Neural Network Framework using Emerging Technologies for Screening Diabetic Retinopathy

by

Koteswararao Chilakala

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday November 22, 2021 at 04:00 PM CET.

Student number:	5121698	
Project duration:	January 4, 2021 – November 22, 2021	
Thesis committee:	Dr. Rajendra Bishnoi,	CE,TU Delft, supervisor
	Prof. Dr. Stephan Wong,	CE, TU Delft
	Dr. Matin Jafarin,	DCSC, TU Delft
	Anders Eikenes,	CEO, Oivi
	Sumit Diware,	CE,Rsearcher

TU Delft

Abstract

Diabetic Retinopathy (DR) is one of the leading causes of permanent vision loss. Its current prevalence is around 45 milions across the globe and is projected to 70 million by 2045. Most of the people with this disease condition belong to remote and low income settings. We can reduce this incidence, if quality medical care is accessible in remote areas. With the current advancements in imaging technologies, fundus examination can be carried out on a handheld device. We need to improve such devices to deliver high quality services through auto detecting DR based on convolutional neural networks(CNNs) in an offline setting. Addressing these challenges, we aim to develop an integrated solution which delivers high compute at ultra-low power consumption. Firstly, we have created 3 datasets of different sizes merging multiple public datasets to create a vigilant model training process. This is to make the CNN model robust to real-world noise. CNNs trained on smaller datasets have shown a 15% accuracy drop on evaluation datasets where as CNNs trained on large datasets showed consistent performance. Secondly, we have proposed a new binary labelling scheme based on the multi-class output to maximize the utility of its softmax probabilities. We have achieved 90.26% accuracy on evaluation dataset with the new scheme. These high performing models along with compression techniques are implemented on resistive random access memory (RRAM) based computational in memory (CIM) architecture. These implementations resulted in atleast 200x improvement in energy consumption for inferring on one image when compared to CPU, GPU and mTPU (google coral dev board). Similarly, latency improvements 28x,130x and 2000x compared GPU,CPU and mTPU are registered. Model quantization with 4-bit precision for weights have preserved the original accuracy and showed 4x improvements in energy consumption on RRAM implementation.

Preface

This thesis represents my final work as an Embedded Systems Masters student at Delft University of Technology and comprises of the research that I carried out in the past 11 months in the Computer Engineering (CE) research group. Although the journey was not easy but it was a fulfilling experience and I am honoured to have experienced this time, although remotely, with the members of the CE group.

I would like to thank DR.Rajendra Bishnoi for his extensive feedback and guidance during this thesis. It was only through your constant questionnaires during the literature survey that I was able to come up with a research question for the thesis. I would like to thank Prof. Stephan Wong for his support in completing my thesis.

To Sumit Diware, thank you for always taking out time to assist me with setting up and troubleshooting the proposed work and discussing about the possible approaches to move formward.

I would also like to express my gratitude to Prof. Matin Jafarian and Anders Eikenes for being part of my thesis committee.

Last but not the least, I would like to thank my family and friends for their constant support and encouragement in successfully completing this thesis, especially during a pandemic.

Koteswararao Chilakala
Delft, November 2021

Contents

List of Figures

List of Tables

1

Introduction

This is introduction chapter of this thesis, if you can read this line with less effort then you should be happy that your vision is better than many people suffering from vision loss. There are around 45 million [13] people having vision loss across the globe. Diabetic Retinopathy(DR) is one of the major causes for this vision loss, it is not reversible and causes permanent damage. If it is detected early then we can manage health of the eye with no further vision loss. DR can also lead to the development of other diseases related to diabetes such as cardiovascular events and peripheral neuropathy. According to epidemiological studies [54],[36], the majority of the people suffering with severe cases of DR are aged (>50 years), from low income and remote settings. Conventional methods[51] are not accessible to these settings as it involves manual grading of the retinal image, captured from a table top retinal camera [14], [2]. Most of these people are currently being screened for DR through screening camps conducted by organizations like, WHO(World health Organization) and International Agency for the prevention of Blindness (IAPB) regularly in collaboration with local hospitals. With the advent of optics and imaging technologies, conventional huge fundus cameras have now become hand-held devices like [11],[15],[34]. These are battery operated devices and can be easily carried to remote areas. Though most of these devices are relatively new to be accepted and replace the current clinical regime, they have shown enough potential to be used for screening and automatic detection of DR. Some of these devices are FDA approved and being used in the above mentioned screening programs.

According to international standards [49], DR is classified into 5 classes (0-4): no DR(0), mild non proliferative DR(1), moderate non proliferative DR (NPDR)(2), severe NPDR and 4) proliferative DR(4). The manual grading of retinal images is a time taking and tedious process. It involves identifying fine features in a large image which can only be performed by trained professionals and ophthalmologists.

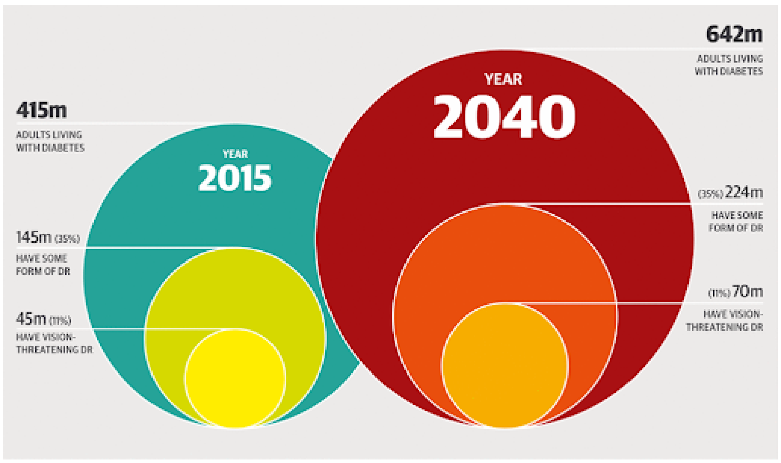


Figure 1.1: Prevalence of Diabetic Retinopathy [13]

Grading DR into 5 classes falls into a typical image classification problem where algorithms detect the severity of DR automatically. Automatic detection of DR is a popular research topic and there is extensive research available to increase its performance. Most of the literature [40], [19], [48], [45], [38], [39], [26], [27] with high performance is available for binary classification where it classifies the images into 'non-referable DR' for 0,1 class and 'referable DR' for 2,3,4 classes. The literature available for multi-class classification [29], [56], [58] is very close to delivering reliable performances. It suffers in classifying DR in mild NPDR and severe NPDR classes with high sensitivity. Most of the above mentioned works used convolutional neural networks (CNNs) for classifying DR. Other methods such as hand crafted filters and image processing routines [22],[28],[43] were proposed but are not comparable to CNNs.
Although this research possesses state of the art techniques, delivering these computationally expensive CNNs onto a handheld device still have many opportunities and scope for development. Firstly, the CNN models reported above have not demonstrated any evidence to support its generalizing capabilities. These hand held devices have their own specific attributes related to image resolution, imaging methods(optical) and sensor properties which can make it challenging to apply the above CNN models. Also these studies [37], [53] have shown that binary classification outcome has decreased the overall confidence on the device and its usability. It suggests that some additional information to assist the binary decision will be more reliable. These require a dedicated machine learning model which can tolerate real world noise with a more comprehensive labelling mechanism. And most importantly these handheld devices do not have resources to deploy these computationally expensive CNN algorithms. This leaves us opportunities to improve accessibility of machine learning algorithms to detect DR and usability aspects of handheld cameras.

Currently, the prime focus of these handheld fundus cameras is to capture a gradable image and detect DR with high sensitivity and specificity. There is little amount or almost no work discussing the system-level aspects of these devices. They depend on off-the-shelf system-on-modules or processors for system development. Using machine learning algorithms to detect DR on a battery operated device requires power efficient and faster run time implementation to maximize usability of the device.It is argued that the current latency for internet dependent grading is less than 1 minute and it does not require any offline module. This work [50] discusses the use of an off-line AI implementation for detecting DR and facilitating it on a large scale device deployment in a poor resources setting. It mentions that the offline feature results in wider and quicker adaptation of the device. In an internet dependent device, a typical image resolution of retinal image 2500 x 2500 will result in higher consumption of energy in data transfers between server and device. It is not suggested for a device targeted to work in remote areas with limited resources. An ultra-low power implementation of an off-line processing module will contribute to the advancement of the device and make it more accessible in screening DR. Some of the studies [58] also mentioned that the fundus image can be used to diagnose other ocular diseases apart from retinal diseases. Therefore, developing such hardware accelerators can be an opportunity to tune the algorithms to detect multiple diseases on a single platform. There isn't any literature found for hardware implementations or benchmarks of CNNs for DR.

Many of the von-neumann architectures based on Graphics Processing Unit (GPUs), Tensor Processing Unit (TPUs) are designed for accelerating machine learning on embedded or mobile platforms. These architectures have different memory and processing units. The growth in artificial intelligence and machine learning have made the applications data intensive. The volume of data processing required by these applications is skyrocketing. The bandwidth available to support these data transfers between memory and processor have reached a bottleneck. Although, memory and processor are making advancements at their own pace. The gap between their pace of developments over the years has created a memory wall, i.e, even if the processor is capable of higher performance, memory could not support that bandwidth for data communication. This has capped the performance of many GPUs and TPUs platforms to a lower than their peak performance. Any solution that can bring memory and processing units closer or even avoid any transfer for computation will help in addressing these challenges.

Some of the emerging technologies show the possibility of computation-in-memory(CIM) which processes data in its location. This technology is called resistive random access memory (RRAM).

CIM architectures based on these emerging technologies offer heavy parallelization to exploit in data intensive applications. The characteristics of the device offer 3.5x in energy management and 7.5 x increment in computational density [47] due to no data transfer. This is a CMOS based memory technology that operates by varying the resistance of the device following kirchoff's / Ohm's law. This is a non-volatile memory, applying certain voltage sets and resets the device. They are implemented in crosscbar framework which offers parallelism to implement matrix vector multiplications. We can potentially exploit this technology to meet the demands of our ultra low-power implementation of machine learning models.

1.1. Problem Definition

High performance of machine learning models - The machine learning models mostly demonstrated in the literature have not presented an evaluation scheme for model deployability in real world scenarios. This lack of information brings many challenges in building high performance models for new fundus cameras. This also reduces the usability aspect of the device.

Classification outcome - The existing multi-class classification outcomes need to have improved performance in detecting mild, moderate and severe cases. The binary classification outcome needs some extra information to assure user on the predicted outcome.

Accessibility - Most of the handheld fundus cameras do not have resources to deploy computationally heavy models. The internet dependency of these devices for the classification outcome is not preferred in a low resources setting as most of the energy consumption will be utilized for transferring high resolution images. Opportunities in RRAM based computational in memory technologies offer greater specification for hardware optimization in the device.

1.2. Contribution

In this work we plan to develop an ultra-low power implementation for auto-detection of DR with the constraints of high accuracy and less latency on a RRAM based CIM framework.

1. **Model Generalizability** - A training scheme to split 4 publicly available datasets into 3 categories of different sizes to perform vigilant training on CNN models. Finally, demonstrated the utility of vigilant training by evaluation on a new dataset in order to deploy them in real world scenarios. Models trained on small datasets have shown higher performance on the trained dataset but failed poorly on new dataset. Models trained on large datasets have shown consistent output on train and new datasets without any performance loss.

2. **New labelling scheme for classification outcome -** Proposed a new labelling scheme to provide a binary outcome but perseves the outcome of multi-class classification to provide extra information to assure user on the predicted outcome. This labelling scheme can be also used for pseudo multi-class classification. The models performing moderately in multi-class classification have shown increase in high sensitivity and specificity. It produced comparable performance with state of the art literature.

3. **First of its kind implementation on RRAM based CIM platform for DR.** Not only the literature on hardware metrics but also on RRAM CIM based architecture. We are one of the early groups to exploit these emerging technologies for this application. When compared to an embedded platform the energy efficiency is of the order 1000x.

4. **Benchmarking on hardware platforms -** The literature on DR is lacking any benchmarking information related to hardware implementations. We have reported energy, run-time information for inferring one image on DR classification tasks in CPU, GPU, TPU(embedded platform) and RRAM device. This performance comparison has shown that RRAM has impressive characteristics for data heavy applications. More than 200x efficiency in energy is reported compared to any hardware platform.

5. **Energy efficiency - compression schemes for hardware** The parallelization and crossbar network in RRAM offers implementation with various bit-width precisions. Deeper models with a

high number of parameters can be pruned in a structured fashion to benefit with energy efficiency. The 4 bit-quantization scheme is valid for machine learning models without any accuracy loss.

1.3. Thesis Organization

The chapters in this thesis is organized as below,

Chapter-2 - It provides information on the background required to understand the outcomes of this thesis. It covers DR and its classification, conventional practices for screening DR, basics of convolutional neural networks(CNN), model architectures exploited, basics of RRAM and ISAAC architecture.

Chapter-3 - presents the related work for binary and multi-class classification of DR. A critical review on this literature .

Chapter-4 - Informs the proposed schemes to conduct this thesis, it includes, small, Medium and Large datasets generation, model training strategies, binary classification with severity score, hardware compression techniques and network mapping on to RRAM.

Chapter-5 - it contains information about experiments conducted and their results after describing the setup and tools used to perform.

Chapter-6 - The conclusion part explains a brief outline of the work accomplished in this thesis and possible future directions for this thesis.

2 Background

2.1. Diabetic Retinopathy

Diabetic Retinopathy(DR) is one of the highly prevalent diseases across the globe. This is an irreversible disease condition which leads to permanent vision loss. In DR, higher glucose levels and blood pressure causes damage to blood vessels in the retina. This incidence is more for people above the age of 50 years and having diabetes.It is highly likely that every diabetis patient with age develops this condition. It is estimated around 430 million [36] people have diabetes and close to one third of its people suffer from some forms of DR. It is growing as one of the global health crisis. It is known that we can provide better treatment when detected at an early stage. Many health / non-profit organizations work continuously to early detect people suffering from DR and help them with managing their health condition from further vision loss.

DR manifests as a combination of swollen blood vessels and blood spillage. These manifestations can vary in size based on the severity. In the manual grading process, trained professionals look for these features in retinal images. Several of these features are labelled as different types of lesions for grading the severity of DR. These features can be seen in Figure 2.1. Some of these features are,

- Microaneurysms (MA) - These features appear as very tiny red dots due to capillary dilation in ultra high resolution retinal images. They as early indicators for DR.
- Haemorrhages (HM) - These are blot or dots in the image and are relatively bigger than MA..
- Hard Exudates (HE) - These are yellow-white retinal deposits caused by leakage of plasma. They appear in outer layers of the retina with sharp boundaries and in different size from small to large.
- Soft Exudates (SE) - They are also called ‘Cotton Wool Spots (CWS)’. As the name suggests it appears in greyish patches due to the swelling of nerve fibre. Usually are in oval or round shape.

As a rule of thumb, MA and HM are red lesions, hard and soft exudates appear in white color. Towards severe cases, we identify intra retinal microvascular abnormalities (IRMAs) and neo-vascularizations (NV).

2.1.1. International standards for classification of DR

As per International standards [49], the severity of DR is graded into 5 classes. They are No DR (0), mild non proliferative DP (NPDR)(1), moderate NPDR (2), severe NPDR(4) and severe proliferative DR (PDR). A sample of retinal images belonging to 5 classes can be seen in Figure 2.2. Though there is a label for each severity, for algorithm development, numeric label is widely used. For binary classification tasks, classes 0,1 are labelled as ‘non-refereable DR(’ and classes 2,3,4 as ‘Referable DR’. Features mentioned in the previous sections are used to design a protocol, each DR and its criteria for severity can be seen in Table 2.1.

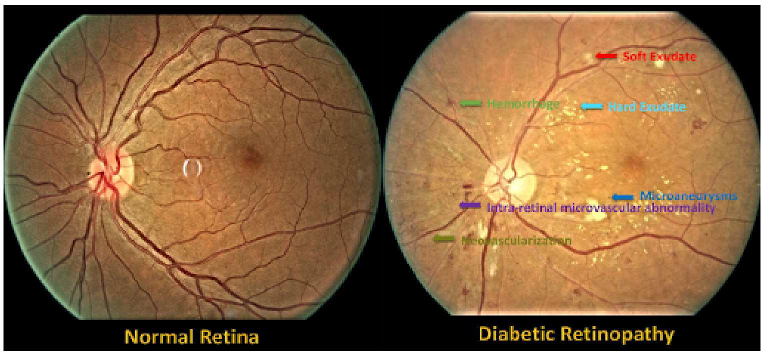


Figure 2.1: An example of fundus/retinal image with and without DR. The image with DR is labelled with types of lesions causing DR. [58]

DR Severity	Responsible features
No DR (0)	No presence of lesions
Mild NPDR (1)	Only MA
Moderate NPDR (2)	In addition to MA, HM and other features but less in number than severe DR
Severe NPDR(3)	One of the below, HMs around 20 Blood spillage in more than 2 quadrants Intra retinal micro vascular abnormalities in at least 1 quadrant. No indications of PDR
Proliferative DR (4)	Vitreous / preretinal HM and NV

Table 2.1: 5 classes of DR severity according to International standards [49]

2.1.2. Diagnosing DR

Anatomically, retina is inaccessible compared to other ocular structures. It needs to be diagnosed with vigilant and standard procedures. There are a wide variety of these devices, starting from table-top devices to handheld or portable devices.

Conventional clinical methods

These fundus cameras are table top devices which are used as a standard for comprehensive retinal examination. The total examination time here can vary from 35 - 45 minutes because it has prerequisite medical procedures to capture a good quality image. Later these images are manually graded by a trained professional. It is a time taking and tedious process because very minute features present in ultra-high resolution images. Many of the first tier hospitals across the globe have started using automated AI methods to detect and classify these images. There are some advancements to avoid pre-medical procedures, but it is not yet a clinical regime. These devices are super expensive and require high maintenance. Dependency on trained professionals, cost and inaccessible tabletop fundus cameras have created opportunities to develop screening devices which can be accessible and used by a technician.

