DESIGN SPACE EXPLORATION OF CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Prasham Shah

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2020

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS 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

Head of Graduate Program

Dedicated to

My Parents: Bela and Alpesh Shah,

My family, friends, colleagues and all well wishers.

ACKNOWLEDGMENTS

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

I am grateful to have been a member of IOT Collaboratory at IUPUI, and to have had the opportunity to work with so many talented people. The lab provided all the hardware and software, which was required for this research and an amazing workplace. I would like to thank my colleagues for their insights, it was great working with them. I would like to thank Maneesh Ayi for being a great friend and colleague.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABBREVIATIONS

A-MNASNET	Augmented MnasNet
AI	Artificial Intelligence
BLE	Bluetooth Low Energy
CNN	Convolutional Neural Networks
CV	Computer Vision
DL	Deep Learning
FC	Fully-Connected
FPGA	Field Programmable Gate Array
GPU	Graphical Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IP	Internet Protocol
LR	Learning Rate
R-MNASNET	Reduced MnasNet
ReLU	Rectified Linear Unit
RTMaps	Real-Time Multi Sensor Applications
SGD	Stochastic Gradient Descent
TCP	Transmission Control Protocol

ABSTRACT

Shah, Prasham. M.S.E.C.E., Purdue University, December 2020. Design Space Exploration of Convolutional Neural Networks for Image Classification. Major Professor: Mohamed El-Sharkawy.

Computer vision is a domain which deals with the goal of making technology as efficient as human vision. To achieve that goal, after decades of research, researchers have developed algorithms that are able to work efficiently on resource constrained hardware like mobile or embedded devices for computer vision applications. Due to their constant efforts, such devices have become capable for tasks like Image Classification, Object Detection, Object Recognition, Semantic Segmentation, and many other applications. Autonomous systems like self-driving cars, Drones and UAVs, are being successfully developed because of these advances in AI.

Deep Learning, a part of AI, is a specific domain of Machine Learning which focuses on developing algorithms for such applications. Deep Learning deals with tasks like extracting features from raw image data, replacing pipelines of specialized models with single end-to-end models, making models usable for multiple tasks with superior performance. A major focus is on techniques to detect and extract features which provide better context for inference about an image or video stream. A deep hierarchy of rich features can be learned and automatically extracted from images, provided by the multiple deep layers of CNN models.

CNNs are the backbone of Computer Vision. The reason that CNNs are the focus of attention for deep learning models is that they were specifically designed for image data. They are complicated but very effective in extracting features from an image or a video stream. After AlexNet won the ILSVRC in 2012, there was a drastic increase in research related with CNNs. Many state-of-the-art architectures

like VGG Net, GoogleNet, ResNet, Inception-v4, Inception-Resnet-v2, ShuffleNet, Xception, MobileNet, MobileNetV2, SqueezeNet, SqueezeNext and many more were introduced. The trend behind the research depicts an increase in the number of layers of CNN to make them more efficient but with that, the size of the model increased as well. This problem was fixed with the advent of new algorithms which resulted in a decrease in model size.

As a result, today we have CNN models, which are implemented on mobile devices. These mobile models are compact and have low latency, which in turn reduces the computational cost of the embedded system. This thesis resembles similar idea, it proposes two new CNN architectures, A-MnasNet and R-MnasNet, which have been derived from MnasNet by Design Space Exploration. These architectures outperform MnasNet in terms of model size and accuracy. They have been trained and tested on CIFAR-10 dataset. Furthermore, they were implemented on NXP Bluebox 2.0, an autonomous driving platform, for Image Classification.

1. INTRODUCTION

1.1 Context

Computer Vision is becoming an essential application in this modern world. With advances in technology autonomous cars, drones and UAVs, robots etc have been enabled with vision capabilities. These technologies use Convolutional Neural Networks to process the images or video input. They are used for applications like Image Classification, Object Detection, Semantic Segmentation etc. Convolutional Neural Networks are a part of Deep Learning, which is a subset of Machine Learning. Due to the advances in the field of AI, Machine Learning capabilities increased and this enabled a whole new field of Deep Learning. This field deals with creating, optimizing and implementing algorithms which enables technology to become self-reliant and gain human level precision. The prime goal is automation of these technologies in a way that they are able to operate perfectly without any human intervention.

1.2 Motivation

Deep Learning is the future of AI. Today, it has applications in almost all industrial sectors and is helping in creating a better world. Vision applications are one of the major breakthroughs which have been made possible because of this field. Top companies like Tesla, Google, Amazon, Microsoft etc are investing billions of dollars in this domain. Our lifestyle will change and so will the modern way of living. Modern technologies are being developed to make our lives easier, healthier and safer. Today, when the world is facing some serious challenges like climate change, health issues, increased crime rate etc, Deep learning has become a very essential tool to face and overcome these challenges.

It is used in developing healthcare technologies, which have almost human level precision and are used to save lives in hospitals. Medical imaging devices which detect even minute particles are used for diagnosis of various diseases. Autonomous cars which will make roads safer and will reduce fatal life-threatening accidents, Autonomous Drones which will revolutionize the logistics and will deliver packages more efficiently, UAVs which will aid militaries with surveillance and help prevent wars, smart cameras which will be able to recognize people and track their activities to reduce crimes. Smart imaging of atmosphere for weather prediction and warnings for natural calamities like tsunami, tornadoes etc. These are a few examples of how CNNs are making a major impact in our lives.

These technologies require computational power, speed, accuracy and precision. In order to make them efficient, the algorithms have to be fast having low latency, working efficiently on low power, being more accurate and consuming less memory. CNNs have so many layers so as the architectures become deeper and wider, giving more accuracy, their computational cost increases. These CNN models have to be more compact and should work as efficiently as the state-of-the-art architectures. After years of research, with the advent of new algorithms, now it is possible to make CNNs more efficient with a fair trade-off between model size and accuracy. This thesis aims to contribute towards this same goal, making CNNs compact in terms of model size and increasing its accuracy so that they can be used for such applications.

1.3 Challenges

- Implementing new algorithms on MnasNet (baseline architecture)
- Training from scratch
- Tuning hyperparameters
- Reducing model size
- Increasing model accuracy

- Implementation on NXP Bluebox 2.0 for Image Classification

1.4 Methodology

- Analyzing the baseline CNN architecture
- Implementing new algorithms
- Modifying the baseline CNN architecture
- Training new CNN architecture with CIFAR-10 dataset
- Tuning of hyperparameters
- Implementing Optimization techniques
- Implementing Data augmentation techniques
- Deploying new architecture on a hardware for specific application

1.5 Contributions

- Design Space Exploration of MnasNet Architecture
- Proposed A-MnasNet and R-MnasNet CNN Architectures
- Image Classification on NXP Bluebox 2.0 using A-MnasNet and R-MnasNet
- Published 2 research papers in IEEE Conferences (third paper accepted)
- Published 1 paper in Journal.

2. LITERATURE REVIEW

This section gives an insight on Convolutional Neural Networks. It will explain why CNNs are used for Deep Learning and why they are used for computer vision applications. This chapter also discusses the MnasNet CNN architecture, which was further improved by Design Space Exploration.

2.1 Convolutional Neural Networks

Convolutional Neural Networks are a special class of Neural Networks which mainly consist of Convolutional Layers, Pooling Layers, Activation Layers and Fully Connected Layers. They are used to extract features from an image or a video input. They are used for various computer vision applications like image classification, object detection, semantic segmentation, face recognition etc.

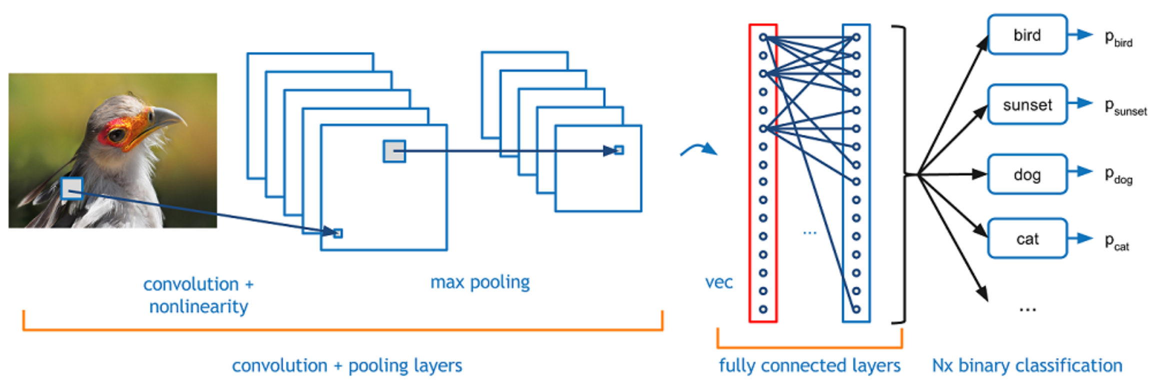

Fig. 2.1. Convolutional Neural Network

The Figure 2.1 shows an example of a convolutional neural network, which is taking an image input and then extracting features from it through various layers and then finally predicting the class of the object in the given image.

CNNs can be divided in two phases:

- Convolutional layers: extracts and learns features.
- Fully connected layers: prediction part

2.1.1 Convolutional Neural Networks and its significance

In fully connected neural networks, all neurons of a layer connect with all neurons of the next layer. They have so many connections that the complexity of the architecture increases by a tremendous amount. The computational cost of such networks is more because the parameters are more. It is not ideal for computer vision applications.

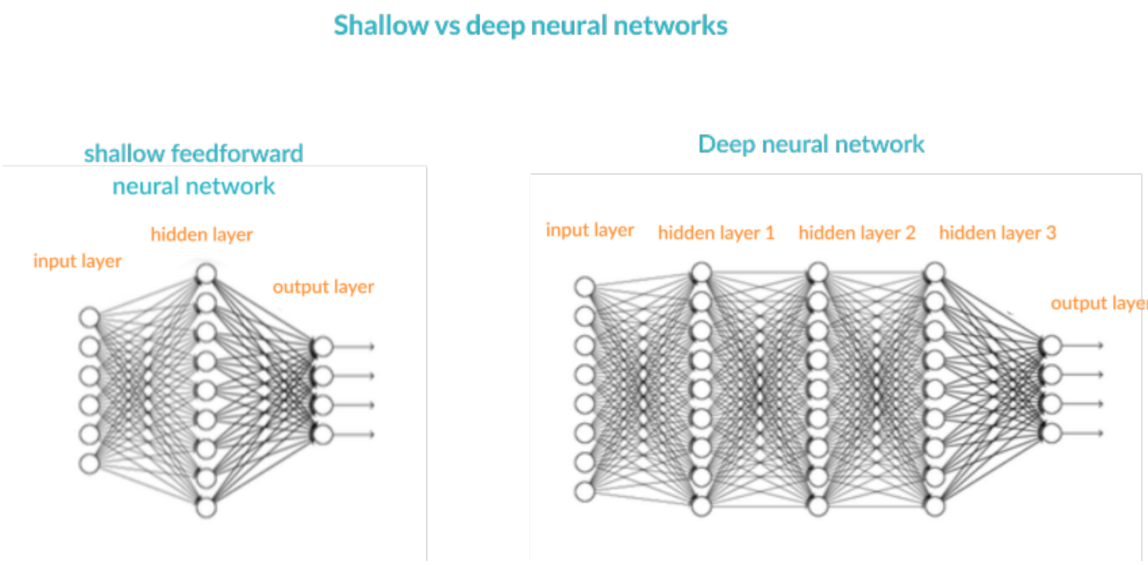

Fig. 2.2. Shallow vs Deep Neural Networks

For computer vision applications, classical neural networks were not as effective as CNNs. Input of the neural network have tremendous amount of data. When this data is as an input to fully connected neural networks, since all the neurons are connected with each other, the network parameters increase by a huge factor.

