (accessed: 04.05.2021).

[14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[15] Aphex34, *English: Typical CNN architecture*, Dec. 16, 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Typical_cnn.png, (accessed: 04.05.2021).

[16] J. Chao, T. Kishigami, K. Minowa, and S. Tsujii, “Artificial neural networks which can see geometric illusions in human vision,” in *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, vol. 3, Nagoya, Japan: IEEE, Apr. 5, 2021, pp. 2209–2212, ISBN: 978-0-7803-1421-4. DOI: 10.1109/IJCNN.1993.714165. [Online]. Available: http://ieeexplore.ieee.org/document/714165/.

[17] Aphex34, *English: Max_pooling with 2x2 filter and stride = 2*, Dec. 16, 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Max_pooling.png, (accessed: 04.05.2021).

[18] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, “Searching for MobileNetV3,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Apr. 5, 2021, pp. 1314–1324, ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00140. [Online]. Available: https://ieeexplore.ieee.org/document/9008835/.

[19] M. Pechyonkin. (Dec. 18, 2018). "Understanding hinton's capsule networks. part i: Intuition.," Medium, [Online]. Available: https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b. (accessed: 04.05.2021).

[20] M. Hassan. (Apr. 5, 2021). "ResNet (34, 50, 101): Residual CNNs for image classification tasks," [Online]. Available: https://neurohive.io/en/popular-networks/resnet/. (accessed: 04.05.2021).

[21] *Neural network intelligence (NNI)*, Apr. 22, 2021. [Online]. Available: https://github.com/microsoft/nni, (accessed: 04.22.2021).

[22] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv:1804.02767 [cs]*, Apr. 8, 2018. arXiv: 1804.02767. [Online]. Available: http://arxiv.org/abs/1804.02767.

[23] M. Ju, H. Luo, Z. Wang, B. Hui, and Z. Chang, "The application of improved YOLO v3 in multi-scale target detection," *Applied Sciences*, vol. 9, no. 18, p. 3775, Apr. 5, 2021. DOI: 10.3390/app9183775. [Online]. Available: https://www.mdpi.com/2076-3417/9/18/3775.

[24] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 5987–5995, ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.634. [Online]. Available: http://ieeexplore.ieee.org/document/8100117/.

[25] R. C. Calik and M. F. Demirci, "In embedded systems image classification with convolutional neural network," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, Turkey: IEEE, Apr. 5, 2021, pp. 1–4, ISBN: 978-1-5386-1501-0. DOI: 10.1109/SIU.2018.8404296. [Online]. Available: https://ieeexplore.ieee.org/document/8404296/.

[26] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne, and R. Rodrigo, "DeepCaps: Going deeper with capsule networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 2575-7075, Apr. 22, 2021, pp. 10 717–10 725. DOI: 10.1109/CVPR.2019.01098.

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, Apr. 5, 2021, ISBN: 978-1-60558-907-7.

[28] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv:1908.08681 [cs, stat]*, Aug. 13, 2020. arXiv: 1908.08681. [Online]. Available: http://arxiv.org/abs/1908.08681.

[29] J. Rajasegaran, *Brjathu/deepcaps*, original-date: 2019-03-15T07:42:55Z. [Online]. Available: https://github.com/brjathu/deepcaps, (accessed: 04.05.2021).

[30] Z. Qiumei, T. Dan, and W. Fenghua, "Improved convolutional neural network based on fast exponentially linear unit activation function," *IEEE Access*, vol. 7, pp. 151 359–151 367, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2948112. [Online]. Available: https://ieeexplore.ieee.org/document/8873678/.

[31] *TensorBoard*. [Online]. Available: https://www.tensorflow.org/tensorboard, (accessed: 04.05.2021).

[32] P. S. P. Kavyashree and M. El-Sharkawy, "Compressed MobileNet v3:a light weight variant for resource-constrained platforms," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, NV: IEEE, Jan. 27, 2021, pp. 0104–0107, ISBN: 978-1-66541-490-6. DOI: 10.1109/CCWC51732.2021.9376113. [Online]. Available: https://ieeexplore.ieee.org/document/9376113/.

[33] S. R. Desai, D. Sinha, and M. El-Sharkawy, "Image classification on NXP i.MX RT1060 using ultra-thin MobileNet DNN," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA: IEEE, Jan. 27, 2021, pp. 0474–0480, ISBN: 978-1-72813-783-4. DOI: 10.1109/CCWC47524.2020.9031165. [Online]. Available: https://ieeexplore.ieee.org/document/9031165/.

[34] *Tensorflow/tflite-support*, original-date: 2020-06-01T03:15:31Z. [Online]. Available: https://github.com/tensorflow/tflite-support, (accessed: 04.05.2021).

[35] *Keras-team/keras*, original-date: 2015-03-28T00:35:42Z. [Online]. Available: https://github.com/keras-team/keras, (accessed: 04.05.2021).

[36] *TensorFlow*, Jan. 27, 2021. [Online]. Available: https://www.tensorflow.org/, (accessed: 04.22.2021).