Handheld devices

Unlike the above, these do not require any pre-medical procedure to capture the retinal image. This exclusion makes it challenging to capture a good quality image. Even though extensive development is going in making this close to the standard, we have taken a trade off in quality of image for accessibility. Typically, it takes around less than 3-4 minutes on an average to capture a retinal image. Most of the fundus cameras [11],[15] use AI bases detecting mechanisms to grade severity of DR. There are very few fundus cameras commercially available for screening, most of these are in the last stages of development. As we move towards more real world settings, we have to be sufficient in handling unforeseen challenges especially with usability (high performance) and accessibility (remote areas) of the devices

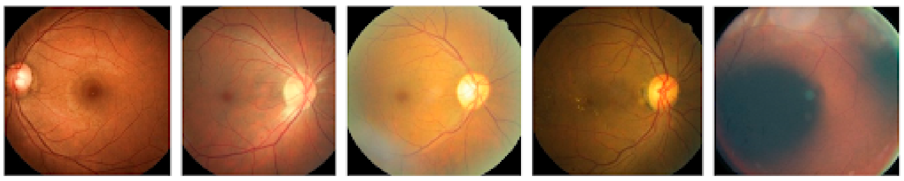


Figure 2.2: 5 stages of DR according to International standards([49])from left to right : No DR(0), mild DR(1), moderate DR(2), severe DR(3), Proliferate DR(4) [58]

Dataset/Tag	Total	Normal	Mild	Moderate	Severe NPDR	Proliferati-ve DR
EyePACS-Train [16]	35126	25810	800	4100	1020	765
EyePACS-Test [16]	53236	39433	1200	6000	1200	1200
DDR [37]	13673	3133	315	2238	118	456
APTOS [17]	3662	1805	370	999	193	295
Messidor-2 [23]	1744	1017	270	347	75	35

Table 2.2: Publicly available datasets for DR detection

Screening programs

Screening programmes play a very crucial role in controlling the prevalence of diabetic retinopathy. They plan activities throughout the year to follow up and timely refer the patients. According to the epidemiology studies [36],[32], most of the people with DR belong to remote and low income settings. This paper [42], also explains the possibility of screening in remote areas with the evolution of fundus cameras. It hopes that advancements in smartphone technologies can very well be used in addressing this global health challenge. The near to perfect handheld devices should focus on real world challenges for wider acceptance of the device. This paper [50], mentions that an offline AI detection system with handheld fundus camera will address the challenge of delivering high quality medical services in low income and resource settings.

2.1.3. Public Datasets

DR is a popular medical application which is widely explored in technical communities as well as in industries to deploy reliable CNN algorithms with high accuracy. Unfortunately, most of the industries have a private dataset and perform multi-class annotations with clinical experts. This process avoids any bias from certain clinicians and is appropriate as a reference standard. Sine we could not afford such a setup in this scope of work, we have collected 5 publicly available datasets with 5-stage classification. Table 2.2 provides information about these datasets.

EyePACS - Train and EyePACS - Test datasets are collected from an online Kaggle challenge [16] to detect DR conducted in 2016. It consists of images from both mydriatic and non-mydriatic fundus cameras where the FOV of the retina varies between 40 - 50 deg. Most of these images are collected from different parts of the USA. The provider mentions that this dataset resembles any real world dataset with artifacts such as out of focus, over-exposed or under-exposed. Similar to the above, APTOS database is also collected from Kaggle [17] but from a different online challenge conducted in 2019. These images are provided by a hospital based in India. Out of the listed datasets in the Table 2.2, it has less number of images. With DDR dataset, it is a moderately sized dataset between APTOS and EyEPACS-Train/Test with 13673 images. These images are mydriatic images collected between 2016 and 2018 from 147 countries in China.The authors claim it to be a qualitative large dataset when compared to other publicly available datasets.

2.2. Basics of Technical Concepts

This section provides necessary technical background for the used technical concepts in this thesis.

2.2.1. Convolutional Neural Networks

We are implementing a CNN based solution in this work to auto detect DR severity. CNNs are built by stacking varieties of layers to facilitate efficient learning of the models. Training these networks is an

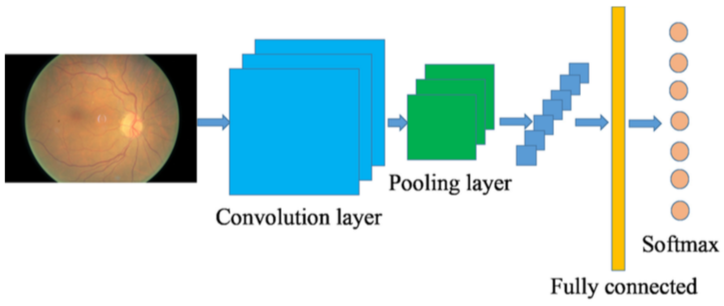
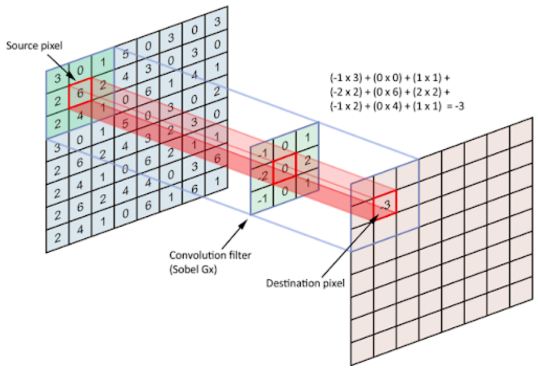
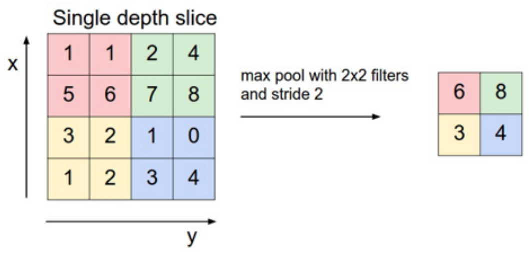


Figure 2.3: A basic CNN model outlinne used for DR detection [25]


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

(a) Convolution operation in CNN [1]



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

(b) Max pool coperation in CNN [1]

Figure 2.4: Basic operations in CNN

iterative process of updating network parameters while minimizing the loss between ground truth and the predicted outcome. It requires more sophisticated approaches in the form of a learning algorithm to run the process on a given training dataset. An overview of a typical CNN model can be seen in the below Figure 2.3.Some of the common layers used in these networks are max-pooling layer, activation layer, convolution layer and fully connected layer.

Convolution Layer

This layer consists of a collection of filters (or kernels) with different kernel sizes. It performs convolution on the n-dimensional input data using all the filters in this layer to extract a feature map. Each filter results in a different feature map in the output. In the training process, these filter values get updated based on a learning algorithm. You can visualize a convolution operation in each layer in Figure 2.4

Max-Pooling Layer

The purpose of a pooling layer is to reduce the number of parameters in the network. The input to this layer is reduced based on the conFigured stride value to a lesser output size. There are a variety of pooling layers such as max, min, GAP pooling etc. These methods help in preserving the dominant information present in the input data. A Max pooling operation can be seen in Figure 2.4

Activation Layer

The goal of this layer is to map the input data to output using a non-linear function such as sigmoid or Rectified Linear Unit (ReLU). For example, the mathematical representation of ReLU is $f(x)_{ReLU} = max(0, x)$. The non-linearity induced in this layer helps to learn unusual patterns in the data.

Fully Connected (FC) Layer

As shown in the Figure 2.5, in this layer each neuron is connected to all the neurons in the previous layer, and sends the output to all the neurons in the next layer. In some cases, multiple fully connected layers of different sizes are stacked together to obtain a feature vector for classes or dimensionality reduction. The output of each neuron in a FC is the summation of the input data multiplied by its weight parameter and add its bias.

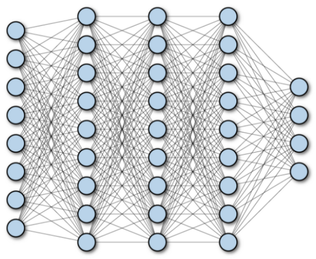


Figure 2.5: Fully Connected layer in CNN [6]

Training CNN

A full CNN model built for conventional computer vision tasks has millions of parameters to learn during training. Training CNNs to auto detect DR severity requires large training datasets with known ground truth. Initially, these parameters are set to random values or loaded from a pre-trained model as done in transfer learning. Then a severity grade is generated for each image in the training dataset. The generated severity grade is now compared to the ground truth and parameters are updated to decrease the prediction error on the image. This process is repeated many times with all the images in the training dataset until acceptable performance is registered. As a result, we get a general model to compute severity on diabetic retinopathy images. The equations here represent a typical back propagation operation performed while updating the parameters in the CNN mdoel. It explains that gradient at every neuron propagates back to all neurons in the previous layer.

$$w_{if^t} = w_{if^{t-1}} = \Delta w_{if^t}$$

$$\Delta w_{if^t} = \eta \times \frac{\partial E}{\partial w_{ij}}$$

$$E(p, y) = -\Sigma_i y_i log(p_i) where i \in [1, N]$$

w_{if^t} is the final weight in the current training epoch (t), $w_{if^{t-1}}$ is the weight belonging to the previous layer (t-1), η is the learning rate, prediction error E is calculated using categorical - cross entropy loss function, p and y are prediction and ground truth labels respectively.

The optimizers used in this work are Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam). The former suggests calculating gradient on every training sample, while the later one is an updated form of SGD with features tolerating its noise which prevents convergence of the CNN model. Each of these methods help the CNN model to converge with some tuning parameters which highly influence the training process. Some of these hyper-parameters are described in this below Table 2.3. In the scope of the thesis, we have considered only InceptionV3 and Densenet121 ImageNet computer vision models for these classification tasks. Inception V3 and DensenNet121 are CNNs with variety of layers stacked in a symmetric or asymmetric fashion.

Hyper-parameter	**Effect on CNN training**
Learning Rate	Kind of a velocity component to reach the convergence. used in weights update
Momentum	It has an accelerating effect on the learning rate. It helps in taking quicker steps.
Batch size	Number of images for a weight update step
Regularization parameter	Reduces overfitting
Number of hidden layers	Depth of the neural network can be helpful in learning complex structures
B1,B2 (Adam optimizer)	Have an averaging effect on predicting the next step in weight update.

Table 2.3: General hyper-parameters and their effect on training process.

InceptionV3

This is reported in [52] paper with the aim to scale the networks for big data applications but efficiently utilize the extra compute added while scaling up. The Figure shown in 2.8 is the base inception module where ‘n’ can be adjusted to the application requirement. Typically, ‘n’ takes the values 3,5,7. The asymmetric organization of the filters in the base module with factorized convolutional kernels

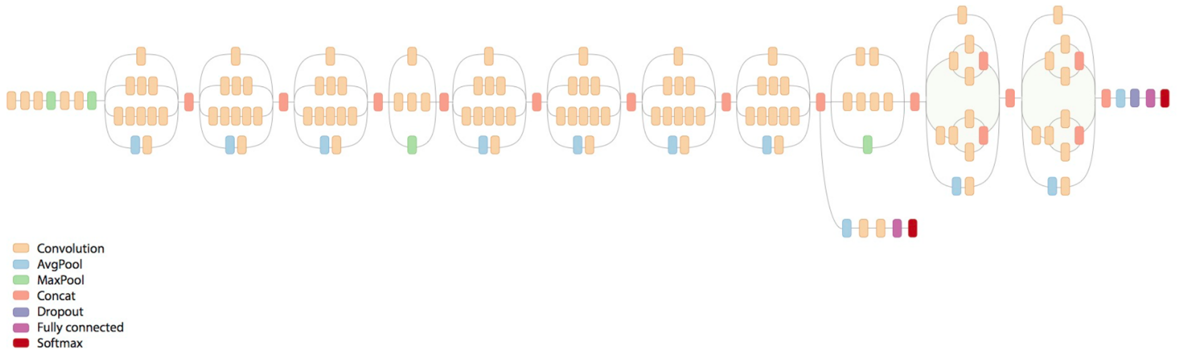


Figure 2.6: InceptionV3 network Architecture [52]

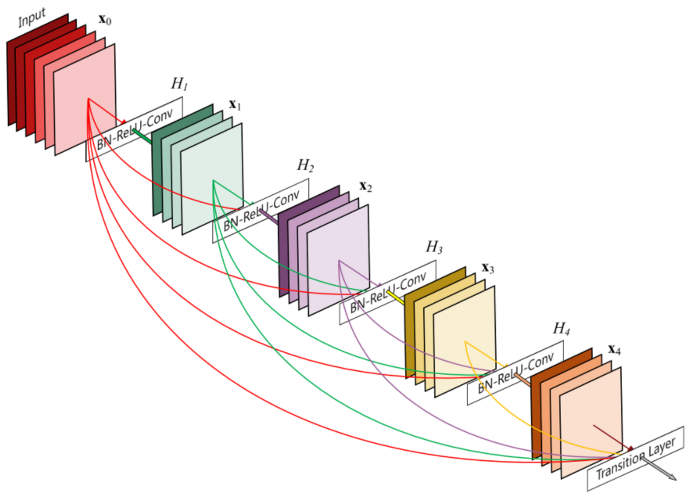


Figure 2.7: DenseNet121 network Architecture [31]

makes it fast to deliver high performance with lesser parameters. InceptionV3 built on this architecture has reported state of the art top-1(17.3%) and top-5 (3.5%) errors on the ILSVRC 2012 classification challenge. Each of this convolution layer in the inception base module is cascaded with a pooling layer and an activation layer. Several of these base modules with n = 3,5,7 are combined to build a full InceptionV3 network with a depth of 42 layers as shown in Figure 2.6. At the end of these convolutional layers, there is a FC layer with softmax classifier to spit out the prediction or severity. Total number of parameters present in InceptionV3 is around 23.8 Million.

2.2.2. DenseNet121

This is reported in the paper [31] aims to validate the performance of deeper models with shorter connections between layers that are top and layers connected to the bottom layers. The network shown in Figure 2.7 is built with dense blocks and cross layer connectivity. Since there are too many feature maps reaching the dense block, it offers more vigilant learning. Each dense block consists of Batch Normalization layer (BN), Activation layer (ReLU) and Convolution layer. Similarly the transition layer consists of Batch Normalization (BN), Convolution layer and Average Pooling. Due to the dense blocks, there are less number of feature maps which are redundant feature maps. At the same time the increased feature maps from the previous layer adds a regularizing factor and minimizes overfitting with minor datasets. The total number of learning parameters in this network is 7.2 Million.

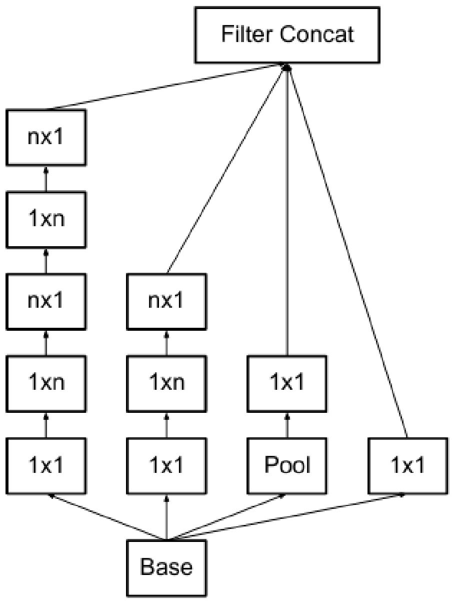


Figure 2.8: Building block of InceptionV3, n =3,5,7 [52]

2.3. Hardware Implementations

We will discuss the variety of hardware platforms considered in this work. To run a neural network any hardware primarily implements an efficient matrix vector multiplication(MVM) framework. Multiply and Accumulate (MAC) are the elementary operations of MVM, they are implemented and scaled according to the type of hardware and available resources to run several operations together.

In von-neumann architectures, we have a memory bank and a separate processing element, which performs memory fetch, decode and execute to implement any basic operation. CPUs perform these operations almost in a serial fashion hence are inefficient w.r.t speed as more data transfers are necessary. GPUs possess dense processing elements which can parallelize these operations better than CPU. There are several ASICs like TPU(Tensor Processing Unit) with special MVM processing elements. Likewise the development of novel processor structures has been going at a brisk pace whereas the advancement in memory banks has been going slow. The low band-width from memory to processor in this architecture is saturating to support the requirements of heavy data dependent applications. We can see the memory wall bottleneck in the Figure 2.10 . It not only increases the latency due to numerous data transfers but also adds to the energy consumption for highly efficient processors. This memory wall has been the driving force to build in-memory computing systems where there is no need for data movement. This can be seen clearly in the Figure 2.9. This is discussed in detail in the next section.

2.3.1. Emerging Computational In Memory Technologies

The term ‘emerging’ should notify that these memory technologies do not belong to the family of memory chips such as SRAM (Static Random Access Memory) , DRAM (Dynamic Random Access Memory) and other volatile memories that are commonly used as hard drives or RAM in our computing devices. These conventional memories possess several limitations such as charge retainment for DRAM, leakage power for SRAM. Also scaling down beyond 10 nm was not successful, so there is a need for emerging technologies with ideal characteristics of low operating voltage (<1 V), improved data retention (> 10 years), lesser energy consumption and scalability. At the core of these CIM architecture, memristors form the building block with the ability to store and process the information. These were first proposed by Chua in 1971 and first demonstrated in Hewlett-Packard (HP) labs in 2008. Interestingly, it is proposed as the relationship between the basic electrical quantities charge (q(t) and magnetic flux-intensity (ψ(y)). It is an unexplored basic circuit element when compared to widely used inductor, resistor and capacitor devices. Although it was not researched rigorously during

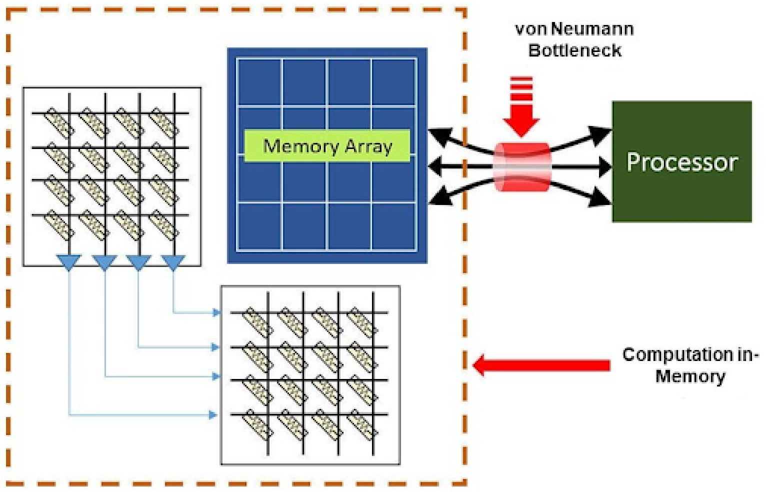


Figure 2.9: Advantages of Computation in memory w.e.t von-neauman architecture [24]

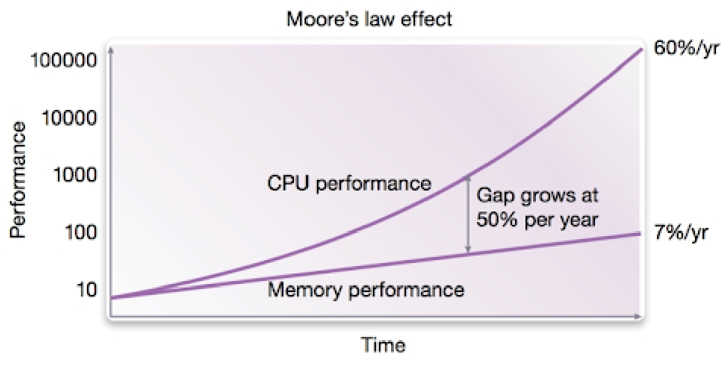


Figure 2.10: Memory Wall [9]

its inception but today this has been an answer to the bottleneck of data movement from memory to processor. More Importantly, memristors are non-volatile memories which makes it interesting to exploit for heavy data applications.