A CNN uses a way that an image is made out of more modest subtleties, or includes, and makes a system for dissecting each element in seclusion, which illuminates a choice about the picture all in all. As a component of the CNN, there is additionally a fully connected layer that takes the final product of the convolution/pooling cycle and arrives at a classification decision.

2.1.2 Input Layer

The input can be an image input or a video stream. Both are basically a collection of pixels which are placed in an array to form an image. Each pixel has a numeric value in the range of -255 to +255. The numbers represent the color value of the pixels in the image or video. The image or video stream is first converted to feature maps which consist of an array of numbers stacked together using NumPy library. It is illustrated in Figure 2.3.

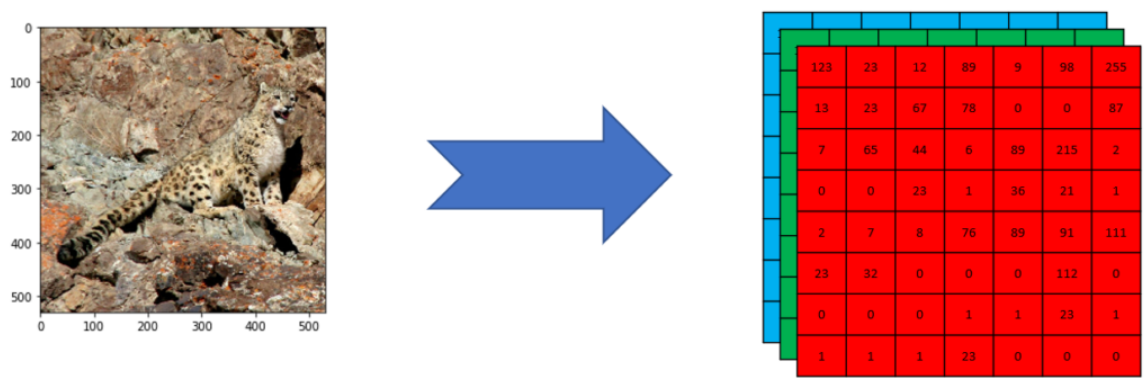

Fig. 2.3. Image Input

2.1.3 Convolutional Layers

Convolutional layers are significant building blocks which become the backbone of CNNs. A convolution is the straightforward use of a filter to an information that outcomes in an activation. Rehashed utilization of similar filter to an information brings about a map of activations called a feature map, showing the areas and quality of a distinguished feature in an information, for example, an image. The advancement of convolutional neural networks is the capacity to consequently get familiar with countless filters in equal explicit to a preparation dataset under the limitations of a particular prescient displaying issue, for example, image classification. The outcome is exceptionally explicit highlights that can be distinguished anyplace on input.

The CNN is a particular sort of neural network intended for two-dimensional data, despite the fact that they can be utilized with one-dimensional and three-dimensional information. Key to the CNN is convolutional layer, which gives the network its name. It performs an operation, which is known as “convolution”.

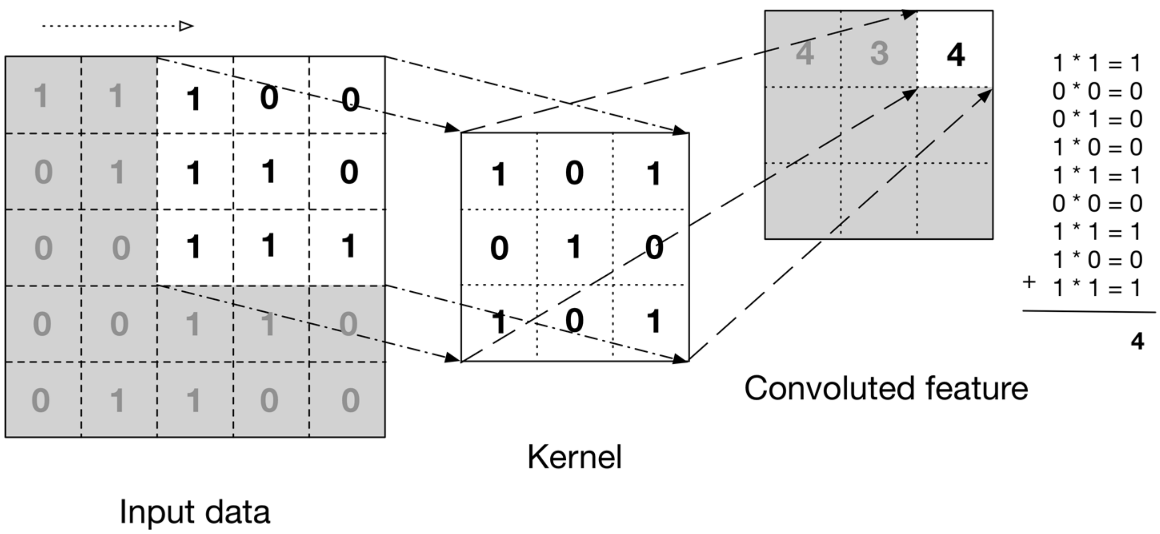

Fig. 2.4. Convolutional Layer

A convolution is a multiplication of weights with input, followed with an addition of bias. It is a linear operation. The array of weights known as filter or kernel slides on the input array performing convolution on each element. The result of each operation is added to the output array which is known as feature map. This process of extracting features from the image happens throughout the CNN’s convolutional layers. This process is illustrated in Figure 2.4.

The filter performs dot product or scalar multiplication with the elements of the input array. Since the size of output feature map depends on the filter size, the number of features extracted per layer can be changed by changing the size of filter. Usually, the size of filter is kept less than the size of input array. Utilizing a smaller filter is deliberate, as it permits a similar filter (set of weights) to be increased by the input array on various occasions at various points on the input. This efficient utilization of a similar filter over a picture is an influential thought. In the event that the filter is intended to distinguish a particular sort of highlight in the input, at that point the utilization of that filter methodically over the whole input picture permits the filter an occasion to find that include anyplace in the picture. This ability is ordinarily alluded to as interpretation invariance, for example the overall interest in whether the component is available as opposed to where it was available.

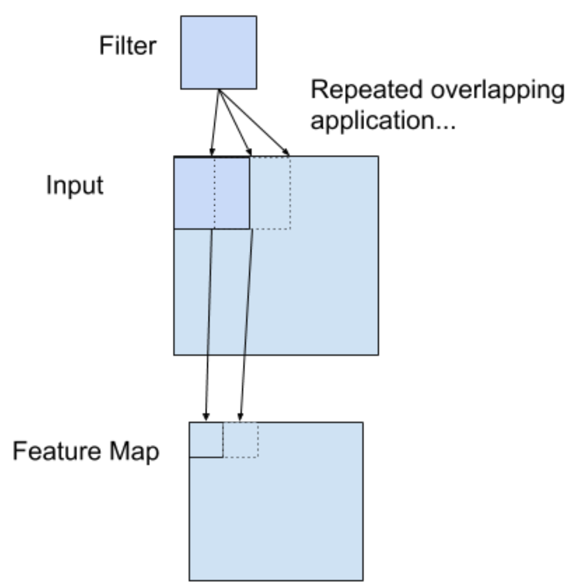

Fig. 2.5. Example of a filter to a Two Dimensional input to create Feature Map

The output of the multiplication operation between the filter and the input array is a solitary value. As the filter is applied on various occasions to the input array, the outcome speak to a filtering of the input. In that capacity, the two-dimensional yield array from this activity is known as a “feature map”. When a feature map is made, we can pass each value in the feature map through a non-linearity, for example, a ReLU, much as we accomplish for the yields of a completely associated layer.

In synopsis, we have an input, for example, image data, and we have a filter, which is a bunch of weights, and the filter is efficiently applied to the input data to make a feature map as shown in Figure 2.5.

2.1.4 Pooling Layers

The Pooling layer is accountable for lessening the spatial size of the Convolved Feature. This is to reduce the computational power needed to deal with the data through dimensionality decrease. Besides, it is helpful for separating predominant aspects which are rotational and positional invariant, in this way keeping up the cycle of successfully preparing of the model.

There are mainly two types of Pooling Layers in a CNN: Max Pooling and Average Pooling. The functionality of these two types of layers are demonstrated in Figure 2.6.Max Pooling restores the maximum value from the segment of the picture covered by the Kernel. Whereas, Average Pooling restores the average of the multitude of values from the bit of the picture covered by the Kernel. Max Pooling additionally proceeds as a Noise Suppressant. It disposes of the loud actuations out and out and furthermore performs de-noising alongside dimensionality decrease. Then again, Average Pooling just performs dimensionality decrease as a commotion stifling component. Subsequently, we can say that Max Pooling plays out significantly in a way that is better than Average Pooling.

The Convolutional Layer and the Pooling Layer, together structure the I-th layer of a Convolutional Neural Network. Contingent upon the complexities in the pictures, the quantity of such layers might be expanded for extracting low-level features significantly further, however at the expense of more computational power.

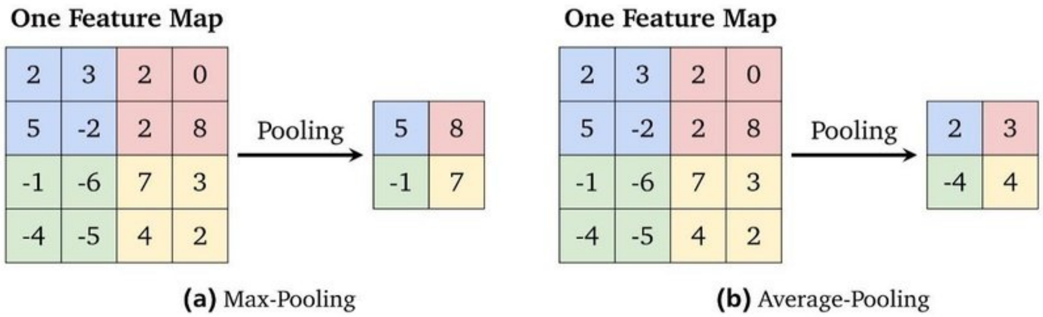

Fig. 2.6. Max Pooling and Average Pooling

2.1.5 Fully Connected Layers

Fully connected layers are a basic part of Convolutional Neural Networks (CNNs), which have been demonstrated fruitful in perceiving and arranging pictures for PC vision. The CNN cycle starts with convolution and pooling, separating the picture into features, and breaking down them autonomously. The consequence of this cycle takes care of into a fully connected neural organization structure that drives the last grouping choice.

The target of a fully connected layer is to take the consequences of the convolution/pooling cycle and use them to order the picture into a name (in a straightforward arrangement model). The yield of convolution/pooling is straightened into a solitary vector of values, each speaking to a likelihood that a specific feature has a place with a name. For instance, if the picture is of a feline, features speaking to things like hairs or hide ought to have high probabilities for the mark “cat”.

Figure 2.7 delineates how the input values stream into the main layer of neurons. They are increased by weights and pass through an enactment work (ordinarily ReLu), simply like in an exemplary counterfeit neural organization. They at that point pass forward to the yield layer, in which each neuron speaks to an classification label.

The fully connected portion of the CNN network experiences its own backpropagation cycle to decide the most precise weights. Every neuron gets weights that

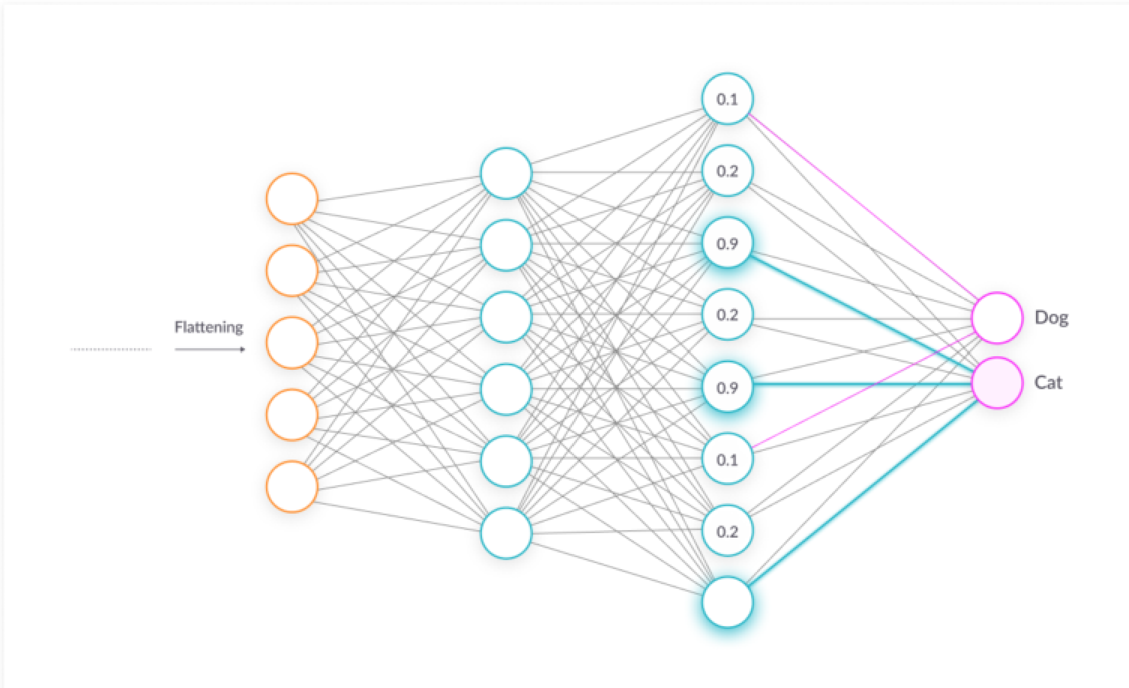

Fig. 2.7. Fully Connected Layer

organize the most proper label. At long last, the neurons "vote" on every one of the labels, and the victor of that vote is the classification choice.

2.1.6 Activation Layers

Activation layers are used to add non-linearity in the CNN. They determine the correct non-linear relation between the input and output signals. Different types of mathematical functions are used to add this property of non-linearity. Some commonly used activation functions like ReLU, Tanh, Sigmoid etc are represented in Figure 2.8

2.2 MnasNet (Baseline) Architecture

It is an arduous task to design, train, and evaluate convolutional neural networks for large datasets as it is time consuming and requires extensive domain knowledge.

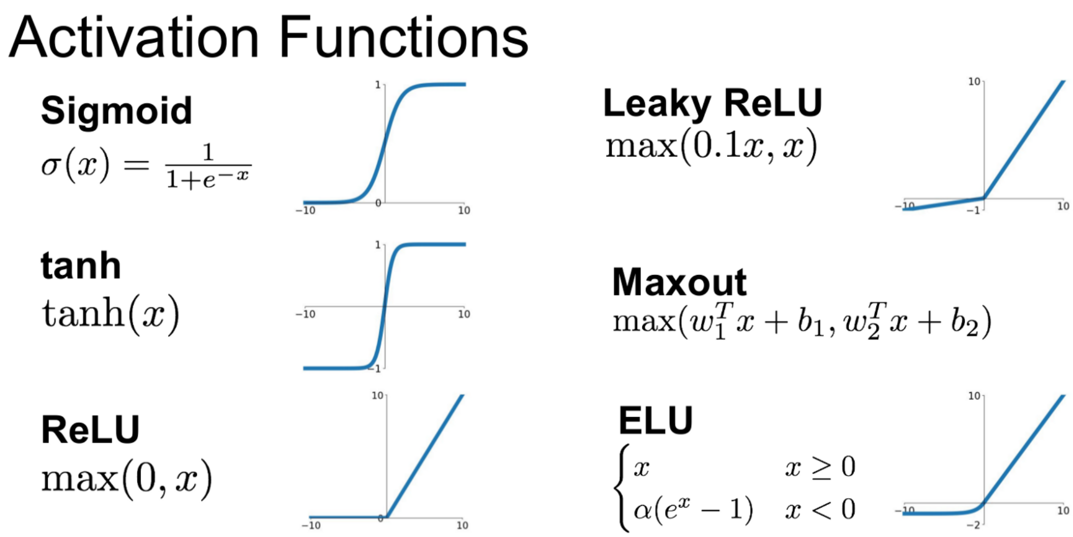

Fig. 2.8. Commonly used Activation Functions

To solve the problem of design a CNN model, the Google Brain team designed a model called NasNet (neural architecture search network) which searches a search space of possible convolution, pooling, and other blocks with variable strides, kernel sizes, and more. However, this model did not search for efficient models that can be run on mobile platforms. Thus, MnasNet was developed.

The following were the main contributions:

- The authors introduce latency information when evaluating models to discourage larger models with expensive operations. This leads to a good trade-off between accuracy and latency.

- On ImageNet classification task, MnasNet model achieves 74.% top-1 accuracy with 76ms latency on a Pixel phone.

- On the COCO object detection task, MnasNet achieves both higher mAP quality and lower latency than MobileNets.

They have introduced a neural architecture search approach, which optimized accuracy and latency on mobile devices using reinforcement learning. By using their automated approach, they propose various architectures called MnasNet-A1, MnasNet-A2 and MnasNet-A3. They show that diversity of layers in such resource-constrained models yield better trade-offs between accuracy and latency of the model. They have shown that their architecture outperforms other models like MobileNetV1, SqueezeNext, ShuffleNet, MobileNetV2, NASNet and many other models.

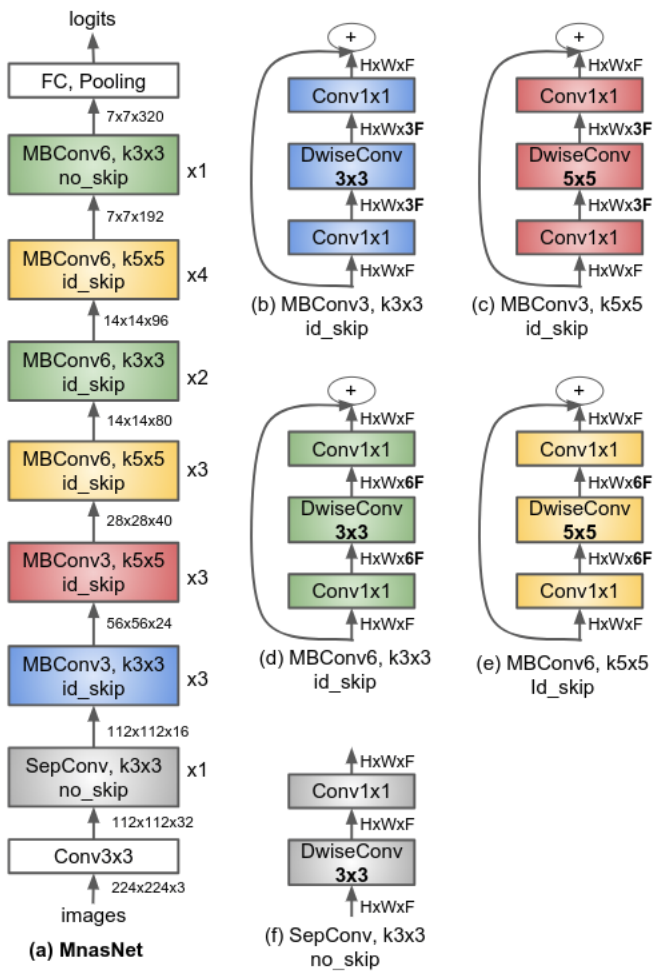

Fig. 2.9. MnasNet architecture

As shown in Figure 2.9, every block except one is of the same structure. The structure goes as follows:

Conv2D(1x1) - BatchNormalization - ReLU6 - DepthwiseConv2D - BatchNormalization - ReLU6 - Conv2D(1x1) - BatchNormalization - ReLU

Depending on the structure, the block may or may not of skip connections from input to output of the last layers. The SepConv layer just has DepthwiseConv2D, Conv2D(1x1), BatchNormalization, and finally ReLU6 activation layer.MnasNet uses Convolution Ops, depthwise separable convolution, mobile inverted bottleneck layers to extract features. It uses RMSProp optimizer, Batch Normalization and Dropout regularization.

Table 2.1 shows the MnasNet architecture which is trained with CIFAR-10 dataset where t: expansion factor, c: number of output channels, n: number of blocks and s: stride.

Table 2.1.
MnasNet Architecture

MnasNet Architecture					
Layers	Convolutions	t	c	n	s
$32^2 \times 3$	Conv2d 3×3	-	32	1	1
$112^2 \times 32$	SepConv 3×3	1	16	1	2
$112^2 \times 16$	MBConv3 3×3	3	24	3	2
$56^2 \times 24$	MBConv3 5×5	3	40	3	2
$28^2 \times 40$	MBConv6 5×5	6	80	3	2
$14^2 \times 80$	MBConv6 3×3	6	96	2	1
$14^2 \times 96$	MBConv6 5×5	6	192	4	1
$7^2 \times 192$	MBConv6 3×3	6	320	1	1
$7^2 \times 320$	FC, Pooling			10	

3. HARDWARE AND SOFTWARE

- NXP Bluebox 2.0
- Intel i9 9th generation processor with 32 GB RAM
- Aorus Geforce RTX 2080Ti GPU
- Python version 3.6.7.
- Pytorch version 1.0.
- Spyder version 3.6.
- RTMaps Studio
- Livelossplot

3.1 NXP Bluebox 2.0

BlueBox 2.0 [35] by NXP is a real-time development stage that gives the necessary presentation, functional security and car unwavering quality to build up oneself driving vehicles. It is an ASIL-B and ASIL-D agreeable hardware system, a coordinated bundle for making self-ruling applications, for example, ADAS systems, driver help systems. It is involved three autonomous systems on chip that are S32V234: vision processor, LS2084A: register processor, and S32R274: radar microcontroller.