With the research going at the brisk pace, currently there are memory technologies using memristive devices. They are 1) phase change memory, 2) spin-transfer torque - magneto resistive random access memory (STT-MRAM) and 3) Resistive Random access memory (RRAM). Out of these RRAM technology has advantages in easy fabrication, based on metal-insulator-metal (MIM) structure (Figure 2.11), run time in nano scale and long data retention. We explore this technology more in this work.

RRAM uses memristive conductance to store the values and facilitate the aspect of computation-in-memory(CIM). These characteristics of the RRAM are exploited in the literature to realize a full neural network and build efficient hardware platforms in terms of area, computational complexity, less latency and energy consumption. The characteristic no data movement from memory to processing element characteristic of RRAM devices have reported 5.5x energy improvement and 7.5x computational density [47]. The set voltage (V_set) applied to the device breaks down the oxide layer and makes a conducting path. Thereby increasing the conductivity of the oxide layer to a low resistance state (LRS). We also have reset voltage (V_reset), which reduces the conductivity of the oxide layer leading to a high resistance state (HRS). The SET and RESET process of RRAM can be seen in Figure 2.12). So even after the removal of the applied voltage the RRAM can still hold its value, thus making them non-volatile memories. A very small voltage (V_read) is applied across the device results in small sensed current which can be used to find the state of the device.

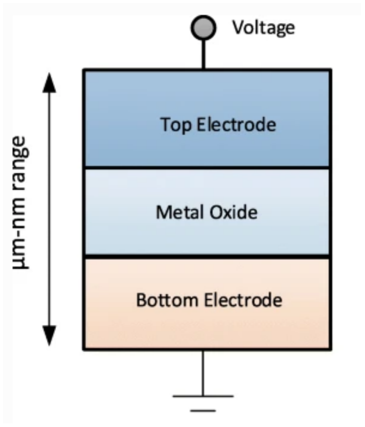


Figure 2.11: Basic structure of RRAM [57]

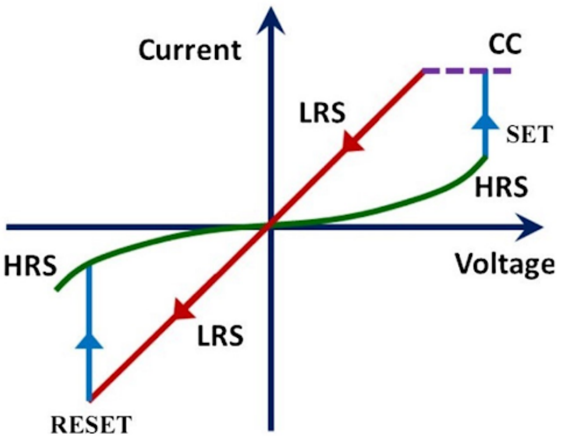


Figure 2.12: IV characteristics of SET and RESET process [44]

Crossbar Arrays

We are exploring this technology in this work as it has shown ultra-low power capabilities and lesser latency compared to conventional technologies. To implement convolution neural networks (CNNs), we need to understand the implementation of one MVM operation in RRAM.

An MVM is implemented by RRAM devices arranged in this crossbar architecture as a shown in Figure 2.13. Full parallelization of MVM can be implemented using these architectures because every cell has a meristor connecting the row line and the column line. If a column of memristors are set with resistances $R = R_1, R_2, R_3...R_n$, then their corresponding conductance will be $G = G_1, G_2, G_3..G_n$. If a set of voltages $V = V_1, V_2, V_3...V_n$ are applied to each of the row then a resultant current $V_i * G_i$ senses at each cell. The cumulative current (I) running in the line of each column is the sum of individual currents generated in that particular column. The current in each column, $I = \Sigma_i V_i \times G_i$ represents the dot product operation of vectors V and G as shown in Figure 2.13. In the case of CNN, it can be used for accumulating the convolutional output of one neuron in each bitline. This parallelism can be scaled to perform MVM in one step. Peripheral circuits are important to streamline the dataflow and interpret the analog output of the MVM correctly. In that context, the sample and hold circuit (SH) receives the bit current and passes it to a shared analog to digital converter (ADC) unit seen in Figure 2.13. The analog output is converted to digital representation when transferring the results to other digital units. Likewise a digiti to analog converter(DAC) also converts digital inputs to analog values when applying to each row.

Implementing and scaling crossbars can have operational issues because of the non-idealities in RRAM. Hence it can lead to functional issues leading to very less reliability than conventional technologies. Some of the reasons for these non-idealities are process variation in fabrication and computational process. Repeated correcting mechanisms to the architecture results in high read/write

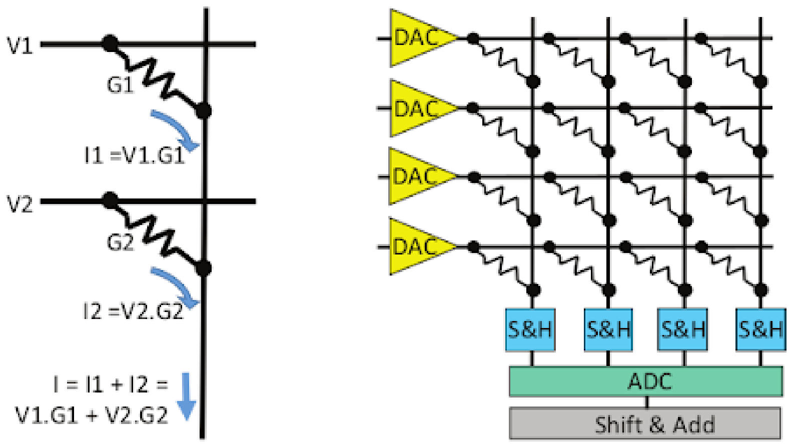


Figure 2.13: Crossbar array structure of RRAM to implement MVM [47]

operations which can reduce the system performance. Some of the error correcting mechanisms like error correcting codes, parity bits can be employed but results in higher energy consumption. Extensive research is ongoing in modeling the non-idealities and developing the countermeasures. The architecture in section(ISAAC Architecture) has considered these non-idealities and proposed a good architecture to realize CNNs close to real world implementations.

ISAAC Architecture

From the previous sections, we have some basics of the emerging computation-in-memory technologies, principle and working of a memristor based RRAM device and MMV implementation on RRAM devices. However, to implement these operations and execute neural networks, we need a layout or framework which integrates the crossbar array, peripheral circuits and other processing elements for intermediary functions. Such a framework is presented in ISAAC [47] architecture to design a full fledged accelerator based on crossbars. Also it comprises all the analog and digital components required for a CNN accelerator. This is one of the widely accepted architectures to design CNN accelerators for various applications. The energy, latency and area benchmarking of CNNs dealt in this work use this architecture. The Figure 2.14 provides an overview of the ISAAC architecture.

The proposed ISAAC architecture comprises multiple tiles at a high level connected with on-chip mesh. Each tile consists of eDRAM buffers to store inputs and a number of multiple in-situ multiply-accumulate (IMA) units. Further it has output registers to aggregate results, all the components in tile are interconnected with a shared bus. The tile also has shift-add units, ADCs, DACs and crossbar arrays to facilitate MVM. To run an inference on this architecture, the dataflow starts from the I/O interface connected to the first tile starting the inference. These inputs are fed to appropriate IMAs to perform MVM . These results are sent to ADCs and aggregate result on shift and add units. It completes the result of a convolutional layer in CNN following these sigmoid and Max pooling units are applied. The final result is sent to the eDRAM buffer to prepare it as input for the next layer. This result propagates similarly through multiple tiles and IMAs till the final layer spits the output. Such hierarchy of chipset / tiles /IMA can easily organize the data flow between processing elements with least data movements and better resource utilization.

2.4. Hardware Compression Techniques

This section focuses on the compression techniques used for heavy parameter CNNs used for optimization in energy consumption.

2.4.1. Pruning

Neural networks considered in this work are very deep and large. These large networks would require more storage size, data access time, energy consumption and computational resources. In case of embedded / mobile based applications these constraints will have serious design challenges in

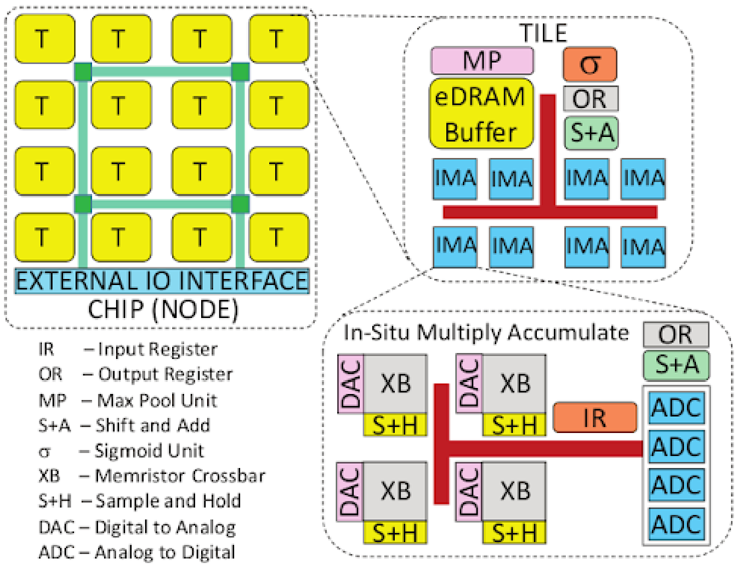


Figure 2.14: ISAAC Architecture [47]

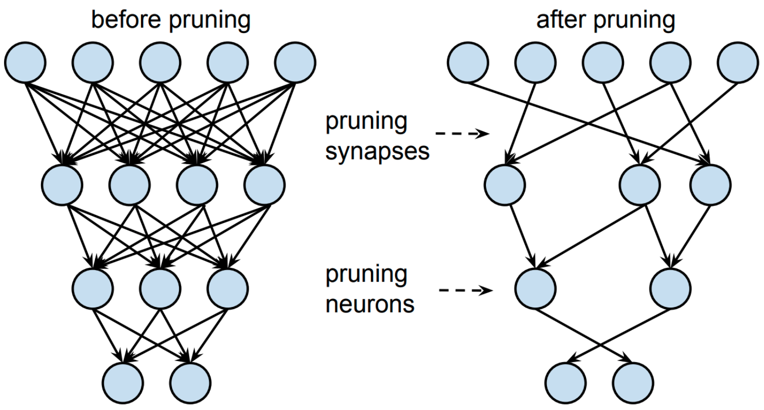
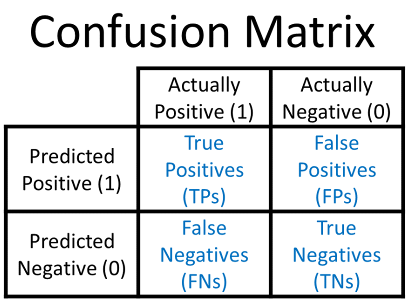


Figure 2.15: Basics of Pruning [30]

resource allocation. Pruning techniques have been instrumental in these situations where it reduces the model size significantly. This technique is an iterative process of removing in-effective weights / connections and neurons in the network followed by re-training the network to preserve the accuracy. We can implement multiple schemes based on our target application to decide a pruning threshold to remove a particular connection or neuron. The Figure 2.15 showcases before and after training architecture of the pruned network. However, to exploit the pruning results on the hardware, it should be able to skip removed connections while running inference on the network. Most of the conventional CPU, GPU and TPU benefit only from size reduction. An inference engines which can efficiently manage sparsity in the model is required. Further the implemented schemes will be discussed in a later section of the report.

2.4.2. Quantization techniques

Quantization is nothing but mapping a 32-bit floating point to a lesser bit representation. Millions of parameters in a model when mapped to lesser precision, It offers a smaller model footprint, reduced memory requirements and faster computation. There are few quantization methodologies to map the values, in this thesis we discuss uniform quantizer


Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 2.16: Confusion Matrix [3]

Uniform Quantizer

For a given floating point with range (x_min, x_max) which needs to be quantized to the range of $(0, 2\hat{}b - 1)$ levels, where b is bit-width. We need to define two quantization parameters : Scale(Δ) and zero-point (z). Given these parameters, the following equations can be used for conversion,

$$X_{int} = round(x/\delta) + z$$

$$x_Q = clamp(0, 2\hat{}b - 1, X_{int})$$

$$Clamp(a, b, x) = a, x <= a$$

$$Clamp(a, b, x) = x, a <= x <= b$$

$$Clamp(a, b, x) = b, x >= b$$

For de-quantization,

$$X_{float} = (X_Q - z) * \delta$$

Besides these we have other quantization methods such as Uniform symmetric quantizer and Stochastic Quantizer. But uniform affine quantizer is widely used in CNN applications. The quantization parameters δ, z are identified by solving an optimality problem to preserve the original value. While quantizing a model the (min, max) range is identified by filter / channel values then the derived δ, z values will be applied to all the values in the filter / channel. Intricacies in quantizing a model for inference will be detailed in the later chapters. Most importantly, bit-width is also a hyper-parameter in quantization which can be conFigured based on the application and the available hardware resources.

2.5. Performance Metrics

We have used standard performance metrics for CNN models and hardware benchmarking to evaluate the performance of the proposed schemes. CNN algorithms are commonly evaluated on accuracy, confusion matrix, F1-score, AUC-ROC and kappa score. Hardware performance is evaluated with power or energy consumption, area of the target device and run time for inference. The following section explains the metrics used in the thesis,

- **Accuracy** - This is estimated by comparing CNN's prediction of an image with its ground truth which is calculated as the ratio of correct predictions to total number of images.

- **Sensitivity and Specificity** - Sensitivity indicates the ability to classify True Positives (TP) and minimize False Negatives (FN). Sensitivity is the ability to identify the True Negatives (TN) and minimize False Positives (FP). They are calculated using the below formulas,

$$Sensitivity = \frac{TP}{TP + FN}, Specificity = \frac{TN}{TN + FP}$$

  – True Positive (TP) - Total number of images identified as Positive when their ground truth is also Positive.

  - True Negative (TN) - Total number of images identified as Negative when their ground truth is also Negative.
  - False Positive (FP) - Total number of images identified as Positive when their ground truth is Negative.
  - False Negative (FN) - Total number of images identified as Negative when their ground truth is Positive. These should be as low as possible because a 'miss detection' can have serious implications in a medical application.

- **F1- score** - It is a better representation of sensitivity and specificity, especially in cases of unbalanced data in all the classes in a classification task. In the course of this thesis, this metric will be recorded besides sensitivity and specificity and the formulae to calculate F1-score is,

$$F1 = \frac{2 * TP}{2 * TP + FP + FN}$$

- **AUC-ROC** - AUC-ROC is a graph plotted between sensitivity and (1-specificity). Area under the curve values has a (0,1) and values close to 1 represent better capability of the algorithm in classification tasks. Mostly this representation is preferred in binary classification tasks as we can vary the threshold to classify the output probabilities of the algorithm.

- **Cohen Kappa Score** - Cohen Kappa score is a statistical metric used in multi-class classification to measure the agreement between two graders. The novelty of this metric is, it also considers the probability of occurrence by chance. This appears to be a more vigilant method to evaluate the model in our case. The range of these values are (0,1), a negative value is also possible to indicate that there is not relation between the graders. The formulae of this metric is as follows,

$$K = \frac{P_o - P_e}{1 - P_e}$$

- **Confusion Matrix** - Besides all these metrics, it is a very intuitive representation of the performance. This essentially is a table with ground truth as their column headers and predicted label as its row header. This is very commonly used in both binary and multi-class classification tasks where each cell value is calculated as the number of predicted images of class as its column header when the ground truth is its row header. The Figure 2.16 represents a confusion matrix for binary classification. Similar matrix can be made for multi-class classification.

3
Related Works

3.1. Premise on DR classifiaciton

Automatic detection of DR severity is a popular research topic. It has been extensively researched for almost a decade. The rise in applicability of machine learning in day to day utilities has stirred the research in this domain to use its advantages to detect DR. Automatic detection of DR is commonly accomplished as a binary classification task of (non referable DR and referable DR as mentioned in 2.1) or 5-class classification as per international standards [49]. Besides these, efforts are also put into detecting lesions and other micro or macro features causing DR. Some of the pioneering works using CNNs [29],[58],[19],[27] have shown us evidence for this classification task. Prior to this, researchers have used hand crafted filter designs [28],[22],[43] and image processing techniques to extract retinal blood vessels to detect lesions causing DR.This approach is also widely exploited using CNN based techniques [41],[21],[18] to auto detect lesions and features related to DR for its classification. In a classification task, high sensitivity and specificity is very much a criteria for medical applications, as we cannot afford too many false negatives which can turn out to become life threatening conditions. Most of the research efforts [19],[56],[45],[40], focus on achieving high performance metrics for this task.A variety of CNN topologies including hierarchical CNNs, CNNs cascaded with variety of classifiers [27], [46] and ensemble approaches [32],[5] are developed to deliver high performance. Identification of micro features for mild and moderate class makes it challenging to achieve deployable results in multi-class classification [37],[56]. Hence commonly, binary classification is widely used in screening programs for early detection which suffices the purpose. Currently, we are witnessing FDA approvals for an automatic DR detection algorithm [48], [29]. The research in this domain has taken a leap forward from a research based environment to delivering those efforts in a real world setting.