It utilizes one of the Cortex-A72 layers cape processors out of the 8 processors and an inserted vision chip S32V234. It incorporates Level 1 conveying crash admonitions, programmed slows down and keeping up a set vehicle good ways from others. Level 2 innovation execution of vehicle directing, brake, and quicken naturally inside restricted conditions and requirements, not dispensing with the need of a human

Fig. 3.1. NXP BlueBox 2.0

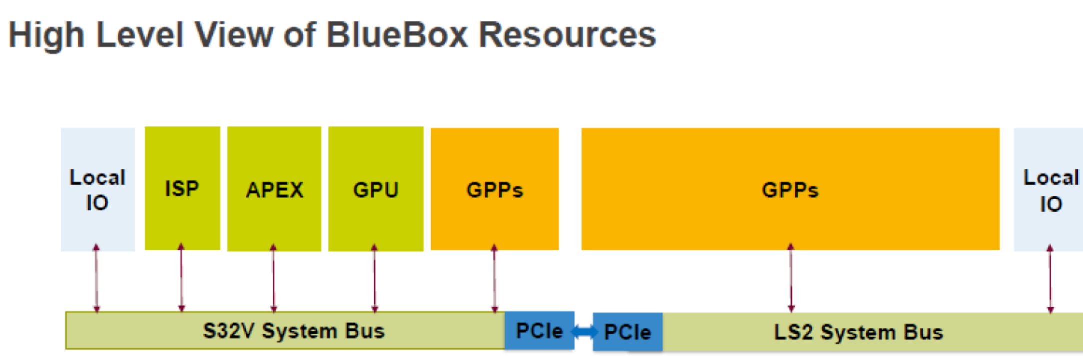

Fig. 3.2. High level view of NXP Bluebox 2.0

driver. Level 3 independent applications, for example, the moving of the total hand over security basic functions in specific circumstances from the driver. The test here is furnishing independent vehicles the capacity with more calculation and memory assets with a bomb evidence system. It works on the free installed Linux OS BSP bundle for both the S32V and LS2 processors with the assistance of RTMaps. It functions as the focal registering unit of the system. Subsequently, giving the ADAS system to be equipped for sending effective and better CNN designs.

3.1.1 S32V234

The S32V234 is a vision processor. It is mainly used to process image or video data. The figure 3.3 shows the features of this processor. It is an essential component of NXP Bluebox 2.0. This processor is used to perform perception applications. It enables the system with applications like Image Classification.

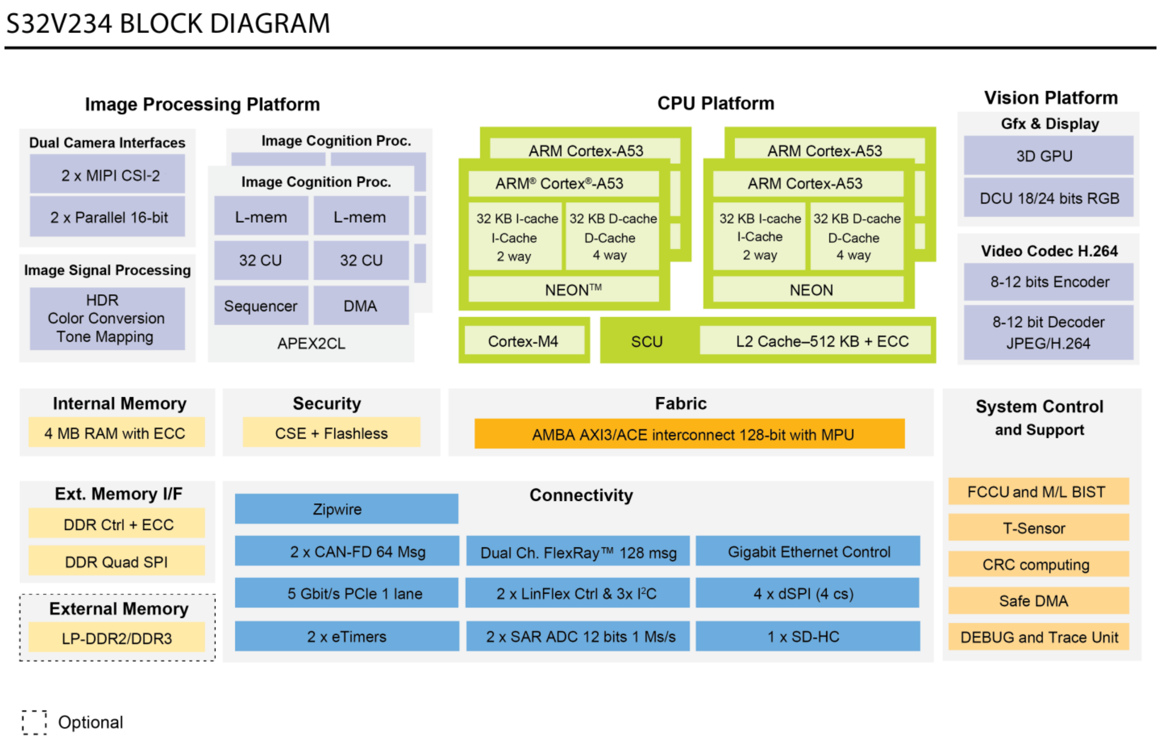

Fig. 3.3. S32V234 Vision Processor

Figure 3.4 demonstrates the different stages of the vision application process on S32V234 processor. It shows the different components, which are utilized in performing such applications.

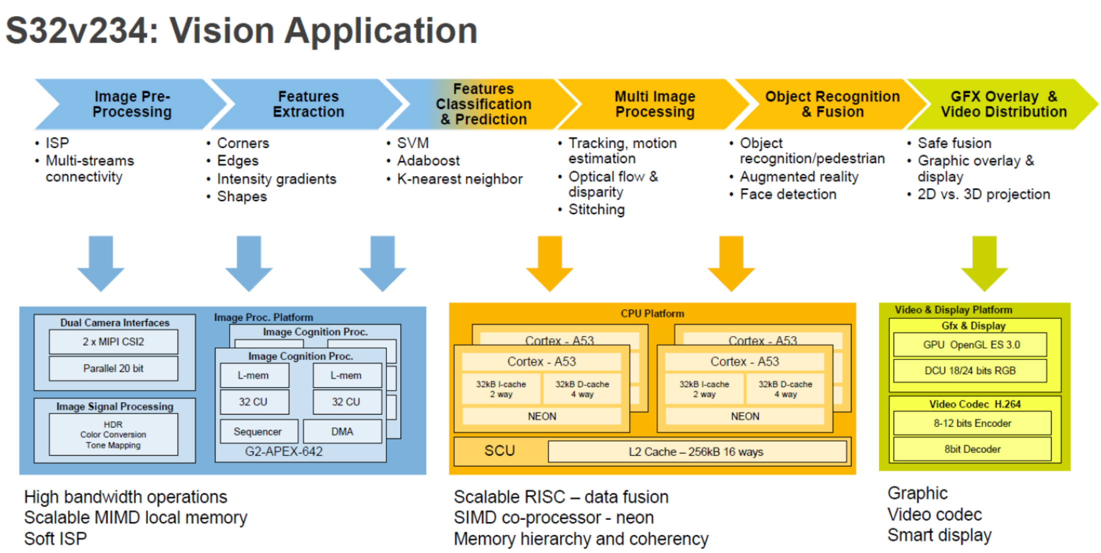

Fig. 3.4. Vision Application on S32V234

3.1.2 LS-2084A

The LS2048A is the main computational component of NXP Bluebox 2.0. All the major computations are performed by this processor. Figure 3.5 shows the main features of this processor. This processor is used to process the data which is used to perform machine learning applications. It enables NXP Bluebox 2.0 to perform real time vision applications.

3.2 RTMaps (Real-time Multi-sensor applications)

RTMaps is a non-concurrent superior platform and have an advantage of proficient and easy utilization structure for fast and robust developments. As its name suggests, it is used for real time multi-sensor applications. The simplest method to create, test, approve, consolidate and implement applications intended for the development of multi-modular based applications. It has different components, which are used for various applications.

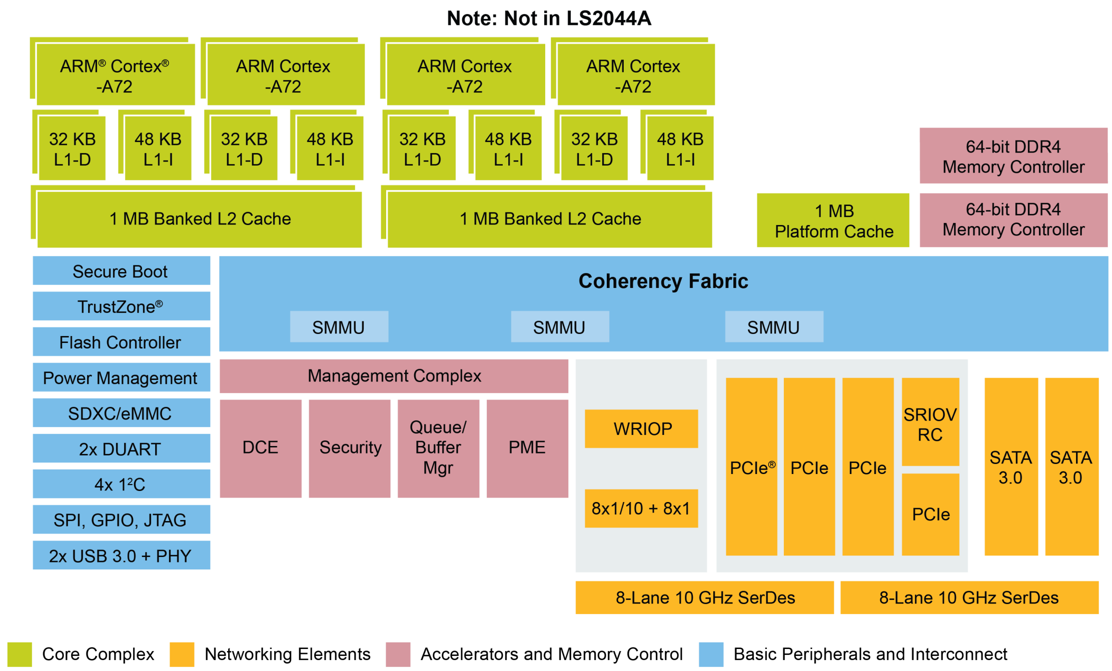

Fig. 3.5. LS2084A Processor

RTMaps Runtime Engine is the core of RTMaps. It takes care of the basic tasks like segment registration, buffer the board, time stepping, stringing, and needs. It corresponds with the outside applications and the board.

RTMaps Component Library comprises of all the essential libraries required to perform various applications. It provides suppport for Python, Pytorch, Tensorflow, C++, MATLAB Simulink models and so on.

RTMaps Studio is like an IDE which enables user to develop various applications. It is user friendly and has a very easy-to-operate User Interface. It is used to perform Image Classification on NXP Bluebox 2.0. It establishes a TCP/IP connection with the NXP Bluebox 2.0 to transmit and recieve data. Figure 3.6 shows the connection between them.

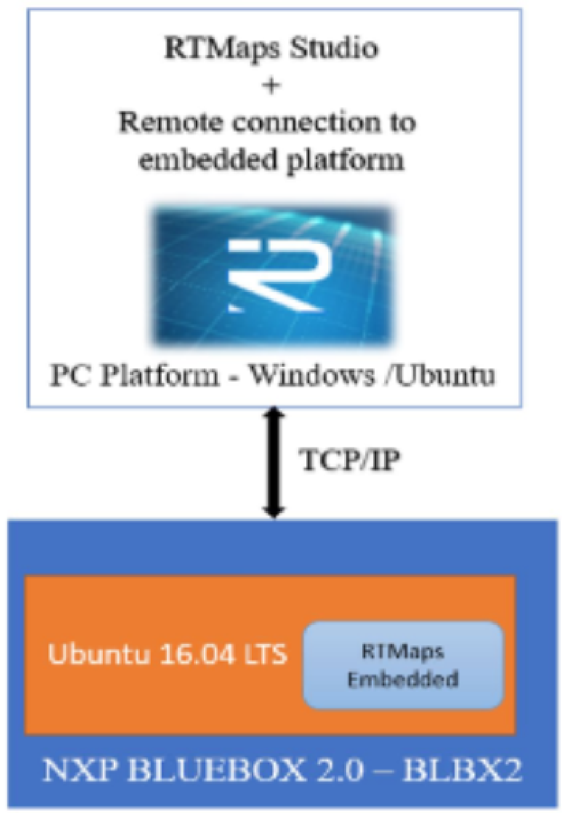

Fig. 3.6. RTMaps connection with Bluebox 2.0

4. PROPOSED ARCHITECTURES

This chapter introduces the proposed CNN architectures and discusses the proposed algorithms and their features.