We are currently seeing a shift in the focus of research and development in this domain towards improving the algorithms to support accessibility so that it is quickly adapted and widely accepted. Some of the works [53], [37] reported validation of CNN based algorithms in a hospital setting. Not only the utility of CNN algorithms but also the manufacturers of handheld devices have been reporting [50] promising results of their clinical studies. While the CNNs to detect DR are showing the state of the art results, different imaging methods and reduced image quality in handheld devices have its challenges to improve it further. It has been one of the active areas, since the deployment of handheld devices into the field. Some works [34] show innovative techniques to transfer the existing learnings of CNN based detection in DR for images collected on handheld devices. There is a little or no evidence on the system level aspects of the device as most of these use cloud APIs to deploy intelligent services. Given the premise of the research in DR detection, we discuss the strengths and opportunities in the literature below. We have not included lesion detection based classifications due to unavailability of huge datasets with pixel level annotations.

3.2. DR Classification

RishabGargeya1 et al [27] proposed a CNN based scheme cascaded with a decision tree to identify the images belonging to DR (1,2,3,4 classes). The overview of their architecture to implement this data driven implementation in Figure 3.1. They have used EyePACS private dataset for training with

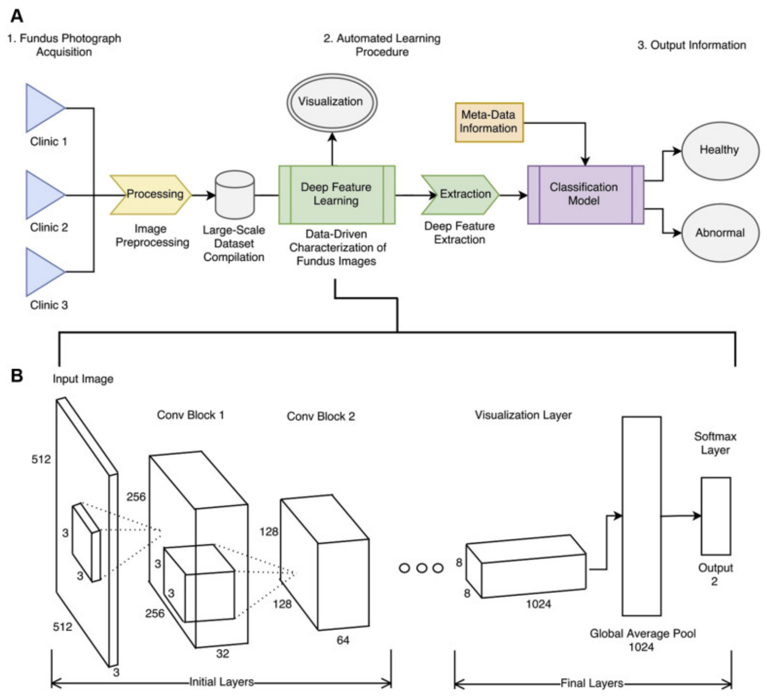


Figure 3.1: CNN based DR detection flow used in [27]

labels generated by medical experts. Each image will have a label of ‘0’ or ‘1’, where ‘0’ corresponds to no DR and ‘1’ corresponds to 1,2,3,4 classes. The CNN used here is a custom developed residual network. It is cascaded by a gradient boosting tree. Gradient boosting tree receives a vector of 1027 to provide the classification output. It has achieved 0.97 AUC with 94% sensitivity and 98% specificity in training. They have used Messidor-2 datasets [23] for evaluation of the model which resulted in 0.95 AUC score. A 93% sensitivity and 98% specificity is reported for Messidor-2 evaluation. With these acceptable accuracies, authors have further tested its performance to identify mild stage DR which gave an output of 74% sensitivity and 80% specificity with 0.83 AUC value.

Gulshan et al [29] primarily developed models for identifying referable DR((rDR) -> 2,3,4 classes) and vision threatening DR(vtDR -> 3,4). It is trained on approximately 1,20,000 images and evaluated on the Messidor-2 dataset [23]. It achieved 86% sensitivity and 98.4% specificity for rDR and 87.8% sensitivity and 98.2% specificity for vtDR on the Messidor-2 dataset. They have trained a single network (InceptionV3) to output multiple binary decisions to decide rDR, vtDR and fully gradable. Transfer learning is adapted for faster convergence in the model outcome. As a result, an ensemble of 10 models with binary decisions of various aspects will generate the final outcome. The output of this model is defined as the linear average of each prediction in the created ensemble model. The training dataset used here is not publicly available, also they have adopted a hierarchical image annotation policy involving 54 licensed ophthalmologists based in the US with varied range in experiences. An intense data annotation paradigm is demonstrated to provide clean data with less artifacts. Besides that there are no alterations made to the data or the network in the training process. Their subsequent paper [35] reported by JonathanKrause, et.al, proposes that adjudication of image level grades by experts is important for better learning outcomes. It achieved 97.1% sensitivity and 92.3% specification of the Messidor-2 dataset.

ShaohuaWan et al [56] work exploits the conventional neural networks like AlexNet, VGGNet-s, VGGNet-16, VGGNet-19 and Resnet on the EyePACS - Train (35126 images) dataset. These are CNNs similar to inceptionV3 and DenseNet121 with a different set of convolutional layers stacked in an unsymmetric fashion. Due to the noisy data and the limited number of data sets, normalization schemes and data augmentation are adopted to preprocess. Also, the fundus images are cropped to a smaller size in order to eliminate the extra areas. Non Local Means Denoising (NLMD) presented by Buades [20] et al is used in pre-processing the image to standardize the input image. Also classes were balanced using augmented images, each class created 20x images to balance with ‘0’ class. All

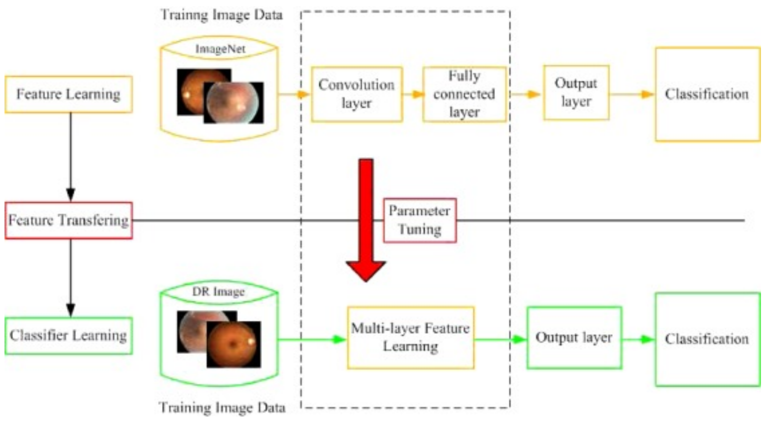


Figure 3.2: overview of the work in [56]

the models were downloaded and used with random initialized weights and transferable weights from ImageNet (shown in Figure 3.2). The training process is performed for 5-class classification on the considered dataset for all the models. Out of all the models, VGG-s network performed well with an accuracy of 95.68% accuracy.

Yi Zhou et al [58] intends to build a quality dataset as the publicly available datasets contain many inconsistencies and artifacts. They have built a dedicated dataset for DR with both pixel level and image level annotation. It is named as fine grained annotation for DR (FGADR), consisting of 1842 images with pixel level annotation, 1000 images are provided with image level annotations in addition. These efforts are accomplished by collaborating with medical personnel. To fully evaluate the quality of this dataset, they have performed benchmarking tests for multi-class classification. Benchmarking is performed by considering training a bunch of CNN models on merged EyePACS-Train and FGADR(1842 images) dataset. Finally they are evaluated on the (EyePACS-Test set, FGADR (1000 images)). The results on this benchmarking with their 5-class accuracy value are like inceptionV3(83.89%, 81.44%), DenseNet121 (85.39%, 83.49%), ResNet-50 (84.56%, 82.39%) and VGG-16 (83.63%, 80.43%). Other ensemble strategies and a bunch of CNN models also resulted in similar performances. The utility of this dataset is further evaluated by segmentation tasks which do not come under this scope of the thesis.

Recep E.Hacisoftaoglu et al [34] have reported their efforts to develop deep learning framework to detect DR for smart phone based retinal image systems. The goal is to find the correct field of view (FoV) to perverse the performance of the deep learning algorithm. The experiments were designed such that a bunch of CNN models are trained and evaluated on different sets of datasets. They have also merged different datasets to understand the influence of variety in performance of the datasets. The datasets used in this work are EyePACS-Train(35126 images), Messidor-2(1748 Images), UoA-DR (University of Auckland - DR) (200 Images), Messidor-1 (1187 images). Different combinations of these datasets are trained and tested on AlexNet, GoogleNet and ResNet50 architectures for rDR (2,3,4 classes) and vtDR (3,4 classes).ResNet50 have produced highest results in this search for better model. It achieved 98.6% accuracy, 98.2% sensitivity and 99.1% AUC in detecting vtDR. This performance is recorded on a merged dataset. It provides evidence that variation in train dataset results in qualitative models. Further to customize these models for smartphone based retinal image systems. The dataset is cropped at for percentages with the optic disk being the center to simulate the FoV of a smart phone based system. It is observed that the accuracy dropped from 98.6% to 83.5% when cropped to 60% of the image. Finally, it concludes saying that several challenges can occur due to limitations of imaging sensor systems, narrow field of view and computational power on smartphones. These have to be considered while designing algorithms for smart phone based imaging systems.

TaoLi et al [37], addresses the challenges in clinical deployment of high performing models in the literature. They fail to perform well because of the limitations in publicly available datasets. To solve this problem, they have developed a dataset called DDR which consists of 13673 images in collaboration with local hospitals in China. To evaluate this dataset, they have trained it on a bunch of CNN models including VGG-16, ResNEt-18, GoogleNet, DenseNet121, Inception. Out of these, inception provided better results but DenseNet121(74.31%) have performed well in mild and severe classes where these have less number of images. Further they have performed benchmarking for lesion detection which did not provide satisfactory results as multi-class classification. This paper shows serious concern for model generalization as the model accuracy is not a good representative qualitativeness. It also emphasizes the lack of confidence in diagnostic reports provided for detecting DR as what features resulted in the predictions remain unknown.

Rory Sayres, PhD et al [53] have conducted a comprehensive study to understand the influence of DR grading with some assistance to the grader. This is an extension of work reported by Krause et al [35]. They have conducted experiments for 3 scenarios 1) no assistance, 2) with grades only and 3) with grades + heatmap. The confidence of the grader increased with either of the assistance and the sensitivity increased significantly. Interestingly, grades+ heatmap had negative impacts, the graders overestimated some cases of No DR. But it helped in cases of DR. Overall,the study was conducted for 5 scale grading. Overall the increased transparency can achieve better results than model alone and grader alone. In other work [50], they have evaluated the utility of an offline AI algorithm for large scale deployment of handheld devices in remote areas.

Parshva Vora et al [55] have made efforts to implement binary classification of DR on an embedded NVIDIA Jetson board TX2. They have tried to implement a full scale trainable network for detecting DR. They have used GoogleNet CNN to detect DR. It is trained on EyePACS -Train and EyePACS-Test datasets. K-fold cross validation is implemented for 30 epochs on the embedded board. It achieved 76% of accuracy and no hardware metrics were reported. But this could be seen as a potential possibility for implementing DR on an embedded platform for deploying in remote areas. Similarly [22], [28] have mentioned hardware implementations for fundus images and not for detecting DR. They do not use CNNs in their implementations.

3.3. Discussion on related works

We have seen broad categories of work being accomplished in this domain, automatic detection of DR. From the literature, we can draw parallels between extensive development of high performing CNN networks and device development of cameras to make them accessible. We can see efforts lacking to integrate these research outcomes. Most of the successful CNNs developed needs heavy compute and only accessible through cloud APIs. The works mentioned earlier [29] are developed on huge datasets which are annotated through an intensive process using trained specialists and ophthalmologists. One can understand the requirement of huge data but similar resources are not practically available for anyone. These industry experts have not exclusively reported high sensitivity or specificity of multi-class classification. The reference standard available at the moment which is being followed by the researchers in this domain is to predict referable DR classes 2,3,4 with high performance. Also this meets the requirement of screening programs to identify DR. There is more research needed in improving multi-class classification performance. Some of the challenges to build such state of the art are unreliable annotations, high class imbalance and artifacts in publicly available datasets.

Second parallel is about device development, as [34] said we need to address many issues of down graded camera, low compute power while designing solutions for smart phone based cameras. Most of the existing CNN solutions for DR are captured from conventional table top devices, hence existing CNN solutions need to be customized for data being generated on the handheld devices. Model generalizations capabilities are required for quick translation of algorithms developed in a research or controlled setting into clinics.

Lastly, we have seen very little or no amount of research on hardware realization of these CNN implementations. The system level effective implementations needs to be discussed more with integrated development. Though internet dependency solves the issues temporarily. In a battery operated device and data intensive application, further optimizations need to be discussed. Because the energy consumed to process one image in some of the architectures is far less than the energy consumed for data transfers in the internet based implementation. Also for remote settings an ultra low power consuming accelerator will suffice the requirement of the device.

The global health crisis of DR can be only addressed with these accessible handheld devices which can deliver high performance in detecting DR. With the advent of optics and imaging devices, these huge fundus cameras are now transformed into handheld devices. To build a fully automated handheld screening device, we require an edge computing framework on a device with low power, faster and compact size while maintaining the performance.

4
Proposed Methods

4.1. Overview

We aim to propose an ultra low power accelerator to run CNNs on a handheld fundus cameras. Overall we try to address accessibility and usability aspects of these handheld devices. For usability, we require high performance outcomes from the CNNs used to auto-detect DR. To this we have added a new binary labelling scheme generated from multi-class outcomes of the CNN models. These labels increased performance but also preserves the outcome of a multi-class to assist the user with extra information for assurance. we have developed a vigilant model evaluation scheme to address the real world scenarios. For accessibility, the high performing models were considered for hardware compression techniques which are modified to give energy efficiency and less latency. We deployed our neural networks on embedded platform using open source tools and RRAM using its simulator. Finally, we have performed benchmarking on conventional and emerging technologies (CPU,GPU,mTPU and RRAM) for energy consumption and latency.

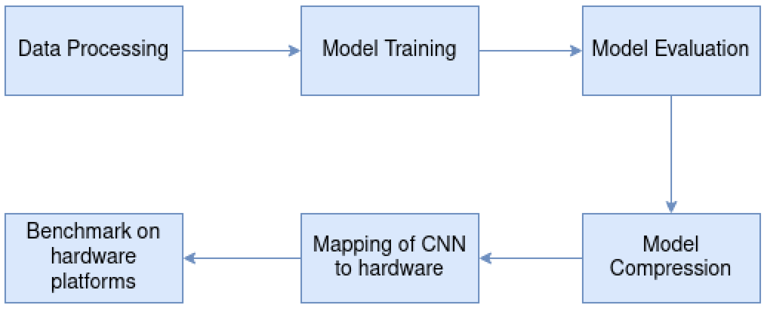


Figure 4.1: Overview of the development steps followed in this thesis

4.2. Data Processing

This section deals with the extra processing we implement on the datasets collected for classification tasks. Publicly available datasets are prone to containing artifacts and corrupt files. It is mandatory to run a sanitary check before using these datasets. In the datasets considered for this thesis, most of the files are of .png, .jpeg, .JPG. The corrupt files can be easily identified using a command line in the linux environment. The commands to do sanity check are pngfix and jpeginfo [8]. For the EyePACS dataset we have got around 30-40 corrupt files and the rest of the datasets do have corrupt files under 10 images.

CNN models which perform vision classification tasks have a fixed input shape, usually it is (224 x 224 x 3) or (299 x 299 x 3). The datasets contain images of varied size, ranging from 2k to 4k resolution. These files are resized to the target input shape before feeding to the networks. A simple algorithm runs to check for the borders of the retina against the black background in the x and y

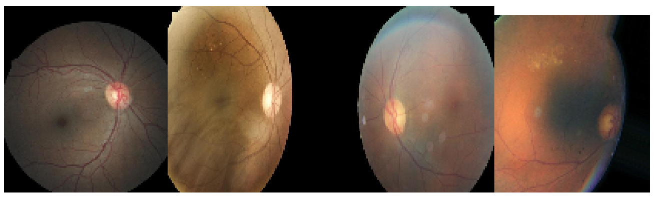


Figure 4.2: Datsets with aplied affine transforms on the resized image.

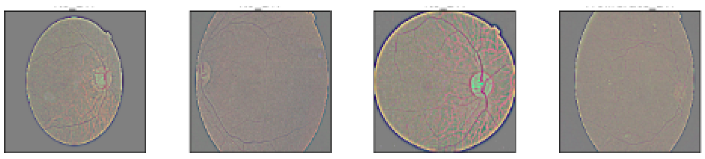


Figure 4.3: Gaussian filter applied on the retinal image

direction from center of the image. The x and y limits recorded are used to resize the image to target input shape.We preserve the information without cropping out in random scaling methods.

Python frameworks for vision classification tasks using CNNs apply data augmentation techniques. These are affine transforms applied randomly while accessing the image. These help in inducing more variation in the input data. The result of such transforms can be seen in Figure 4.2.

Lastly, a common image processing technique to apply gaussian filters for aggressive standardization. It is applied based on the size and quality of the dataset. This technique is only used in while implementing grid search method in this thesis. The results of these filters can be seen in Figure 4.3.

4.2.1. Small, Medium and Large datasets

We have collected 5 sets of datasets according to Table 2.2. Out of these, the Messidor-2 dataset is only used as an evaluation dataset and will not be included in any training process. Rest of the datasets are considered for CNN model development and fine tuning. Although literature has contributed to binary classification and multi-class classification, there are challenges found in reporting high sensitivity in mild and moderate classes. We have seen very little scientific work considerate about the generalization and reliability of their CNN model. In this work we demonstrate such evaluation schemes with different sizes of datasets. One of the huge challenges with public datasets is inconsistent labeling, poor quality of the images and unbalanced classes. These challenges are considered as real world noises which makes the model tolerate high variance. To explore the opportunities with public datasets, we have divided the datasets into small(S), medium(M) and large(L) sized datasets. They are merged in different combinations to generate resultant datasets. It eventually can be used to explain the generalization of the model. This splitting scheme is clearly explained in the Table 4.1.

EyePACS -Test and EyePACS-Train dataset have a huge number of images but most of it is due to the number of images in class 0. While merging the datasets we have randomly reduced the number of images in class with more than 10x imbalance with the rest of the classes. In that way, we benefit from better representation of the learnings from the data.

4.3. Model Training strategies

InceptionV3 and DenseNet121 have shown state of the art top-1 error on ILSVRC 2012 classification challenge. . This means that the bottom layers in the network are efficient in learning high level features in the images. So we have decided to train the top layer / Fully connected layer while freezing the rest of the layers to learn during training training process. This is achieved by configuring the output size of the Fully Connected layer to 5 classes for DR screening and followed by a softmax activation function. The modified CNN structure can be seen in Figure 4.4

Tag	Merge Datasets	Train Images	Validation images	Test Images
Small Dataset (S)	APTOS	0 - 1059 1 - 229 2 - 589 3 - 112 4 - 191	0 - 382 1 - 67 2 - 197 3 - 33 4 - 48	0 - 357 1 - 69 2 - 205 3 - 43 4 - 53
Medium Dataset (M)	APTOS + DDR	0 - 1775 1 - 598 2 - 1808 3 - 257 4 - 733	0 - 620 1 - 202 2 - 565 3 - 87 4 - 250	0 - 605 1 - 191 2 - 627 3 - 80 4 - 221
Large Dataset (M)	EyePACS Train, Test + APTOS + DDR	0 - 6021 1 - 4314 2 - 5992 3 - 1465 4 - 1887	0 - 2034 1 - 1432 2 - 1975 3 - 517 4 - 603	0 - 1945 1 - 1434 2 - 2033 3 - 522 4 - 627

Table 4.1: Small (S), Medium (M,), Large (L) Datasets

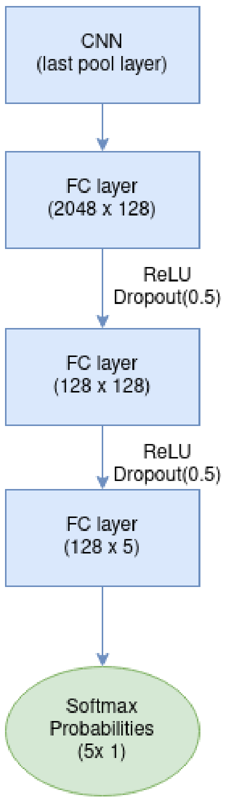


Figure 4.4: Modified neural network for training [7]

4.3.1. Grid Search

When we start to train an unknown entity, we do not have a starting point. Sometimes we can guess or do trial and error. We use grid search instead of using any educated guess. Grid search basically is a procedure to tune the hyper-parameters of the model. Each of the hyper parameter in the Table 2.3 have certain operating range of values for the network. Grid search essentially samples minimum number of values from each hyper-parameter and build sets of configuration with values sampled from each hyper-parameter. We can randomly pick a configuration from the sets of combinations and run the CNN model. This can be seen in the Figure 4.5 for 2 hyper-parameters. The results of the experiments will be qualitatively analysed to identify the region of interest. Now fine sampling or selection is performed in the region of interest. We continue until we get satisfactory results. Though it is a tedious process, we get an overall understanding on the behavior of the CNN model.

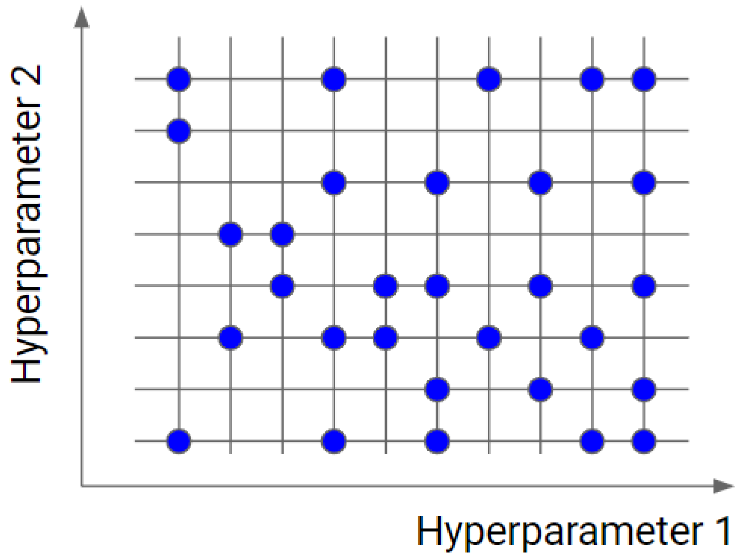


Figure 4.5: A grid search implementation with random sampling [7]

4.3.2. Manual Fine Tuning

The initial trials showed us that the CNN model could easily overfit because of the high imbalance in the number of images between each class. It can also result in variable validation/training errors as a result of over-estimated learning rate. These nuances can only be countered by manually tuning the entire training process. It is a constant iterative process of changing the model hyper-parameters (regularization, learning rate and dropout) to counter overfitting unless both the training and validation produce inter-twining curves for accuracy and categorical loss to reach a saturation point. The iterative process of manual fine tuning can be seen in Figure 4.6

4.4. Binary classification with Severity Label

One of the challenges in detecting DR into 5-stage classification is to classify mild, moderate classes with higher confidence which leads to marginal miss classification in the final grade. This eventually affects the overall performance of the model. During the process of training CNN models, some of our experiments showed that even though the model has less accuracy, we could visually appreciate that the model is learning the pattern and the mis-classifications happen marginally resulting in lesser performance. Also every model irrespective of the dataset size in training has clearly distinguished classes 0,1 and (2,3,4). We cannot settle for binary classification because of a positive output without any additional information which will have low confidence in the overall outcome of the device. Even tele-medicine, remote screening services preferred and favored some additional information with a binary label. This is similar to study reported in [53].

The sole purpose of this method is to maximize the sensitivity and specificity of binary classification of CNN models trained for multi-class classification. A binary classification scheme which can detect with high sensitivity and specificity and convey the level of severity with score using the softmax probabilities of the multi-class classification can be designed to address this issue. The severity score ranges between (0,1) where 0 being less severe and 1 being high severity. A severity score can be sufficient to an extent but for severity score ranges between (0.4 - 0.6) can still not be sure of its outcome. So we have added one more binary variable which indicates the difference in classification outcome of the less severe classes (0,1) and severe classes (2,3,4). So the proposed severity label(SL) is,

$$SeverityLabel(SL) = (B, I, s\_score)$$

Where, B, is the binary classification output, I, exclusivity index and s_score as severity score. As mentioned earlier, ‘B’ and ‘I’ are binary values and s_score ranges between (0,1). Pseudo algorithm for generating the severity label from softmax probabilities of multi-class classification is given in 1. To

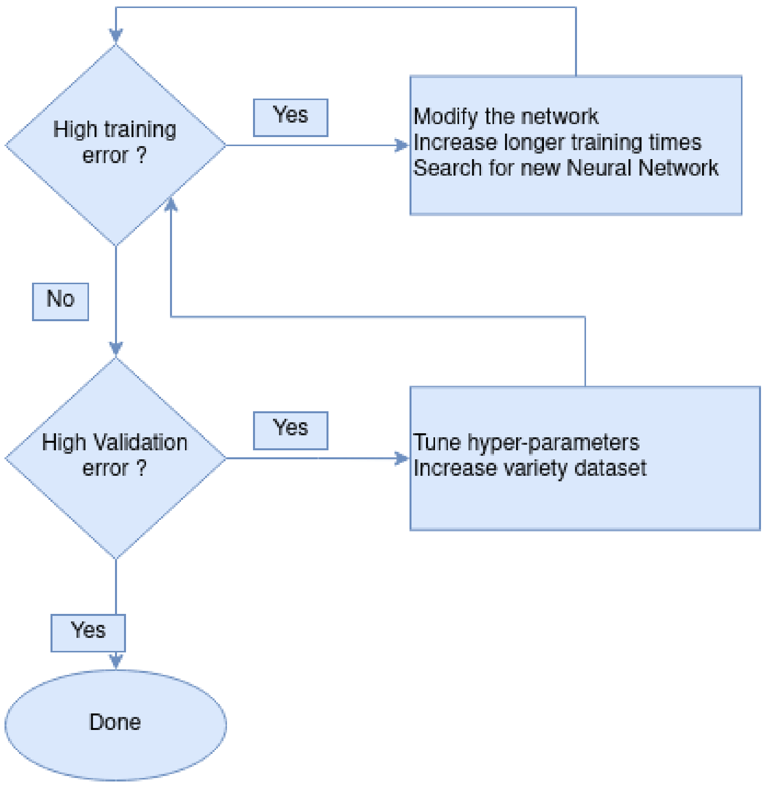


Figure 4.6: Manual tuning procedure

implement this scheme we need to set a threshold which is used in deciding the value for I .

When the softmax probabilities are available, the difference in the accumulated probabilities for classes 0,1 and 2,3,4 is calculated. For example, consider an input class label 1 and the softmax output probabilities are [0.21,0.33,0.4,0.03,0.04].The calculated difference value will be -0.06 is very marginal. The model output will be class '2' which is miss classified marginally whereas the severity label will be (0,1,0.47). The B value 0, conveys that it is classified into less severe class 0,1 and the I value 1 indicates that decision is very marginal and the user can verify with the severity score 0.47. In short, the patient has mild DR which is the ground truth. The utility of severity label is given in the Table 4.2.

This scheme has multi-fold advantages, 1) binary classification, 2) pseudo multi-class classification and 3) easy to interpret with high confidence on the device. The set threshold can be varied to further increase the performance based on the user setting. This labelling scheme can be very well used in the screening programs where the detection with high confidence is required.

4.5. Model Compression

In the scope of this thesis, we have explored compression techniques and lite weight networks to efficiently implement the software trained models onto the target hardware. These models are exploited with major constraints of low energy consumption and high accuracy.

4.5.1. Pruing Methodologies

Pruning compresses the network by increasing the sparsity in each layer of the network by a user-defined percentage of sparsity. The increase in pruning percentage not only decreases the model size but also affects the accuracy. To understand the effect on accuracy of the model, the model is pruned and trained for a range of pruning percentages.

We have implemented a method where the weights with lower magnitude get pruned. We implement this by following the process in Figure 4.7. First;y, all the neuron nodes are appended with a mask which represents the importance of that neuron in the final outcome of the model. We implement

Algorithm 1: Pseudo Code for Severity Label(SL) generation

```
Input: Exclusivity Threshold $Ex_{thresh}$,
       Softmax Probabilities of model trained for multi-class classification $soft\_max\_probs$,
$severity\_label$ : Severity Label generated from soft_max probabilities;
$s\_score$ : Severity score calculated from softmax probabilities for classes 2,3,4;
$sa\_score$ : safety score calculated from softmax probabilities for classes 0,1;
Set exclusivity threshold;
$Ex_{thresh} = 0.25$ ;
$severity\_label = None$;
$s\_score = soft\_max\_probs[2] + soft\_max\_probs[3] + soft\_max\_probs[4]$;
$sa_score = soft\_max\_probs[0] + soft\_max\_probs[1]$;
$difference = s\_score - sa\_score$;
if $difference > Ex_{thresh}$ then
    $severity\_label = [1, 1, s\_score]$ ;
else
    if $difference < Ex_{thresh} and difference >= 0$ then
        $severity\_label = [1, 0, s\_score]$ ;
    else
        if $difference >= -Ex_{thresh} and difference < 0$ then
            $severity\_label = [0, 1, s\_score]$ ;
        else
            $severity\_label = [0, 0, 0]$ ;
        end
    end
end
```

(B,I)	Inference
0,0,	No DR
0,1	Possibility of DR , check the severity score (<0.2 can be ignored)
1,0	DR found, attended immediately.
1,1	DR found with high severity

Table 4.2: Utility of the severity label

it by first pruning the weights with lower magnitude and train as it is implemented in model fine tuning process in section 4.3.2. We should be vigilant in choosing the hyper-parameters especially learning rate value. With a lower learning rate it will be difficult to learn from a lower accuracy in the successive steps. We have varied the learning rate in a small range around the value used in original training of the model. We prune the weights and try to compensate it by learning and we continue it until our desired percentage of sparsity preserves the original accuracy.

4.5.2. Post Training Quantization (PTQ)

We aim to implement only inference of the the model on the hardware platforms. We do not aim to implement training, so we can approach with the procedure shown in Figure 4.8. It is a building block to implement model quantization for varied precisions. In this technique, We first convert the model weights from float 32 bit to its lower bit precisions 4,8,16. This is achieved without training process by using the quantizer mentioned in section 2.4.2.1 to implement the conversion process. All the operations in the convolution layer are performed in the set lower precision bit. We do not prefer to change the bit pecisions of remaining layers as it results in aggressive quantization and effect the overall accuracy. For example, if two weight values of w-bit precision are multiplied, it requires the output to posses higher bit precision than w-bit so that the output value do not get clamped. It applies to activation layer. Since more than 75% of the operations in a CNN are convolutions, to preserve the original accuracy we do not quantize remaining layers in the CNN. In this way, we quantize a full model and retain the accuracy. In the scope of the thesis, we implemented W8A32(weight 8-bit,

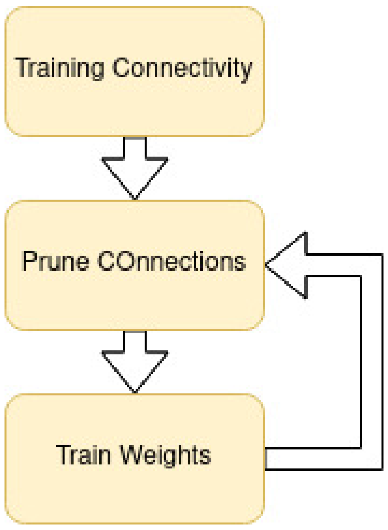


Figure 4.7: Training routine for pruning the CNN model

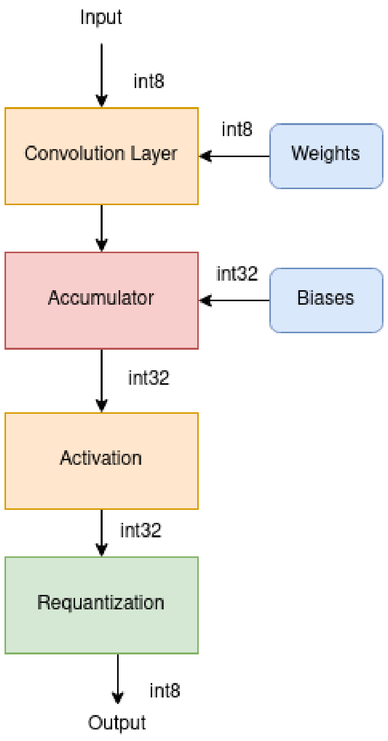


Figure 4.8: Flow chart of the post quantization scheme

activation 32-bit), W4A32 schemes where we already have the baseline for 32 bit implementation. One of the hardware requirement need activations to be quantized to 8 bits, so overall we have baseline, W8A8,W8A32 and W4A32 quantization schemes.

4.6. Implementation of neural network on RRAM based ISAAC architecture

CPU and GPU hardware platforms perform matrix vector multiplication using standard frameworks like tensorflow or pytorch. For TPU we have tflite frame work to implement int8 operations. CNN framework is built based on the dedicated accelerator architecture provided in ISAAC [47]. We have learned in section 2.3.1.2, it possesses all the digital and analog components required for implementing neural networks. Though the networks are deeper, we execute these CNNs layer wise. An individual framework for each layer with its weight parameters is designed and integrated while execution using ISAAC pipeline. This pipeline essentially manages the data-flow from one layer to the other. It has an aggressive approach of flushing the buffer as soon as the data to perform one complete operation is available to perform on crossbar. In that way we can have lesser sized buffers and faster throughput. Now that we have the pipeline to implement a full CNN, we will focus on the design parameters to be derived for implementing each convolution layer. Each layer in a CNN typically consists of convolution operation, max pool operation and activation function. The components to implement max pool and activation function are already included at tile level in ISAAC architecture. The convolution operation uses IMA crossbars to implement the inherent MVM. Firstly to perform MVMs, the weight parameters of the layer are to be loaded onto the memristive devices. A weight parameter in ISAAC architecture is 16-bit, if each memristor holds for w=2 bit width then a weight parameter requires 16/w memristors which are placed in the same row. When input is applied, the resultant output of a single weight parameter is received in multiple bilt lines which are adjacent to each other. All the filter parameters which convolve for the same output channel are expanded as a 1-D vector placed as a single column in the crossbar. This can help us calculate the required rows and columns in the crossbar array. DACs for each row provides input for every cycle and one successful operation requires 100 ns as mentioned in ISAAC. Mapping these deeper and long neural networks does help in obtaining efficiency when compared to conventional technologies. This technology can also be exploited with hardware compression and optimization techniques specific to RRAM. The hardware compressions mentioned in the section 4.5 will give further efficiency compared to its baseline implementation.

5 Experiments and Results

This chapter provides details on the various procedures and setup to implement proposed methods. Later part of this section explains the results obtained from various experiments. Firstly, the hyper-parameter search for tuning the CNN algorithm to generate the baseline model accuracy is implemented. It is followed by pruning techniques and quantization effects on baseline model accuracy and its design metrics. As we aim to explore the utility of emerging technologies for DR screening, we have considered various hardware platforms to compare and benchmark the performance of CNN algorithms. The information regarding the specification of the hardware platforms and other software tools used are also included in this section.

5.1. Setup

The specifications of CPU and GPU listed in Table 5.1 are taken from work laptop (Lenovo Legion Y540) Though we faced issues while working on this system with huge datasets and high resolution images, GPU offered good support to run a batch sizes of 4,8 with resized images under 512 x 512. We aim to build an accelerator for hand-held devices, it is only fair to draw comparison with current trending mobile accelerators for CNNs. We have used Google's coral dev board with Edge TPU (mTPU) board to compare results with later implementations of emerging technologies. Unlike, CPU and GPU it offers int8 implementation which is one of the hardware compression algorithms explored in this work.

RRAM - The hardware implementation on this platform is through simulation of the ISAAC [47] architecture. The basic unit of ISAAC architecture as mentioned in section 4.6 can be scaled to simulate the entire CNN model. The specifications mentioned in Table 5.2 for various components in this basic unit of ISAAC architecture are used to evaluate CNN implementation on RRAM based CIM. A simulation framework is built to estimate these values.