Table 4.1.
A-MnasNet Architecture

A-MnasNet Architecture					
Layers	Convolutions	t	c	n	s
$32^2 \times 3$	Conv2d 3×3	-	32	1	1
$112^2 \times 32$	SepConv 3×3	1	16	1	2
$112^2 \times 16$	MBConv3 3×3	3	24	3	2
$112^2 \times 24$	Harmonious Bottleneck	2	36	1	1
$56^2 \times 36$	MBConv3 5×5	3	40	3	2
$112^2 \times 40$	Harmonious Bottleneck	2	72	1	2
$28^2 \times 72$	MBConv6 5×5	6	80	3	2
$112^2 \times 80$	Harmonious Bottleneck	2	96	4	2
$14^2 \times 96$	MBConv6 3×3	6	96	2	1
$14^2 \times 96$	MBConv6 5×5	6	192	4	1
$7^2 \times 192$	MBConv6 3×3	6	320	1	1
$7^2 \times 320$	FC,Pooling			10	

t: expansion factor, c: number of output channels, n: number of blocks and s: stride

Table 4.2.
R-MnasNet Architecture

R-MnasNet Architecture					
Layers	Convolutions	t	c	n	s
$32^2 \times 3$	Conv2d 3×3	-	32	1	1
$112^2 \times 32$	SepConv 3×3	1	16	1	2
$112^2 \times 16$	MBConv3 3×3	3	24	3	2
$112^2 \times 24$	Harmonious Bottleneck	2	36	1	1
$56^2 \times 36$	MBConv3 5×5	3	40	3	2
$112^2 \times 40$	Harmonious Bottleneck	2	72	1	2
$28^2 \times 72$	MBConv6 5×5	6	80	3	2
$112^2 \times 80$	Harmonious Bottleneck	2	96	4	2
$14^2 \times 96$	MBConv6 3×3	6	96	2	1
$112^2 \times 80$	Harmonious Bottleneck	2	192	1	2
$112^2 \times 80$	Harmonious Bottleneck	2	96	4	2
$14^2 \times 96$	MBConv6 5×5	6	192	4	1
$112^2 \times 80$	Harmonious Bottleneck	2	288	1	1
$7^2 \times 192$	MBConv6 3×3	6	320	1	1
$7^2 \times 320$	FC,Pooling			10	

4.1 Features of A-MnasNet and R-MnasNet

4.1.1 Convolutional Layers

Different types of convolutions are used to extract features from an image or a video input. Depthwise Seperable layers were used in MnasNet. In order to extract features more efficiently, Harmonious Bottleneck Layers were added to the architecture. These convolutional layers extract features from the spatial dimensions along with the channel dimensions but it changes the scale along these dimensions as well.

There is contraction-expansion of spatial dimensions while keeping the channel dimensions constant and expansion-contraction of channel dimensions while keeping the spatial dimensions constant. The computational cost of Harmonious Bottleneck Layers is less than the depthwise separable convolutional layers. This strikes a decrease in the model size of the architecture and increases its accuracy.

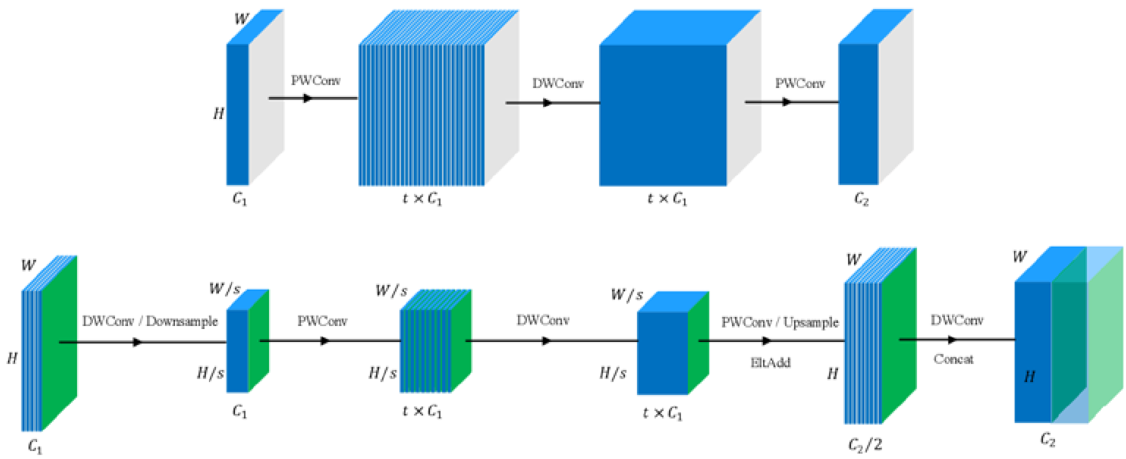

Fig. 4.1. Comparison of Depthwise Separable Convolution Layer and Harmonious Bottleneck Layer.

The spatial size of input/output feature maps is (H x W), C1/C2 are input/output feature channels, (K x K) is the kernel size and s denotes stride.

The total cost of depthwise separable convolution is:

$$(H \times W \times C1 \times K \times K) + (H \times W \times C1 \times C2) \tag{4.1}$$

The total cost of harmonious bottleneck layer is:

$$B/s^2 + (H/s \times W/s \times C1 + H \times W \times C2) \times K^2 \tag{4.2}$$

where, B is the computational cost of the blocks inserted between the spatial contraction and expansion operations. It is evident that by squeezing the channel expansion-contraction component and using a pair of spatial transformations yields a slimmed spatial size of wide feature maps in each stage, which reduces the computational cost.

4.1.2 Activation Functions

Activation functions are used to introduce non-linearity in neural networks. They determine the correct non-linear relation between the input and output signals. In 2019, Mish was introduced and it outperformed all other activation functions. It is a new type of gated softplus function. The softplus activation function can be represented as: Figure 4.2 shows the graphical representation of Mish. For comparison, Figure 4.3 shows commonly used activation functions along with the graph of Mish activation.

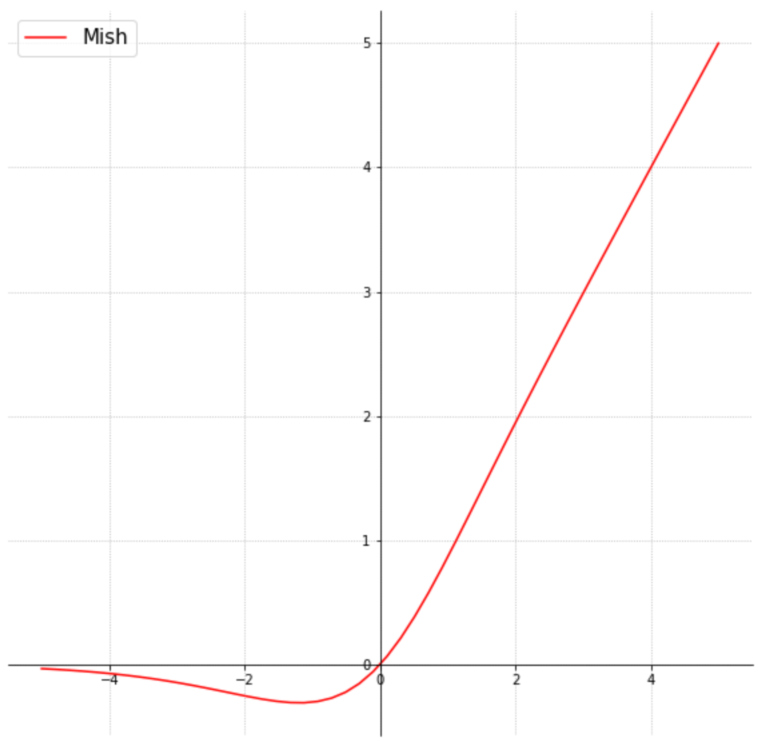

Fig. 4.2. Mish Activation Function

Mish avoids saturation due to near zero gradients, strong regularization effects, preserves small negative gradients and has effective optimization and generalization. After implementing it in R-MnasNet, the accuracy of the model increased from 90.14% to 91.13%.

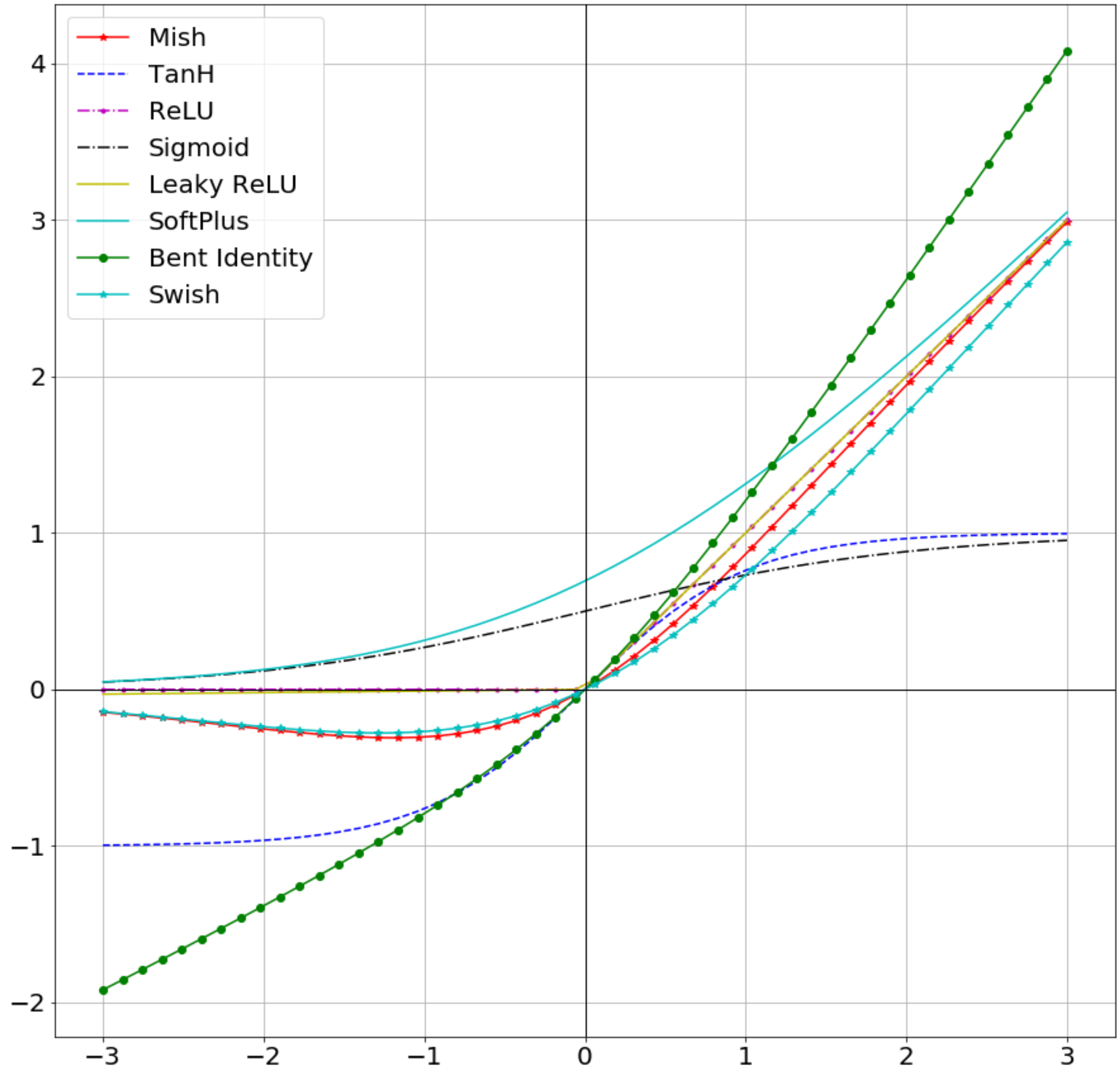

Fig. 4.3. Common Activation Functions

4.1.3 Data Augmentation

AutoAugment was used for data augmentation. AutoAugment learns the best augmentation policies for a given dataset with the help of Reinforcement Learning (RL). A policy consists of 5 sub-policies and each sub-policy applies 2 image operations in sequence. Each of those image operations has two parameters: The probability of applying it and the magnitude of the operation (e.g. rotate 20 degrees in 65% of cases). There is a controller that decides the best data augmentation policy at that instant and tests the generalization ability of that policy by running a child model experiment on a small subset of a particular dataset. After the child experiment is finished the controller is updated with the validation accuracy as the reward signal, using a policy gradient method called Proximal Policy Optimization algorithm (PPO). In this research, AutoAugment is used on CIFAR-10 dataset. The accuracy of A-MnasNet was 92.97% but after using AutoAugment the accuracy increased to 96.89%. The accuracy of R-MnasNet was 88.54% but after using AutoAugment the accuracy increased to 90.14%.

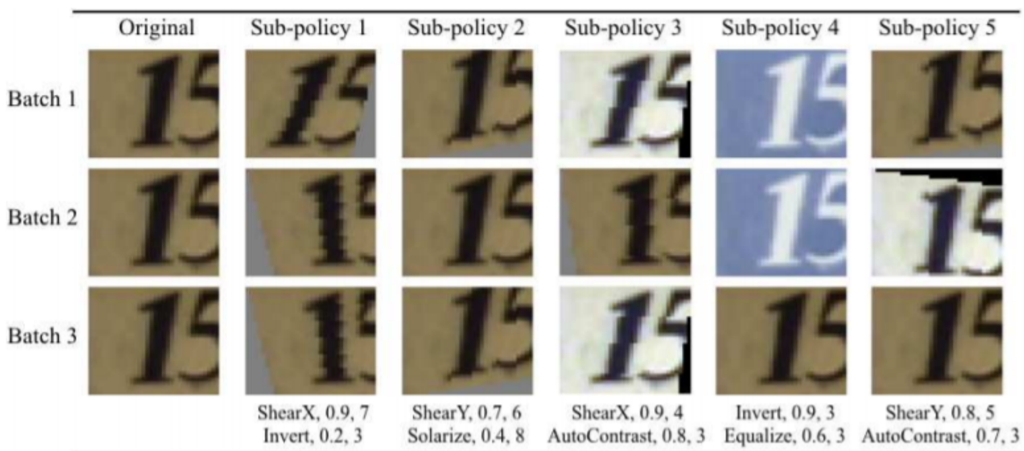

Fig. 4.4. AutoAugment

4.1.4 Learning Rate Annealing or Scheduling

While training a network, different learning rates are used to increase its accuracy. According to a pre-defined schedule, the learning rate is reduced while training the model. Some techniques like step decay, time decay, exponential decay and cosine annealing are very famous. Figure 4.5 illustrates step decay based learning rate performs better than other learning rate schedule methods. Therefore, this method is used for training A-MnasNet.

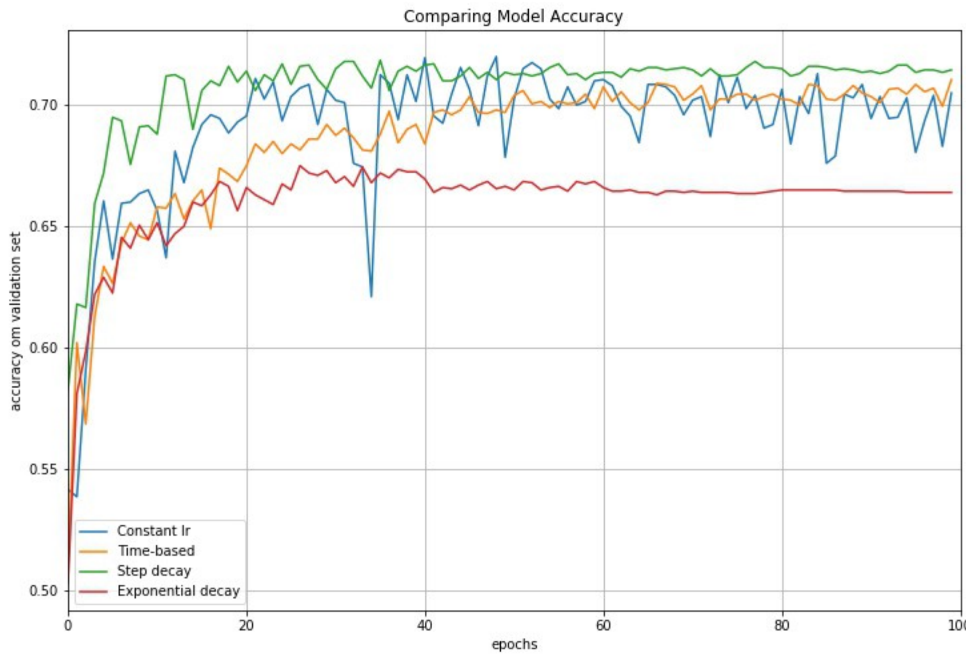

Fig. 4.5. Comparison of different LR scheduling methods.

4.1.5 Optimizers

RMSprop (Root Mean Square Propagation) was used to train MnasNet. SGD (Stochastic Gradient Descent) was used to train A-MnasNet and R-MnasNet with momentum equal to 0.9. Learning rate scheduler was used while training the network.

5. RESULTS

A-MnasNet and R-MnasNet are used for Image Classification on NXP Bluebox 2.0. These models have an accuracy of 96.89% and 91.13% with a model size of 11.6 MB and 3 MB respectively. They outperform the baseline MnasNet architecture in terms of model size and accuracy. A comparison of these models is shown in Table 5.1

Table 5.1.
Comparison of models

Comparison of models		
Architecture	Model Accuracy	Model size (in MB)
MnasNet	80.8%	12.7
A-MnasNet	96.89%	11.6
R-MnasNet	91.13%	3

These models were trained with CIFAR-10 dataset on Aorus Geforce RTX 2080Ti GPU using PyTorch framework for 200 epochs. The data was divided into batch size of 128 for training set and batch size of 64 for validation set.

Table 5.2 shows the results obtained by scaling the model with different values of width multiplier.

Table 5.3 shows the results obtained by scaling the model with different values of width multiplier.

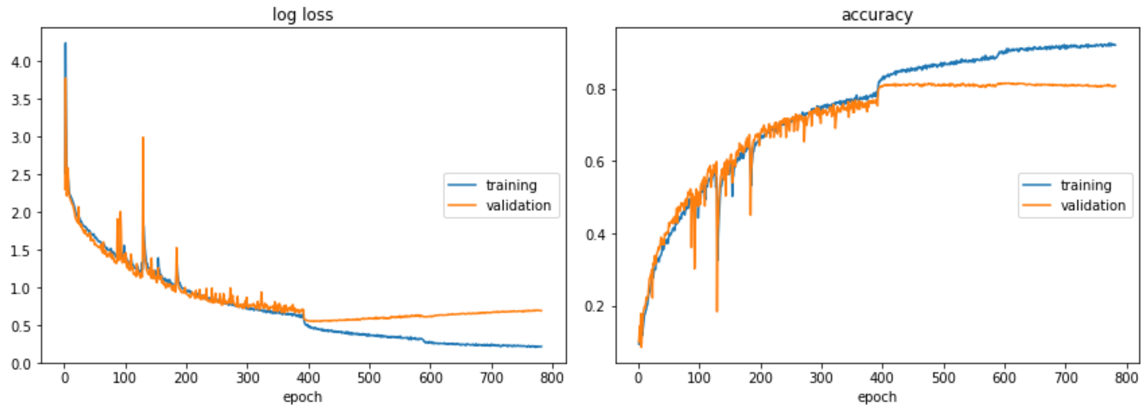

Fig. 5.1. Baseline Training Plots

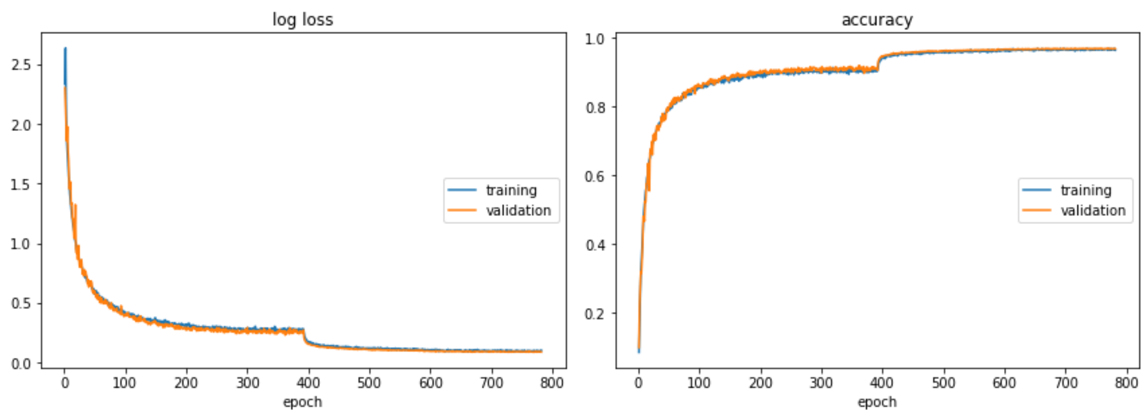

Fig. 5.2. A-MnasNet Training Plots

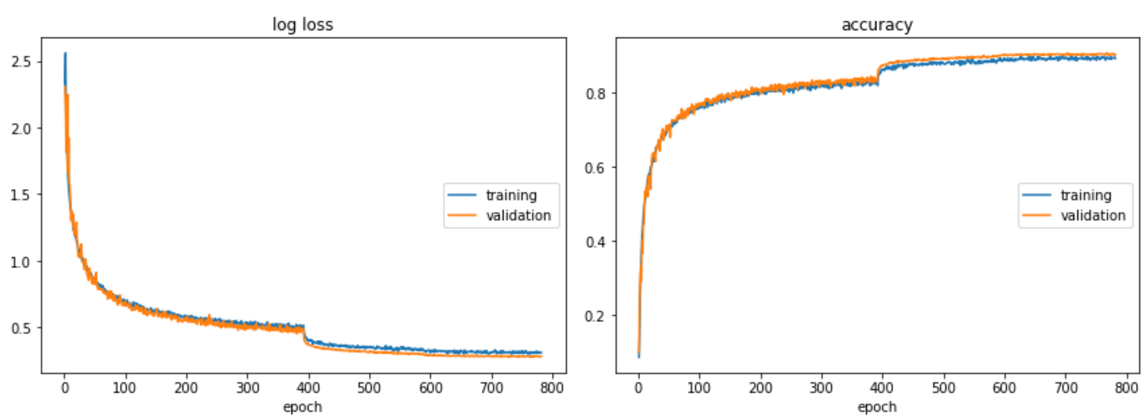

Fig. 5.3. R-MnasNet Training Plots

Table 5.2.
Scaling A-MnasNet with width multiplier

Scaling A-MnasNet with width multiplier		
Width Multiplier	Model Accuracy	Model size (in MB)
1.4	97.16%	22
1.0	96.89%	11.6
0.75	96.64%	6.8
0.5	95.74%	3.3
0.35	93.36%	1.8

Table 5.3.
Scaling R-MnasNet with width multiplier

Scaling R-MnasNet with width multiplier		
Width Multiplier	Model Accuracy	Model size
1.4	92.49%	5.6 MB
1.0	91.13%	3 MB
0.75	90.03%	2 MB
0.5	87.5%	1.3 MB
0.35	84.9%	837.6 KB

6. IMPLEMENTATION ON NXP BLUEBOX 2.0

This chapter will discuss the implementation of the proposed CNN architectures on NXP Bluebox 2.0 for Image Classification.