5.1.1. Software tools and Frameworks

Tensorflow (www.tensorflow.org) frameworks for ImageNet models are used in this thesis for fine-tuning and optimizing CNN models. A suite of sub-modules offered in tensorflow are used to perform pruning, quantization and lite networks. Some of the important tensorflow packages or APIs used in this thesis are tensorflow_model_optimization, prune_low_magnitured, tf.lite.TFLiteConverter and tf.lite.Interpreter . Anaconda3 (https://docs.anaconda.com/) package manager with python3.8.5 environment is conFigured to implement all the software dependencies in this thesis project. Nvidia

CPU	**GPU**	**mTPU**	**RRAM**
16GB RAM 9th Generation Intel® Core™ i7-9750H 6 cores,3200 MHz DDR4 memory	4GB GDDR5 - NVIDIA® GeForce® GTX 1650 896 CUDA cores @ 1485MHz	1 GB LPDDR4, [4] Google Edge TPU coprocessor: 4 TOPS (int8); 2 TOPS per watt	ISAAC Architecture [47] []

Table 5.1: Specifications of the target hardware platforms

Component	Power (mW)	Area (mm 2)
Input Register (IR)	1.24	0.0021
Output Register (OR)	0.23	0.00077
Sample & Hold	0.0000097	4*10 8
Shift & Add	0.05	0.00006
1 RRAM	0.00001831	0.152*10 8
ADC (8bit)	3.1	0.0012
DAC (1bit)	0.0039	17*10 8
Activation Function	0.26	0.0003

Table 5.2: RRAM hardware parameters for 1 unit as specified in ISAAC

software development kits (SDK), cuda-toolkit 10.1 and cuDnn 8.0 were installed to access GPUs with Tensorflow v2.4.1. Tensorboard, which is the in-built logger for tensorflow with a dashboard user interface helped in fine-tuning and tracking the progress of the tensorflow implementation.

5.1.2. Profiling Tools

In the scope of the thesis, we like to record energy consumption and latency for inferring an image on all the hardware platforms. As we have considered a broad spectrum of hardware platforms, we need to use hardware specific utilities for profiling the implementation.

Latency

Latency in this thesis is defined as the amount of time taken by a CNN model to produce inference on one image. The tensorflow framework includes an optional profiler which collects timing information about various events and CPU/GPU usage in model implementation. Besides the rest of the functionality in tensorflow profiler utility, it publishes the amount of time taken to evaluate a batch of images. This information when normalized for one image is recorded as latency for CPU and GPU. In the case of mTPU, we do not have this functionality in its tensorflow framework. Hence we depend on the date-time software package available in python3 framework to record start and end time to the millisecond resolution. The difference between recorded start and end time is divided with the number of images in batch to record the latency value on mTPU. The simulation framework for RRAM considers 100ns for each operation as mentioned in ISAAC architecture. The simulation framework with the hardware setup for CNN, dataflow between model layers and timing information provides the latency values for inferring on one image.

Energy Consumption

When compared to latency, recording energy consumption is a tedious job as it is not available directly but needs to be calculated by measuring available power consumption(P) details provided by intel and nvidia packages. We have used the s-tui [12] tool which monitors CPU's temperature, power, frequency and utilization. s-tui tool only provides command line access to run the tool and record power values with a unique time stamp. We can conFigure the rate (δ_T) upto 100 ms in s-tui to dump the power values in a csv file. Since there is no API available, s-tui is run as a daemon process while the tensorflow framework runs the inference on CPU. Similarly for GPU, we have a nvidia-smi [10] command line tool to record the power consumed by GPU at a specified rate(δ_T) to a csv file. This is also run as a daemon process while the tensorflow framework runs the inference on GPU. In both these cases, power traces are collected before and during executing the inference. The power traces collected before are averaged out and calculated as Idle Power (P_{idle}). This value(P_{idle}) is subtracted from the power traces collected during execution to compensate for other applications/factors running on the system. Now, the power traces are filtered with the timestamps recorded at start and end of the execution. To calculate the final energy consumption, accumulate each power trace multiplied by its rate(δ_T) at which it is recorded. Energy Consumption,$E = \Sigma_i P_i * \delta_T$.

On the other hand, for mTPU, its peak power consumption 2W [4] is used to calculate energy consumption with its latency information. This can be used for a fair comparison while benchmarking. Lastly, for RRAM, the energy consumption values are calculated as the process described in this ISAAC paper [47]. The simulation framework created with specifications in Table 5.2 along with the CNN MVM requirements provides the power values for inference an image. A detailed explanation on the procedure implemented in the simulation framework to calculate the energy consumption is provided in Appendix.

5.1.3. Datasets

The result of the proposed scheme to divide the existing datasets into small(S), medium(M) and large(L) datasets can be seen in Table 4.1. The table provides comprehensive information about train, validation and test division of the images available in each category. In rest of the thesis experiments and results, we use these datasets and category annotations excessively.

5.2. Experiments Performed

The experiments conducted can be broadly classified into three tasks. Firstly, we need to establish baseline CNN model accuracy for detecting severity of DR. This can help explore the possibilities of efficient translation of CNN models onto a variety of hardwares without losing performance of the model. Hence the second task is to evaluate the proposed compression techniques for hardware to understand their effect on model performance. In the third task we benchmark the model with compression techniques on the listed hardware platforms in section 5.1. These experiments will lead us to propose the best schemes to exploit the given hardware with high performance and efficiency.

5.2.1. Generating baseline accuracy

We use the proposed methods mentioned in section 4.3 to conduct these experiments. In scope of the thesis, we are only evaluating the ImageNet models InceptionV3 and DenseNet121 which are popularly used for vision classification tasks. These models best represent the current SOTA in DR screening as well as possess required depth in the network architecture to learn micro features present in the fundus images. To proceed further we need to set the values for each hyperparameter listed in the model training section 4.3. To establish a qualitative assessment in training with publicly available datasets we have performed model training with small, medium and large sized datasets. Thus we can obtain the intricacies and reliability of the trained CNN models. From the collected datasets as shown in Table 2.2, Messidor-2 dataset is not included in model training as it will be used as an evaluation dataset for models to demonstrate scalability and reliability. This is preferred as standard across the literature because the source claims that they have followed vigilant procedure while capturing the image. The rest of the datasets are merged into mentioned combinations to build small (S), medium (M), Large(L) datasets.

Multi-Class classification

This part of the section focuses on building a multi-class classification DR model with high accuracy.

Grid Search Firstly, we need to perform grid search with a set of values for each hyper-parameter as shown in Table 5.3. The grid search parameters for all the category datasets are same but they are trained separately. The range of each hyper-parameter in the grid search is shown in Table 5.3. During the grid search, we have initially chosen to freeze the bottom layers and only train the top layers during training. This gives us an intuition of the behavior of data with the CNN architectures. The results of the grid search for each dataset category is evaluated by tracking categorical loss and accuracy in the training process.

Manual Fine Tuning The complexity involved in datasets have demanded manual fine tuning for each CNN model. The results from the above grid search provides a qualitative hyper-parameter configuration to train the CNN model for its respective dataset. The train and validation datasets provided in Table 4.1 are used to perform the iterative manual fine tuning by changing the values of hyper parameters until a saturation is reached in training and validation accuracies. We can also modify regularization parameters and add dropout layers to the FC layers as seen in Figure 4.4. One of the major changes that we have incorporated in this process is to set all the layers in the CNN model for training and not consider pre-processing of the input data. With this process, we generate the trained Inception V3 and DenseNet121 models for the three categories of datasets in Table 4.1. These trained models are evaluated on their respective test dataset to generate model baseline accuracy. The re-

ImageNet Models	Inception V3, DenseNet121
Batch Size	8,4
Learning Rate	{0.1,0.01,0.001,0.0001}
Full trainable layers	**False**
Optimizer	SGD, ADAM
Epochs	20,40,60
5-stage classification	True
pre-processing	True, False
Weight Initialization	Transfer learning

Table 5.3: Hyper-parameters for grid search

sultant models after manual fine tuning IV3-S (InceptionV3(IV3) trained on small (S) dataset), IV3-M, IV3-L, DN121-S, DN121-M, DN121-L. These models are evaluated on their respective test datasets.

Binary Classification with Severity Label

The multi-class classification in the previous section has provided the softmax probabilities for each model on its respective test datasets provided in Table 4.1. These softmax probabilities are further processed to generate the Severity Label (SL) as mentioned in section 4.4. We calculate the accuracy, F1-score for each model to evaluate the proposed methodology. We group the multi-class labels of the test dataset into Non-referable DR 0,1 and referable-DR2,3,4 . These are used as ground truth to calculate the performance metrics. To evaluate utility of the exclusivity index (I) and severity score (s_score) in Severity Label (SL), we plot separate histograms of images present in each of the 5-classes,for Severity Labels (SL) (B,I)= 0,0,(0,1),(1,0), (1,1). This histogram plot provides more evidence for the qualitative evaluation of our method.

Evaluation for model generalization

This section of the thesis provides a qualitative evaluation of the trained models on a real world dataset which is not included in the training, validation and testing process. The trained models only execute forward path or inference on each image available in the Messidor-2 dataset. The labels provided in this dataset for each image are taken as ground truth to calculate the performance metrics such as accuracy and F1-score. Later, the softmax probabilities of these models are further processed to generate Severity Labels(SL) and calculate performance metrics for severity labels. The performance of the models in the three categories of experiments are considered to qualitatively choose a model to exploit with the compression techniques and used for deployment on target hardware devices.

5.2.2. Evaluation of Model Compression Techniques

We have used the Messidor-2 dataset for final evaluation and their corresponding training datasets for each model. It is applicable for both compression mechanisms listed below.

Pruning vs Accuracy

Pruning methodology explained in the section, is implemented using tensorflow framework and the API prune_low_magnitured . We consider 30,50,80 percentages of sparsity and record its accuracy for evaluation on the Messidor-2 dataset by a pruned model. These models are trained on the train and validation sets in large datasets (L). Similarly, the reduction in memory can be recorded by saving the model in a zip format. For a different pruning percentage, we plot the reduced size of the pruned model.

Quantization Schemes vs Accuracy

We have discussed various quantization schemes in section4.5.2, such as W8A8, W8A32, W4A32 to quantize a CNN model. These are implemented using the tensorflow optimization module(tfmot) .For W8A8, it is implemented using tensorflow lite interpreter on mTPU(Coral Dev board). This compression theoretically results in lesser memory, energy and improved latency for executing a model. In addition to that, it also sets lesser bit resolution of weights and activations, which can reduce the accuracy. This compression is applied on a pruned model. Hence the compounding effects of the compression schemes on the accuracy is observed by plotting for pruned models for the described quantization schemes.

5.2.3. Benchmarking on hardware platforms

One of the key contributions in this thesis is to perform benchmarking the CNN model for DR detection, on a variety of hardware devices involving both conventional and emerging technologies. This consolidated information is not found in the literature. The opportunities present in an offline automated DR detection system can be demonstrated if we can analyze latency, energy consumption, area and accuracy of the deployed models. We have already studied the accuracy of the model in the above sections and the benchmarking area is out of the scope of this work. Nonetheless, we plan to report the area information for the implementation of DR detection on RRAM devices. RRAM simulated framework generates the area value according to ISAAC architecture for the given CNN model. The following paragraphs explain the procedure to benchmark latency and energy consumption using the profiling tools (section 5.1.2). Most importantly, this benchmarking limits to provide results only for deploying the trained models for inference.

Experiments for Latency and Energy Consumption

These experiments are conducted on a bunch of CNN models which are considered in both floating point (32 bit) format and their corresponding quantized format (W8A8,W8A32,W4,A32). We have seen the poor accuracy performance of the W8A8 scheme, but we lack a hardware platform to implement the rest of the schemes. Since latency and energy is for implementation rather than performance, we report latency and energy values for the W8A8 quantization scheme; it can help as a standard for mobile/embedded platforms in our evaluation. These bunches of CNN models include InceptionV3(IV3), pruned InceptionV3 (IV3-P50), DenseNet121 (DN121) and pruned DenseNet121(DN121-P50). These models are the versions trained on large dataset(L) and pruning for 50%. The models in floating point format are implemented using tensorflow framework on CPU and GPU. Since mTPU (google coral dev board) does not have support to run inference on floating point models, the benchmarking results are reported only for quantized models in W8A8 format. On CPU and GPU we use the tf.lite package where the model is implemented in flatbuffer format for executing inference with quantized models. Each bench mark is collected by running the inference on the entire Messidor-2 dataset with a batch size of 4. For the sake of comparison all the values are normalized for one image which is energy (mJ/image) and latency (ms/Image). Latency and Energy consumption values are recorded according to the profiling methods mentioned in section 5.1.2. In the case of RRAM, the simulated framework provides latency, energy and area for a given network. Appendix have detailed analysis of RRAM simulation framework to generate energy and area.

5.3. Results Obtained

The results obtained for the above experiments are shown and discussed in this section.

5.3.1. Generating baseline accuracy

To deploy a model, we should posses high sensitivity and specificity. The experiments related to improve performance aspect are mentioned here.

Multi-Class classification

The efforts of grid search and manual fine tuning lead to training models with multi-class classification. Their outcome for the test datasets is provided here.

Grid Search - There are at least 20 different sets of hyper-parameters implemented for training on each dataset to understand the viable values to effectively implement the learning process. To start with, the maximum training accuracy across all the experiments is 65% for a small dataset (S) can be found in Appendix. whereas the accuracy plot is between (38 - 40) % for large and medium datasets. This is an indication that either it requires more training parameters or more data. Since, it is observed on both large and small datasets, using full trainable layers parameter set to 'True' for the rest of the experiments. This showed a better learning pattern than the performance in grid search. Although

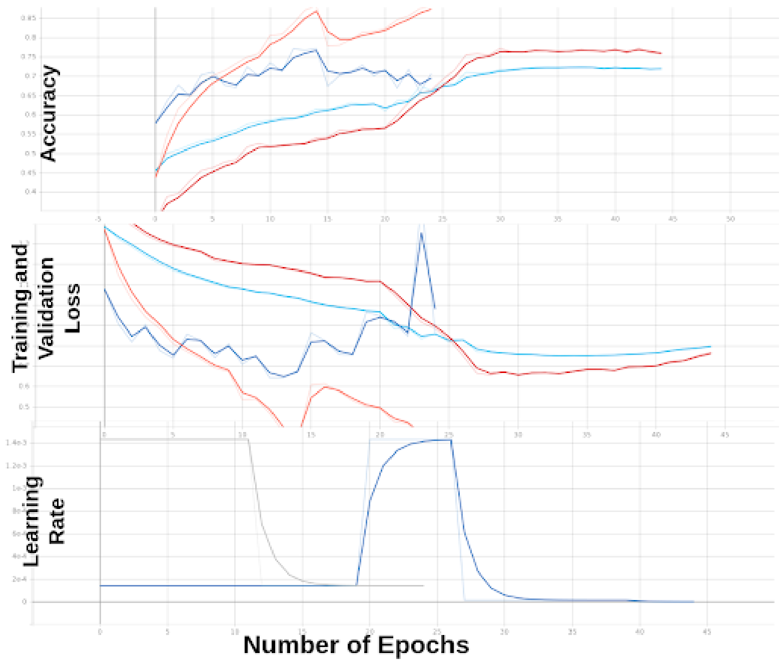


Figure 5.1: Training parameters of model trained on medium dataset (M) against epochs

it has less accuracies, some of the observations which helped the following manual fine tuning are Higher batch sizes converge faster and have smoother accuracy and loss curves. The learning rate values 0.1, 0.01 for both SGD and Adam optimizers have failed to converge. The 65 % accuracy for small dataset is due to 10x class imbalance and few hundreds of images in certain classes. The longer epochs help in understanding the saturation region of the model. SGD and Adam had similar effects on the learning process but their range of learning rates are different. SGD learns well starting from 0.001 and Adam at 0.00001 An ideal starting point for the manual fine tuning would be [weight initialization : Transfer learning, Optimizer: SGD, learning rate: 0.001, batch size : 8 and above (based on the hardware specifications), epochs: 20].

Manual Fine Tuning -

With the learning from the previous section, we train each dataset on the two models (InceptionV3 and DenseNet121) used for exploration. A successful completion of fine tuning majorly lies in setting the appropriate learning rate for different epochs. The output plots for a typical tuning task task can be seen in Figure 5.1. It is plotted for a medium dataset trained on Inception V3 and DenseNet121. Similar pots for small datasets and large datasets are included in Appendix. The plot is not a straight outcome but carefully adjusting the learning rate to avoid overfitting the data. So we have trained for longer epochs and iterated the tuning from a certain epoch to minimize the loss and maximize the accuracy. The plot shown in Figure 5.1 is the final path followed to produce satisfactory results. The sudden rise in the loss function for validation dataset is due to few extreme errors in classification but not a sign of over-fitting which can be inferred from the accuracy plot.

For the 6 experiments of 3 datasets, the results of the performance metrics are shown in Figure 5.2 . These metrics are generated on their respective test datasets as per Table 4.1. The labels of the x-axis represent the model and the dataset used, for example IV3-S : Small dataset trained on InceptionV3.

Firstly, InceptionV3 and DenseNet121 have produced similar results for a given dataset. The maximum accuracy is achieved by models trained on small datasets, which is 83.22% for DenseNet121 and 79.78 % for InceptionV3. At the same time, the performance metrics reduced drastically as we increased the size of the dataset. Some of the reasons randomly reducing the number of images from the dataset to eliminate 10x imbalance between classes. (from section 4.2.1). So the number of correctly classified images reduces which results in the accuracy drop. Less number of features present in the image especially in class 1. So there are more chances for marginal misclassification. Marginal differences while classifying / feature overlap between classes (0,1), (1,2) and (2,3). Artifacts present

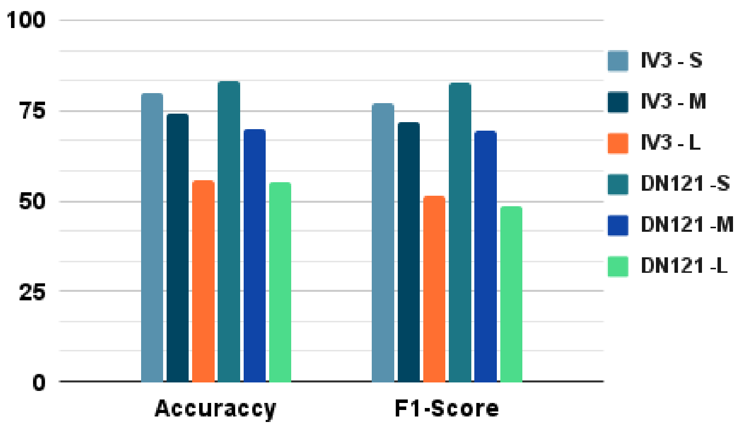


Figure 5.2: Performance metrics for Multi-class classification, Label Format IV3-S : InceptionV3 trained on Small(S) dataset

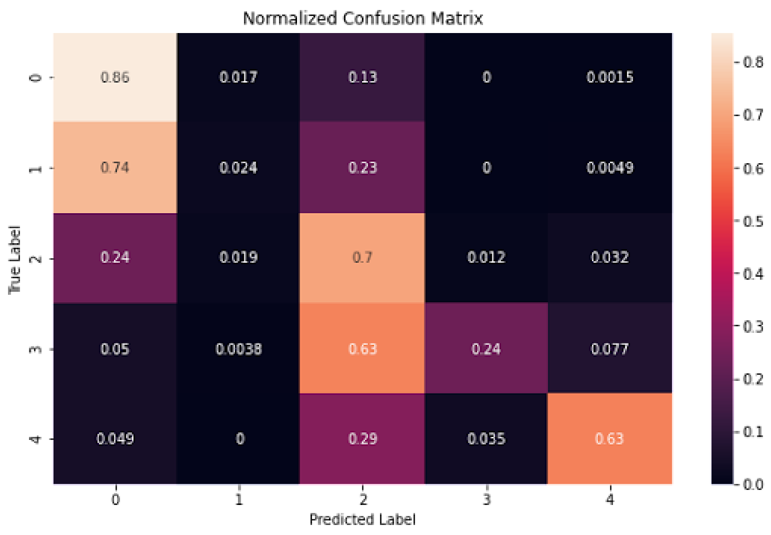


Figure 5.3: Normalized confusion Matrix generated for IV3-L model on test set of large dataset

in larger and medium datasets compared to small datasets.

Further Investigation with the confusion matrix shown in Figure 5.3 provides us information that models have shown satisfactory results in classifying classes 0,1,4 because of their rich features for classification. A clear differentiation / high specificity can be seen in learning mild/No severity classes0,1 and classes with moderate severity and above 2,3, 4. This confusion matrix belong to InceptionV3 model trained for large datasets.

Although we cannot use these in real life settings as they do not show higher sensitivity and specificity. The prediction probabilities from the model have useful information to convey the severity in a better way. The severity label (SL)scheme will be evaluated on these models in the next section to maximize utility of the models.

The Figure 5.5 shows the softmax probability plots of mis-classified inferences in each class - 2,3,4. We can clearly see that these plots have two peaks and these are distributed towards severity class. The decision making from these models can be less aggressive than using maximum of the softmax probabilities. The pattern matching probabilities of all the classes can be used in a better way. This analysis holds good for models trained on medium sized dataset and the effect of the CNN model characteristics such as depth of the network, deeper sub modulus and skip connections is not observed.

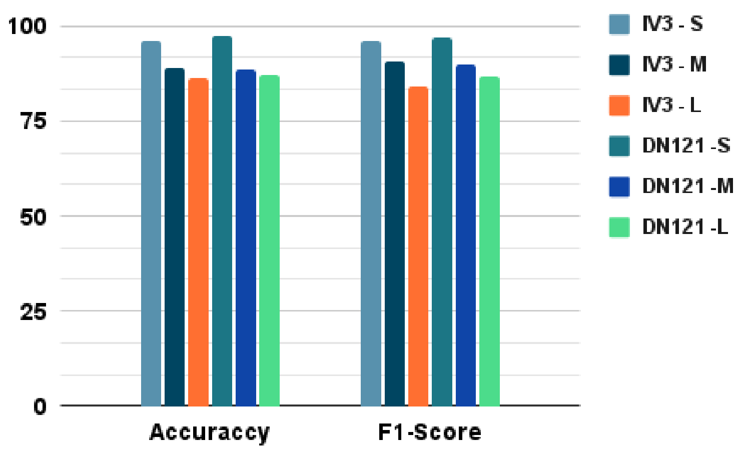


Figure 5.4: Performance metrics forSeverity label classification, Label Format IV3-S : InceptionV3 trained on Small(S) dataset

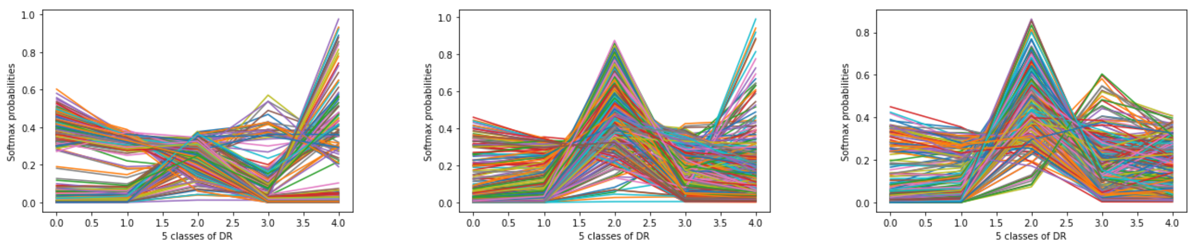


Figure 5.5: Softmax probability plots of mis-classified inferences in each class - 2,3,4 - from left to right

Binary Classification with Severity Score

The observation in the previous section shows that the model classifications can very well make use of this scheme. We have collected softmax probabilities from the trained models on all the datasets. The plot in Figure 5.4, compares the performance metrics of each model with the ‘B’ classification value in the Severity label. We can see there is a level up in all the performance metrics for this classification and a similar pattern to the multi-class classification. The models trained on the small datasets have shown higher performance, DenseNet121 reported 97.57% and Inception V3 reported 96% accuracy. The effect of dataset size is not seen as all the models produced satisfactory results i.e, the reliability F1-score is also above 85% for all which is comparable to the state of the art . This scheme is further evaluated to verify the outcomes of I value in Severity Label. The table 5.4 and Figure 5.6 provides more evidence of its utility.

In Figure 5.6, for each generated SL category, the histograms of images belonging to their ground truth label are plotted. This plot is generated for severity labels generated on softmax probabilities collected when the InceptionV3(L) model is evaluated on a large(L) - test set (6561 images). The confusion matrix can be seen in the Table 5.4. For ground truth labels 3,4 they are almost negligible in (0,0), (0,1) and (1,0) as expected. Similarly, less percentage of images belonging to ground truth labels 0,1 are found in (1,1) and (1,0). 526 images belonging to class ‘2’ are misclassified into severity labels (0,0) which are false negatives. Upon adjudication of these ground truth labels, we can see a

	0	**1**	**2**	**3**	**4**
0-0	1696	1119	526	29	17
0-1	100	112	151	16	23
1-0	62	68	153	14	18
1-1	87	135	1203	463	569

Table 5.4: Confusion Matrix for SL labelling scheme on Large(L) -Test evaluated with IV3-L

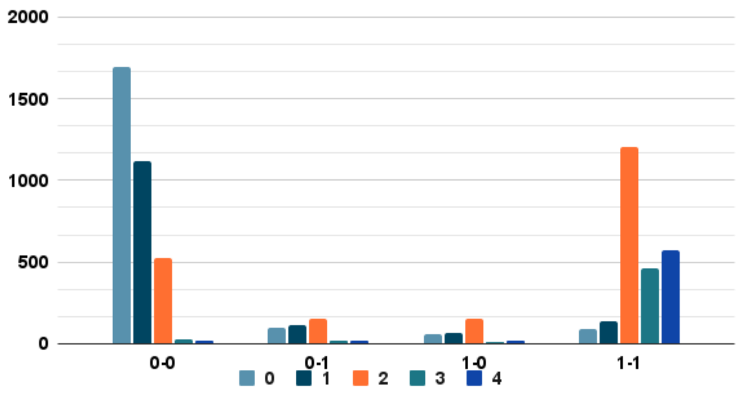


Figure 5.6: histograms of images belonging to their ground truth label are plotted in the category of its severity label(SL)

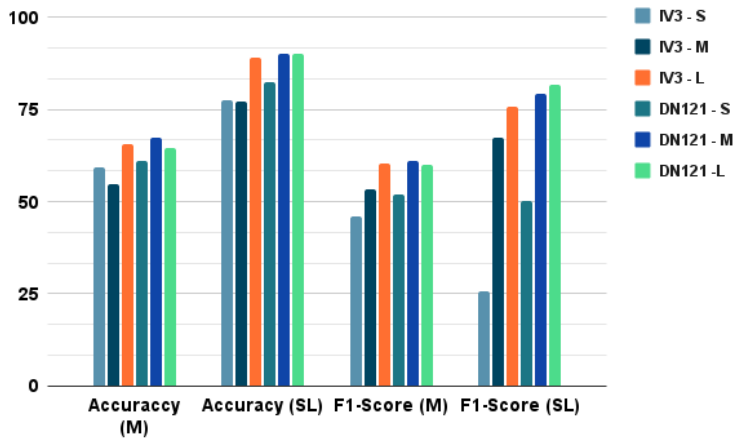


Figure 5.7: Performance metrics for multi-class classification(M) and severity label(SL), when evaluated on Messidor-2 dataset,Label Format IV3-M : InceptionV3 trained on medium(M) dataset

decrease in these false negatives. Besides that the severity label scheme has served its purpose in utilizing the high sensitivity and specificity in binary classification and utilizing multi-class probabilities as additional information for increasing reliability of the deployed device. We have not evaluated the utility of severity score because the value "I" confirms the range of severity scores. Also these performances are reported on test datasets which are part of datasets collected under similar environments. These can be further evaluated for model generalization.

Evaluation for model generalization

This is an extensive and comprehensive method of evaluating the model performances. The models (DN121-S and IV3-S) performing well so far in the earlier sections have shown significant drop in the performance. The F1-score for IV3-S only reported 25.7% which is not reliable for deployment. At the same time, models (IV3-L / DN121-L = 65.53% / 64.45%) have not shown any increase in multi-class classification accuracy, but have not dropped compared to their previous performances. Their corresponding Severity Label accuracies were promising (IV3-L / DN-L = 88.96% / 90.26%). IV3-L / DN-L models have shown that the model is resistant to real world noise and have learned from datasets with more variety. Large dataset is merged randomly from 3 datasets and also the random reduction in datasets to reduce 10x imbalance helps the models to learn weights in a reliable way and shows generalization for deployment. Though we have not seen state of the art performances in multi-class classification, we can proceed further on the evidence on performances with Severity Label. The

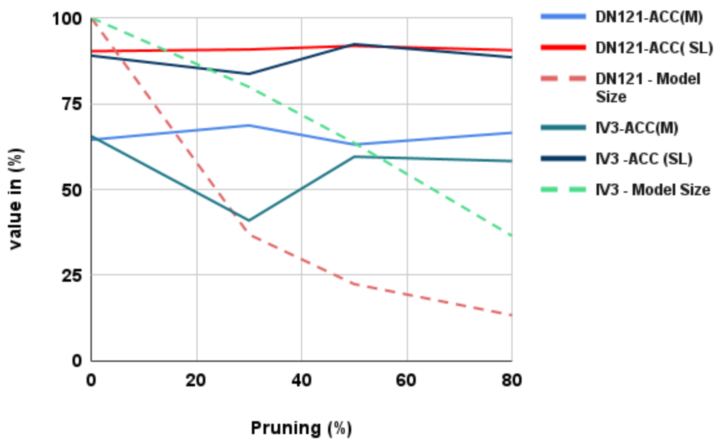


Figure 5.8: Accuracy plot with increase in pruning(%), label format IV3-ACC(SL) : InceptionV3 with accuracy values of severity label(SL), dashed lines represents % of model size compared to its unpruned model

hardware compressions in the following sections are demonstrated on the IV3-L and DN-L models.

5.3.2. Evaluation of Model Compression Techniques

The scope for optimizations in translating CNN models onto hardware can be found here.

Pruning vs Accuracy

One of the performance constraints we have is accuracy which needs to be preserved. The choice of pruning method and percentage can affect the accuracy. The Figure 5.8 demonstrates the implementation of IV3-L and DN121-L models against the 30,50,80 % of pruning conFigured in the pruning algorithm against accuray of the model. We have used a prune low magnitude algorithm to prune unnecessary weight parameters. The dashed line shows the model size which is scaled to the baseline size of the unpruned model. The effect of pruning can be seen in the decrease in model size with increase in pruning %. The DenseNet121 model has compressed more from its baseline than InceptionV3. It can be due to the fact that DenseNet models share collective information between layers in the same dense block and also faster learning is possible with residual connections across the network. The multi-class and severity label accuracy of the models were well preserved except for the dip at the 30% pruning in InceptionV3 models. Considering the slight margin in accuracy and less difference in compressed model size, we can consider 50% pruning percentage to be an optimal one for further compression of the model. Hardware platforms which can effectively manage the sparsity induced in the models can benefit with reduction in latency and memory usage.

Quantization Schemes vs Accuracy

In Figure 5.9, the plot shows the accuracy for each quantization scheme. Overall the accuracies are very well preserved and have not seen any significant reduction except for the W8A8 scheme. As we move from 8-bits to 4-bits there is a marginal decrease in the accuracy. Another important observation is that the performance of the IV3-L-P50(M) model had less accuracy (58.6%) at the W8A32 scheme than the (67.4%) W4A32 scheme. This can be due to quantizer where the 4-bit representations have a closer representation than 8-bit. In some cases it can be due to the stochastic nature of the quantizer while rounding off up or down. Besides that the severity label accuracies also did not vary where the edge platforms and emerging technologies can benefit with it to deliver high specificity and sensitivity. Lastly, the quantization scheme W8A8 has suffered severely when implemented on mTPU (Edge Coral dev board) whereas the rest of them are simulated ones. In addition to weights, the implementation of mTPU converts every variable to an int8 format. Here the input and outputs are also quantized along with weights and activations. As we can see, the weights are being quantized in other schemes but the performance did not drop as it is in W8A8. There is a cumulative decrease in the overall precision of the operations. Quantizing activations can also affect this accuracy . As a rule of thumb, we should have at least twice the bit precision of the weight parameters so that the multiplication of two int8 values

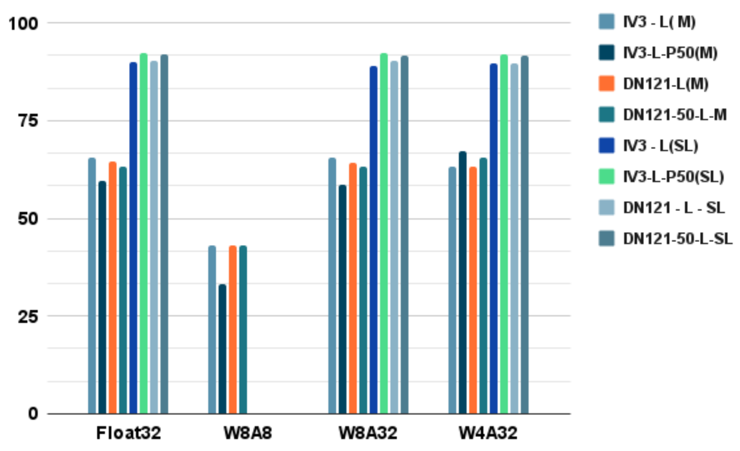


Figure 5.9: Accuracy plot for all the models compressed for different quantization scheme, Label W8A8 - 8 bit weights and 8 bit activations, IV3-S(M) : InceptionV3 trained on Small(S) dataset with Multi-class (M) accuracies.

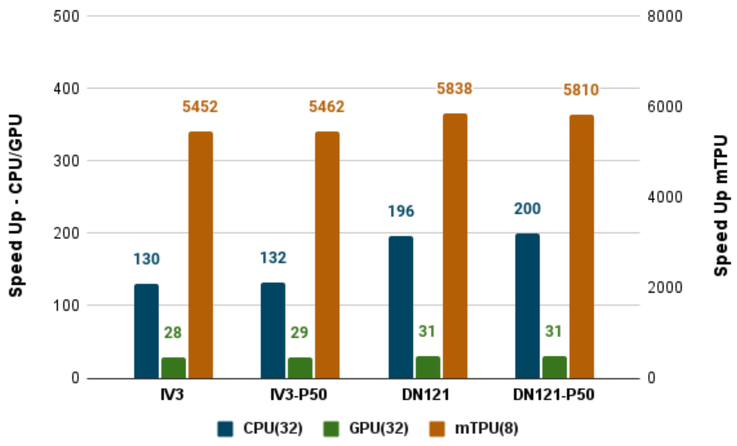


Figure 5.10: Speed Up in run-time achieved on RRAM (0.29 ms/image) compared to conventional technologies. Values plotted here are ratio of run-time(ms per image) on specific hardware to run-time(per image) on RRAM. Label Format DN121-P50 : DenseNet121 mmodel with 50% pruning

does not get clamped. It was not possible to evaluate severity label for W8A8 accuracies because the softmax values obtained here are in int8 format.

5.3.3. Benchmarking on hardware platforms

The below paragraph will provide the benchmarking results of latency and energy consumption conducted on the target devices.

Latency

Table 5.5 and Figure 5.10 demonstrate the latency information collected on CPU,GPU,mTPU and RRAM hardware platforms. In Fig 5.10, shows a speed up plot compared with corresponding RRAM implementation.The latency values are scaled w.r.t their RRAM implementation. Firstly, the latency of RRAM is majorly dependent on the size of the input and output which decides the number of times the given hardware to be used for generating a desired output. As we have a fixed hardware and the compression techniques used here have no effect on input and output shapes, the latency information of the pruned and unpruned model for RRAM did not change. The CNN models used here have high input sizes which eventually resulted in latency comparable to conventional technologies, we have aggressively exploited the parallelism offered by RRAM to increase considerable speedup around 30x on GPU and 150 x on CPU. It is also considerable to see a huge speedup when compared to mTPU,

	CPU(32)	GPU(32)	mTPU(8)	RRAM(32/8)
IV3	37.75	8.125	1581	0.29
IV3-P50	38.25	8.5	1584	0.29
DN121	56.75	9.125	1693	0.29
DN121-P50	58	9.125	1685	0.29

Table 5.5: Latency of benchmarks on hardware platforms (ms/Image)

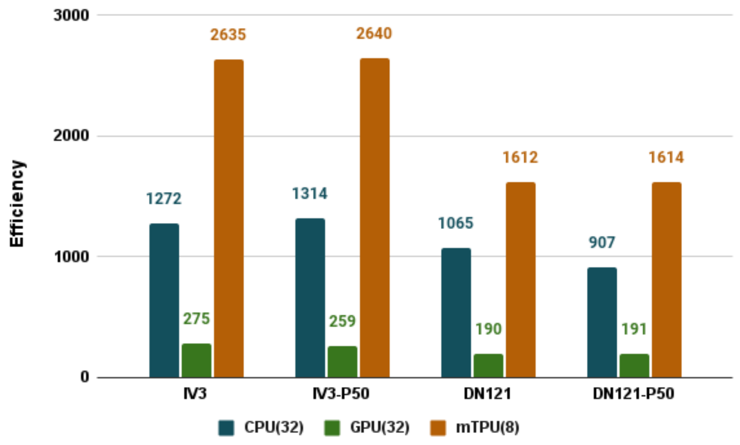


Figure 5.11: Energy efficiency achieved on RRAM (IV3:1.2 mJ/image / DN121:2.1 mJ/image) compared to conventional technologies. Values plotted here are ratio o energy consumed(per image) on specific hardware to energy consumed(per image) on RRAM. Label Format IV3-P50 : InceptionV3 model with 50% pruning

given its less compute memory, frequency, input size and number of weight parameters. This work [33] reported that for inceptionV3, it can process 17 images (150x150) per second. This value is concurring with the obtained results for input size (299x299x3). This values can also be affected by the experiment setup (power supply to the board, host CPU etc).