6.1 Implementation Setup

The proposed architectures were implemented on NXP Bluebox 2.0 for real time application like Image Classifcation. This was done by using RTMaps Studio. It provides an interface between the Bluebox 2.0 and the achitectues via a TCP/IP connection. The architecture is deployed using a python module. The process is illustrated in Figure 6.1.

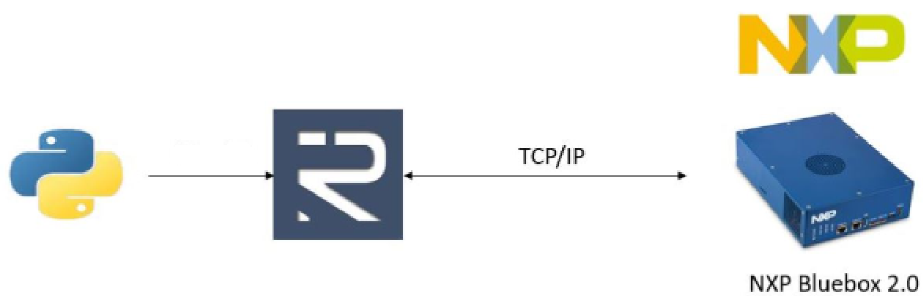

Fig. 6.1. Implementation on NXP Bluebox 2.0

After training the models on CIFAR-10 dataset, they were imported in the RTMaps Studio using its python component. The python module is shown in figure 6.2. A TCP/IP connection is used for data transmission between RTMaps and NXP Bluebox 2.0. The python component in RTMaps has a text editor that allows users to modify their code. It works due to the combination of three functions. They are Birth(), Core() and death().

Fig. 6.2. Python Component of RTMaps

- Birth(): used to initialize and define the parameters.
- Core(): used to import the CNN architectures
- Death(): used to stop the implementation.

6.2 Implementation Results

The models were successfully deployed on NXP Bluebox 2.0 and were able to predict the object in the input images accurately. The model takes input from the Cifar-10 dataset. It randomly selects an image and then predicts the object class. These predictions of A-MnasNet are shown in Figure 6.3 and 6.4 and of R-MnasNet are shown in Figure 6.5 and 6.6.

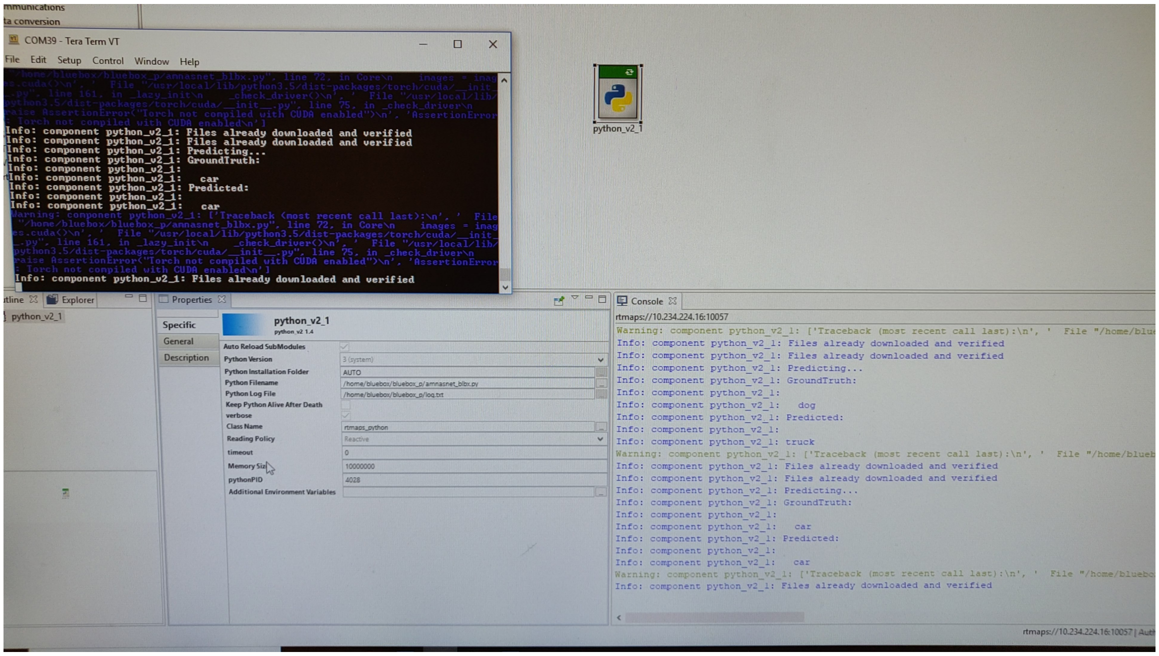

Fig. 6.3. Image Classification using A-MnasNet

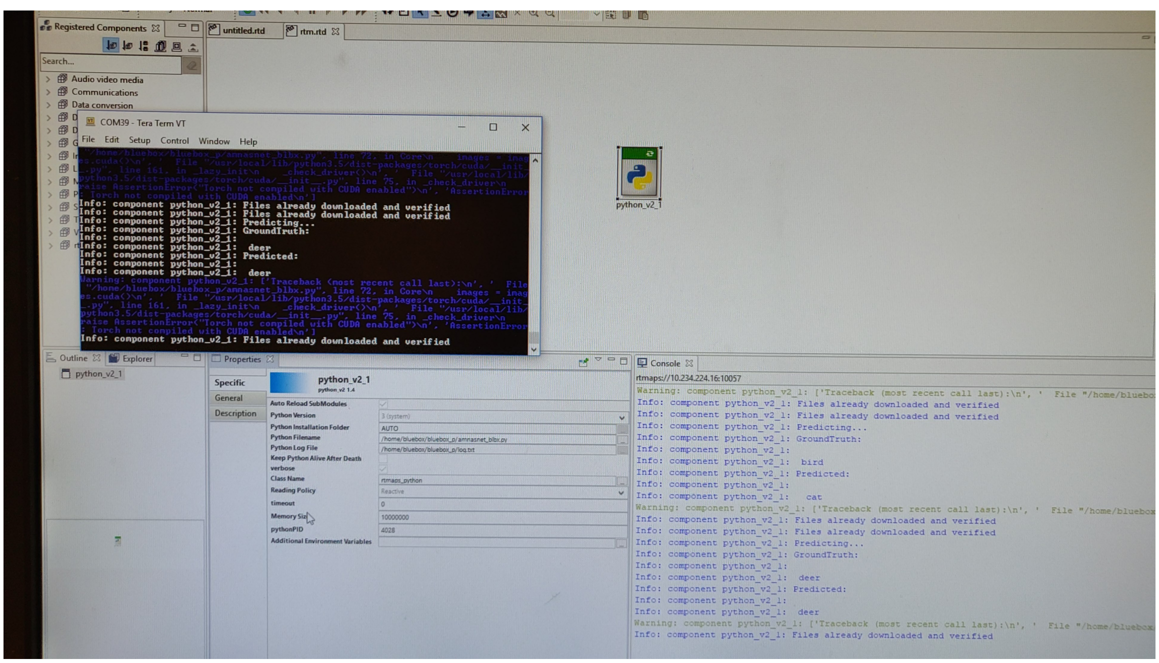

Fig. 6.4. Image Classification using A-MnasNet

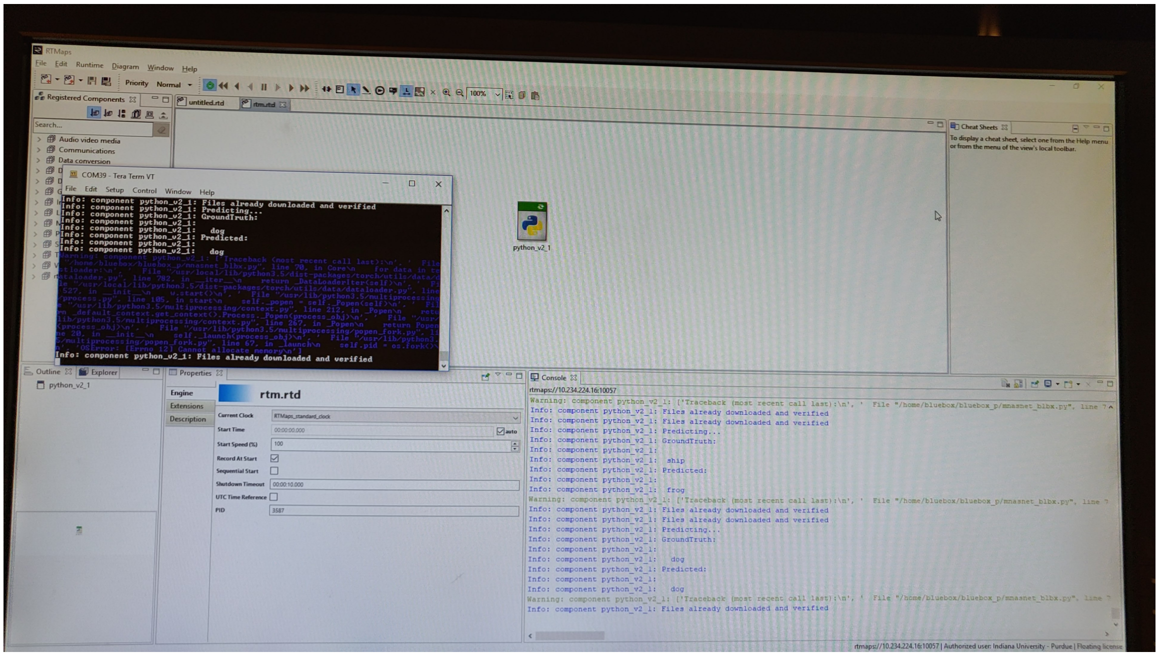

Fig. 6.5. Image Classification using R-MnasNet

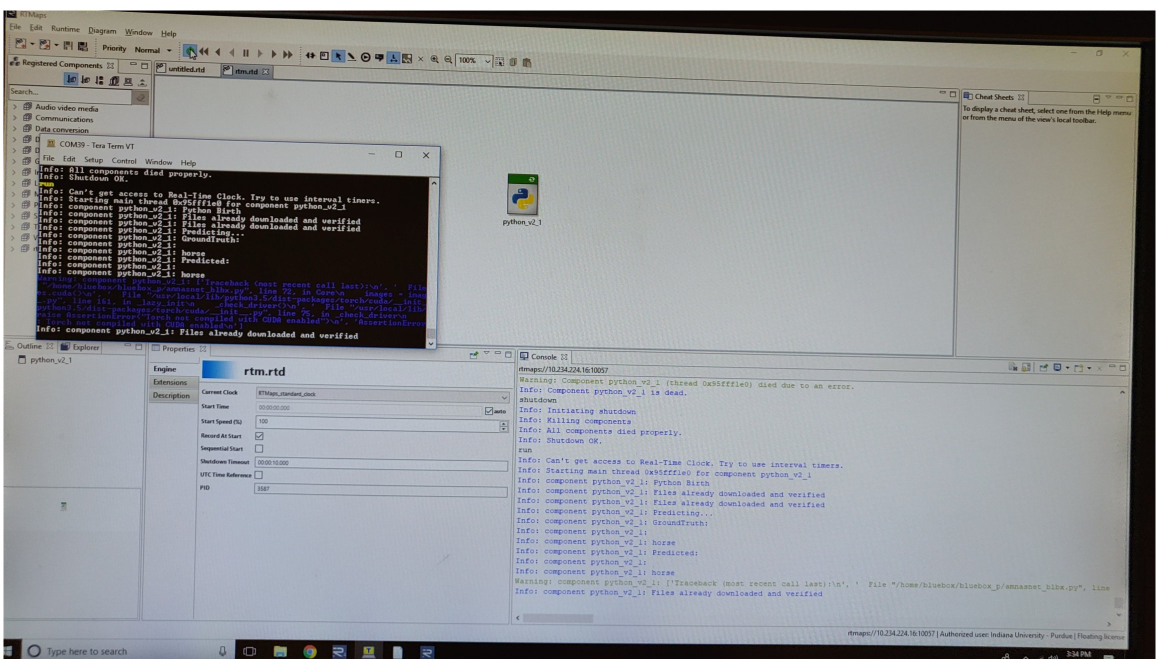

Fig. 6.6. Image Classification using R-MnasNet

7. CONCLUSIONS

This thesis demonstrates Design Space Exploration of MnasNet architecture. It proposes 2 new CNN architectures, A-MnasNet and R-MnasNet, which have been derived from the baseline MnasNet architecture. It proposes algorithms, which were used to modify the baseline architecture. It is evident that by making those modifications, the accuracy and model size of the new architectures improved.

The prime goal of proposing A-MnasNet is to make the model more efficient in terms of accuracy. The accuracy of A-MnasNet is 96.89% with a size of 11.6 MB. It outperforms its baseline architecture MnasNet which has an accuracy of 80.8% and model size of 12.6 MB. Three new layers were added to the baseline architecture. These layers are called Harmonious Bottleneck layers. AutoAugment was used to further increase the accuracy of the model.

The prime goal pf proposing R-MnasNet was to make the model more compact and having a fair trade-off between model size and accuracy. The accuracy of R-MnasNet is 91.14% with a size of 3 MB. It outperforms its baseline architecture MnasNet, which has an accuracy of 80.8% and model size of 12.6 MB. Six Harmonious Bottleneck layers were added to the baseline architecture. Mish activation was used to improve the optimization of the network. AutoAugment was used to further increase the accuracy of the model.

This thesis also demonstrates Image Classification on NXP Bluebox 2.0 using Convolutional Neural Networks. A-MnasNet and R-MnasNet which have been derived from MnasNet have been used for this Computer Vision application. These models, when trained on CIFAR-10 dataset using Pytorch framework, have a validation accuracy of 96.89% and 91.13% with a model size of 11.6 MB and 3 MB respectively. They outperform the baseline MnasNet architecture in terms of model size and accuracy. RTMaps Studio was used to deploy these architectures to NXP

Bluebox 2.0 by establishing a TCP/IP connection. These models can also be used for other computer vision applications like Object Localization, Object Detection, Semantic Segmentation etc on NXP Bluebox 2.0 as well as other mobile or embedded platforms.

8. FUTURE SCOPE

Deep Learning is growing rapidly. Every year, new algorithms are proposed and the research never stops. With the advent of new algorithms, the existing algorithms can be optimized. Design Space Exploration has various parameters which affect the performance of the CNN architecture. It is very important to tune the hyper-parameters to get the best performance of the model. Different optimization techniques can be used to improve the back propagation during training. Initial during the training, the model is initialized with random tensor values. Initialization techniques like Xavier Initialization can be implemented to optimize the initialization of parameters.

Deep Compression is a technique, which is used to reduce the model parameters. It uses pruning, quantization and huffmann encoding on the network to compress the network. This technique has been used on state-of-the-art architectures to reduce their size and has successfully accomplished that. This technique could be used to improve A-MnasNet and R-MnasNet. These architectures were trained and tested on CIFAR-10 from scratch. Transfer Learning can be used to improve the efficiency of these models. This architectures can be also be used for other computer vision applications like object detection, object recognition, semantic segmentation etc.

REFERENCES

REFERENCES

[1] M. Tan, B. Chen, et al. “MnasNet: Platform-Aware Neural Architecture Search for Mobile” arXiv:1807.11626v3 [cs.CV] 29 May 2019, last accessed on 29 Nov 2020.

[2] D. Li, A. Zhou, A. Yao. “HBONet: Harmonious Bottleneck on Two Orthogonal Dimensions” arXiv:1908.03888v1 [cs.CV] 11 August 2019, last accessed on 29 Nov 2020.

[3] E. Cubuk, B. Zoph, D. Mane, V. Vasudevan, QuocV. Le Google Brain, “AutoAugment: Learning Augmentation Strategies from Data” arXiv:1805.09501v3 (2019), last accessed on 29 Nov 2020.

[4] https://www.cs.toronto.edu/ kriz/cifar.html, last accessed on 29 Nov 2020.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ Imagenet classification with deep convolutional neural networks” In Advances in neural information processing systems, pages 10971105, 2012.

[6] A. Howard, et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications.” arXiv:1704.04861 (2017), last accessed on 29 Nov 2020.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L. Chen. “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” arXiv:1801.04381v4 (2019), last accessed on 29 Nov 2020.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z.Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. “Imagenet largescale visual recognition challenge”, International Journal of Computer Vision, 2015.

[9] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. “Quantized convolutionalneural networks for mobile devices”. arXiv preprint arXiv:1512.06473,2015, last accessed on 29 Nov 2020.

[10] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y.Chen. “Compressing neural networks with the hashing trick”. CoRR,abs/1504.04788, 2015, last accessed on 29 Nov 2020.

[11] S. Han, H. Mao, and W. J. Dally.“Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding”.CoRR, abs/1510.00149, 2, 2015, last accessed on 29 Nov 2020.

[12] K. Simonyan, A. Zisserman.“Very Deep Convolutional Networks For Large Scale Image Recognition” arXiv preprint arXiv:1409.1556v6 (2015), last accessed on 29 Nov 2020.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision." arXiv preprint arXiv:1512.00567, 2015, last accessed on 29 Nov 2020.

[14] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprintarXiv:1602.07261, 2016, last accessed on 29 Nov 2020.

[15] S. Ruder. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747, 2017, last accessed on 29 Nov 2020.

[16] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K.Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 1MB model size. arXiv preprint arXiv:1602.07360, 2016, last accessed on 29 Nov 2020.

[17] A.r Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, K. Keutzer "SqueezeNext: Hardware-Aware Neural Network Design." arXiv preprint arXiv:1803.10615v2, last accessed on 29 Nov 2020.

[18] K. He, X. Zhang, S. Ren, J. Sun "Deep Residual Learning for Image Recognition." arXiv preprint arXiv: 1512.03385 (2015), last accessed on 29 Nov 2020.

[19] C. Wang, Y. Xi. Convolutional Neural Network for Image Classification. (http://www.cs.jhu.edu/ cwang107/files/cnn.pdf), last accessed on 29 Nov 2020.

[20] M. Hossain, F. Sohel, M. Shiratuddin, H. Laga (2018). "A Comprehensive Survey of Deep Learning for Image Captioning." arXiv preprint arXiv: 1810.04020, last accessed on 29 Nov 2020.

[21] Z. Zhao, S. Xu, and X. Wu. "Object Detec-tion with Deep Learning: A Review." arXiv preprint arXiv:1807.05511 (2019), last accessed on 29 Nov 2020.

[22] A. Fred, M. Agarap. "Deep Learning using Rectified Linear Units (ReLU)." arXiv preprint arXiv:1803.08375v2[cs.NE], 2019, last accessed on 29 Nov 2020.

[23] X. Zhang, X. Zhou, M. LinJian, Sun. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile." arXiv preprint arXiv:1707.01083v2, 2017, last accessed on 29 Nov 2020.

[24] F. Chollet (2017). "Xception: Deep Learning with Depthwise Separable Convolutions." arXiv preprint arXiv:1610.02357, last accessed on 29 Nov 2020.

[25] B. Zoph, E. Cubuk, G. Ghiasi, T. Lin, J. Shlens, Q. Le. "Learning Data Augmentation" arXiv preprint arXiv: 1906.11172, last accessed on 29 Nov 2020.

[26] T. Dozat (2016). "Incorporating Nestrov Momentum into ADAM." Workshop track-ICLR 2016.

[27] D. Kingma and J. Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980, 2014, last accessed on 29 Nov 2020.

[28] B. Zoph, V. Vasudevan, J. Shlens and Q. Le, "Learning Transferable Architectures for Scalable Image Recognition", arXiv:1707.07012v4 [cs.CV] 11 Apr 2018, last accessed on 29 Nov 2020.

[29] S. Ruder, "An overview of gradient descent optimization algorithms*", arXiv:1609.04747v2 [cs.LG] 15 June 2017, last accessed on 29 Nov 2020.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research 15 (2014) 1929-1958, Submitted 11/13; Published 6/14.

[31] S. Venkitachalam, S. Manghat, A. Gaikwad, N. Ravi, S.Bhamidi and M. El-Sharkawy. Real time Applications with RTMaps and Bluebox 2.0, ICAI'18.

[32] P. Shah and M. El-Sharkawy, "A-MnasNet: Augmented MnasNet for Computer Vision," IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS 2020), Springfield, Massachusetts, Aug 9-12, 2020.

[33] P. Shah and M. El-Sharkawy, "R-MnasNet: Reduced MnasNet for Computer Vision," International IOT, Electronics and Mechatronics Conference (IEMTRONICS 2020), Vancouver, Canada, Sept 9-12, 2020

[34] https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/s32v2-vision-mpus-/s32v2-processors-for-vision-machine-learning-and-sensor-fusion:S32V234, last accessed on 29 Nov 2020.

[35] https://www.nxp.com/design/development-boards/automotive-development-platforms/nxp-bluebox-autonomous-driving-development-platform:BLBX, last accessed on 29 Nov 2020.

[36] Intempora.com. (2018). Intempora - RTMaps - A component-based framework for rapid development of multi-modal applications. [online] Available at: https://intempora.com/, last accessed on 29 Nov 2020.