We have achieved the same RRAM latency values for DN121 and IV3 models with aggressive parallelization of layers closer to the input layer. DN121 required upto hardware scale up for 14 layers whereas IV3 had to be adjusted only for 3 layers. Importantly, none of the platforms have benefited from adopting pruning methodology because of their lack in utilizing random sparsity in the model. In the case of CPU and GPU, we can design intelligent software mechanisms to skip sparsity and reduce model runtime. For RRAM, we have to explore some structured pruning methodologies to reduce the crossbar sizes and benefit from hardware compression. The same holds for mTPU. We could not report latency information for CPU and GPU with their quantization schemes because of a lack of optimized software implementations. We have got values around 10 seconds for processing each image on GPU and CPU. This incompatibility of processors for various bit precisions have not permitted us to run 32 bit models on mTPU . RRAM has been promising in possessing compatibility for varied bit precision. This can be exploited in many artificial intelligence applications. Lastly, we have seen differences in hardware implementations of IV3 and DN121 models. The difference is very marginal in GPU, but it's significant in CPU and mTPU. IV3 model has higher operation intensity (operations / Byte) than DN121 models, In a von-neumann architecture, the memory transfers tend to add up to this latency as the number of times it needs to be accessed for completing the execution. So it explains for CPU and mTPU, for cases of GPU and RRAM, the model parameters are already loaded into processing elements or higher parallelization reduces that difference.

Energy Consumption

Figure 5.11 and Table 5.6 shows us the recorded energy consumption for running inference of an image on all the hardware platforms. In the Figure 5.11, efficiency is plotted by calculating energy consumption of a hardware implementation to its RRAM implementation. The results are promising as the minimum improvement of 200x is registered irrespective of a hardware platform. As similar to

	CPU(32)	GPU(32)	mTPU(8)	RRAM(32)
IV3	1526.89	329.54	3162	1.2
IV3-P50	1577.08	310.728	3168	1.2
DN121	2236.06	398.45	3386	2.1
DN121-P50	1904.39	400.55	3390	2.1

Table 5.6: Energy consumption of benchmarks on hardware platforms (mJ/Image)

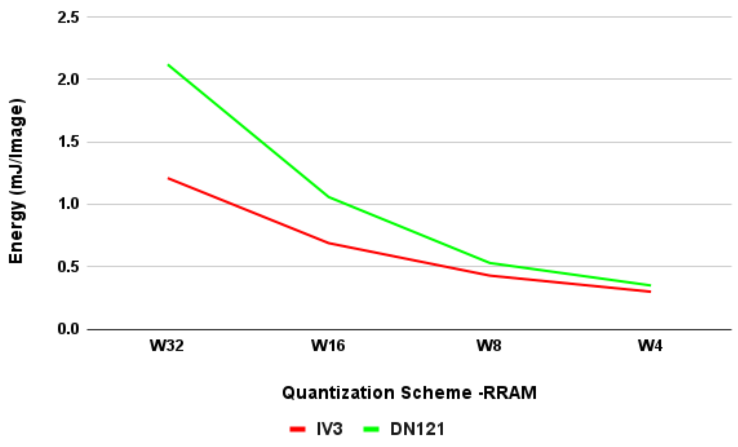


Figure 5.12: Energy consumption for different weight bit widths (32,16,8,4) in RRAM, Label Format W8 : weight values are quantized to 8 bits and IV3: InceptionV3

speedup, even energy consumption has not recorded any difference between a pruned model and unpruned model. As explained above, implementing structured pruning algorithms is necessary to utilize its advantages in RRAM and better sparsity managing capabilities in CPU and GPU. In the case of mTPU, we have seen incomparable (2640x in IV3 / 1612x in DN121) improvements w.r.t RRAM which provides strong evidence of its scope in intelligent systems with higher compute capabilities at ultra-low energy consumption. The more runtime required by the DN121 model has shown more energy consumption when compared to IV3 models. This difference in IV3 and DN121 implementation on RRAM resulted in different efficiency between CPU,GPU and mTPU implementations. In lack of compatible software implementations for 8-bit presion on CPU and GPU, we have not reported their energy consumption. Finally, with the Figure 5.12, RRAM energy consumption for different weight bit precision is plotted. The preserved accuracy as seen in Figure 5.12, can be used in this setting to obtain further reduction in energy consumption. W4 schemes have reported (03mJ/Image for IV3 and 0.35 mj/Image for DN121) which is 4x for IV3 and 7x for DN . The 7x improvement in DN121 models shows that the effect of more run time required by DN121 models has reduced with lesser hardware requirement in the W4 scheme. Using less bit precision reduces the required hardware area and consumes less energy with high preserved accuracy.

6 Conclusion

6.1. Conclusion

We have realized an ultra-low power implementation of CNN for DR detection using RRAM based emerging CIM technologies. At the same time, we have created a benchmark for energy consumption and latency to implement CNN based DR classification on CPU, GPU, mTPU and RRAM.

We have aimed to deploy accessible technologies for handheld retinal cameras and maximize its usability in the field. Firstly, we have addressed the challenges in model generalization where the literature reported high performing models fail when deployed in clinics. We have designed a qualitative model training and evaluation strategies. Merging public datasets into small (S), medium (M) and large(L) datasets have simulated real world scenarios for model training. The methods grid search and manual fine tuning for model training perform an extensive search in finding the correct hyperparameters to output a final CNN model with high performance. InceptionV3 and DenseNet121 CNN models trained on different categories of datasets have shown, models trained on small dataset (S) have high accuracy (IV3 79.92% :, DN121: 81.12%) when compared to models trained on large dataset (L). But none of these models exhibited high sensitivity and specificity comparable to state of the art or are deployable to clinics. The new labelling scheme 'Severity Label (SL)' with its 'B' values has increased the performance of these models with binary classification. The increase in accuracy for large(L) and medium (M) models show that the severity label is a better representation of softmax probabilities. The 'I' and 's_score' values in severity label not only assist the user with additional information to have assurance in outcome but also can generate pseudo multi-class classification. The final evaluation of the trained CNN models on Messidor-2 dataset have exhibited the quality of learning possessed by each model in training. The models trained with small(S) datasets performed poorly when compared to a large (L) dataset. It shows that variety in dataset simulating real world noise helped to train with better representations. It also holds correct for InceptionV3 and DenseNet121 showing no effect of characteristics of network architecture. This evaluation scheme is a testament of models generalizability.

The hardware compression techniques model pruning and quantization are implemented to efficiently translate CNN models onto hardware. Pruning was not effective in hardware realization majorly because sparsity introduced in the model was not translated effectively on all the hardware platforms. Quantization of weight parameters have shown that 4-bit precision also offers accuracy close to 32 bit precision. But its hardware realization had challenges on CPU and GPU due to lack of efficient software mechanism to perform operations at low bit precision. On mTPU, the W8A8 scheme resulted in very inefficient and poor accuracy. This is mainly due to aggressive quantization of input and activations. A suggested scheme for hardware realization is using W4A32. RRAM offered compatibility for all quantization schemes unlike rest for the hardware platforms. Mapping of trained CNN models to RRAM crossbar network is achieved through a simulation framework. Finally the benchmarking results show that RRAM implementation offered 130x, 30x and 5500x speedup when compared to CPU,GPU and mTPU respectively. Similarly, the energy benchmarking for RRAM realization showed close to 200x improvement irrespective of any platform and network architecture. Though inceptionV3 and DenseNet121 had similar performances on algorithm, inceptionV3 have shown better hardware performances. Lastly, the w4A32 scheme on RRAM has decreased energy consumption from 1.2 mj/image (32 bit) to 0.3 mj/image. The benchmarking have clearly shown that CIM based emerging technologies can be deployed to deliver high compute on the edge at ultra-low power consumption.

These implementations can be exploited to provide access to high quality medical care in remote areas. The requirement of high performance CNNs to detect DR at ultra low power consumption is demonstrated in this thesis. In addition to that we have also reported schemes to improve usability of the handheld camera. These kinds of devices will have quicker adaptation and wider acceptance.

6.2. Future Work

The literature on DR auto detection is extensive but in terms of hardware realization and system integration of CNN models there is a lot of scope for research. Especially for smart phones or handheld retinal devices. In that regard, we have following possible directions of future work,

Explore other methods for DR detection In this work, we have only discussed DR detection as a classification problem. But with datasets with pixel level annotations we can segment and classify, this is also one of the directions in which automatic DR detection is addressed. We have seen that InceptionV3 and DenseNet121 have similar performance for every test scenario. So custom neural networks can be developed with less number of layers which can result in lesser energy and area efficiency on RRAM.

Improve accuracy of multi-class classification Having a binary classification for non-referable DR 0,1 classes and refereable 2,3,4 classes will suffice for its current deployment. But as we progress further we need to improve performance detecting each class.

Mapping Sparse networks One of the failures in our experiments is to apply random pruning to reduce the weights and eventually benefit from less hardware utilization. But none of the hardware is capable of handling sparsity induced in the model. So an efficient way of handling sparse matrices or performing structured pruning needs to be implemented to benefit from this method.

Offline training and inference capabilities The promising results of emerging CIM based technologies as seen in this project gives evidence to be able to implement a full-trainable network on the devices. These solutions offer custom training for each individual device and avoids the challenges of imaging attributes like optics, sensor, image resolution etc.

Non-idealities of RRAM We have implemented a RRAM simulation framework to implement CNNs which is close to a real processor. But non-idealities in its implementation will affect the overall performance of the solution. The quantization noise, number of read/write to memory and process variations have to be addressed. A better modeling of these non-idealities can be used in future implementation of CNNs onto RRAM.

A
Appendix

A.1. RRAM simulation framework

Based on the design choices of weight bit width (w), and crossbar dimension, we try to fit layer weight parameters to calculate the number of crossbars required. Based on this we should estimate the number of units required for each circuit component in the architecture. We will be able to derive a specs sheet as shown in Table A.1. Based on this mapping of Convolution layer to for remaining components in the architecture we can estimate the energy, area and latency

Layer Name	**Layer Specs** **Layer shape ->[i,x,y,o]** **Input data shape[x,y]** **Output data shape[x,y]**	**RRAM specs**
conv2d_1	Layer shape [32,3,3,32] Input data shape [111, 111] Output data shape [109, 109]	288 - DACs 512 - S&H 64 - 128x128 crossbars 32 - 32 x 128 crossbars 96 - Input Registers 64 - 8 bit ADCs 32 - 7 bit ADCs 96 Output registers 96 Shift and Add 24 Activation units 24 Max pool units
FC_1	Layer shape [2048,1,1,128] Input data shape [1, 1]] Output data shape [1, 1]	2048 - DACs 2048 - S&H 256 - 128x128 crossbars 256 Input Registers 256 - 8bit ADCs 256 - Output Registers 96 Shift and Add

Table A.1: Mapping of first convolution layer (conv2d_1) and fully connected layer (FC_1) of inceptionV3 to RRAM crossbar .This mapping analysis is used for calculating energy and area

I - input channel, o- output channel, x - length of the filter, y - breadth of the filter

A.1.1. Energy Consumption

Energy is calculated by estimating power consumption of each circuit component to perform its function. First we calculate energy taken to do one operation, meaning from application of input to having the output in the buffer for one value. A read operation for cross bar is given as 100 ns, we demonstrate the energy calculation as below,

First convolution layer(conv2d_1) in InceptionV3

Circuit components	**0 - 100 ns**	**Circuit Components**	**100 - 200 ns**	**Circuit components**	**200 - 300 ns**
288 - DACs	1.125mW	64 - 8 bit ADCs	198.4 mW	96 Output registers	2.76 mW
512 - S&H	5uW	32 - 7 bti ADCs	84.4 mW	96 Shift and Add	4.8 mW
64 - 128x128 crossbars	19.2 m W			24 Activation units	6.24 mW
32 - 32 x 128 crossbars	2.4 mW			24 Max pool units	9.6 mW
96 - Input Registers	14.88 mW				
Total	45.48 mW		282.8 mW		23.4 mW

Table A.2: Detailed analysis to calculate energy consumption for conv2d_1 layer in inceptionV3 to produce 1 output value on a RRAM crossbar architecture

Total Energy / operation = 100 ns * 45.48 mW + 100 ns * 282.8 mW + 23.4 mW * 100 ns
Total Energy / operation = 343.805*10^-12 J
This implementation as to perform 109 x 109 such operations, so the total energy for this particular layer be,
Total energy consumption = 109 * 109 * 351.68 * 10^-12 = **4.084 uJ**

Fully Connected Layer in InceptionV3

Circuit components	**0 - 100 ns**	**Circuit Components**	**100 - 200 ns**	**Circuit components**	**200 - 300 ns**
2048 - DACs	8mW	256 - 8bit ADCs	793.6 mW	256 Shift and Add	7.36 mW
2048 - S&H	20 uW			256 - Output Registers	12.8 mW
256 - 128x128 crossbars	76.8 mW				
256 Input registers	39.68 mW				
Total	180.48 mW		793.6 mW		20.16 mW

Table A.3: Detailed analysis to calculate energy consumption for conv2d_1 layer in inceptionV3 to produce 1 output value on a RRAM crossbar architecture

Total Energy / operation = 100 ns * 180.48 mW + 100 ns * 793.6 mW + 20.16 mW * 100 ns
Total Energy / operation = 935.24 *10^12 J
This implementation as to perform 1 x 1 such operations, so the total energy for this particular layer be,

Total energy consumption = 935.24 * 10^-12 = **0.9352 nJ**

A.1.2. Area estimate

unlike,energy consumption, area is fixed value, whichi convey the amout of resources to fully implement the neural network. So any circuit component which is need even once in the whole model execution, we still have to consider it and estimate it. An area estimate to the above mentioned layers.

First convolution layer in inceptionV3

Circuit Components	**Area estimate in sq.mm**
288 - DACs	0.000047812
512 - S&H	0.00004
64 - 128x128 crossbars	0.0016
32 - 32 x 128 crossbars	0.00005
96 - Input Registers	0.00168
64 - 8 bit ADCs	0.0768
32 - 7 bit ADCs	0.026832
96 Output registers	0.00924
96 Shift and Add	0.00288
24 Activation units	0.0072
24 Max pool units	0.00576
Total	0.1321298 sq.mm

Table A.4: Area estimate for first convolution layer in inception V3

Fully connected layer in inception V3

Circuit Component	Area estimate (sq.mm)
2048 - DACs	0.0003399
2048 - S&H	0.00016
256 - 128x128 crossbars	0.0064
256 Input Registers	0.0044
256 - 8bit ADCs	0.3072
256 - Output Registers	0.02464
96 Shift and Add	0.00288
Total	0.34601 sq.mm

Table A.5: Area estimate for fully connected layer in inceptionV3

Bibliography

[1] Basic cnn. URL https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks

[2] Canon fundus cameras. URL https://www.laservision.nl/product/canon-cr-2-plus-af-funduscamera/.

[3] Confusion matrix. URL https://glassboxmedicine.com/2019/02/17/measuring-performance-the-confusion-matrix/.

[4] Coral dev board. URL https://coral.ai/products/dev-board/.

[5] Kaggle competition. URL https://www.kaggle.com/c/diabetic-retinopathy-detection.

[6] URL https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html.

[7] Grid search. URL https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/.

[8] Jpeginfo - sanit check for jpeg files. URL https://github.com/tjko/jpeginfo.

[9] mmeor wall. URL http://semiconductorexpert.blogspot.com/2014/07/bottlenecks-dram-moores-law.html.

[10] nvidia-smi, profiling tool for gpu. URL https://nvidia.custhelp.com/app/answers/detail/a_id/3751/~/useful-nvidia-smi-queries.

[11] Remedio. URL https://www.remidio.us/fop.php.

[12] s-tui, a command line profiler for cpu. URL https://github.com/amanusk/s-tui.

[13] Sight for all. URL https://sightforall.org/diabetic-retinopathy-initiative/.

[14] Topcon fundus cameras. URL https://topconhealthcare.com/product-category/fundus-cameras/.

[15] URL https://www.zeiss.nl/vision-care/voor-oogzorgprofessionals/producten-en-diensten/zeiss-instrumenten-en-systemen/visuscout-100-handheld-fundus-camera.html.

[16] Eyepacs test and train datasets from kaggle, 2016. URL https://www.kaggle.com/c/diabetic-retinopathy-detection/data.

[17] Aptos diabetic retinopathy dataset, 2019. URL https://www.kaggle.com/c/aptos2019-blindness-detection/data.

[18] Kemal Adem. Exudate detection for diabetic retinopathy with circular hough transformation and convolutional neural networks. Expert Systems with Applications, 114:289–295, 2018.

[19] Wejdan L. Alyoubi, Wafaa M. Shalash, and Maysoon F. Abulkhair. Diabetic retinopathy detection through deep learning techniques: A review. Informatics in Medicine Unlocked, 20: 100377, 2020. ISSN 2352-9148. doi: https://doi.org/10.1016/j.imu.2020.100377. URL https://www.sciencedirect.com/science/article/pii/S2352914820302069.

[20] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 60–65. IEEE, 2005.

[21] Piotr Chudzik, Somshubra Majumdar, Francesco Calivá, Bashir Al-Diri, and Andrew Hunter. Microaneurysm detection using fully convolutional neural networks. Computer methods and programs in biomedicine, 158:185–192, 2018.

[22] N.S. Datta, R. Banerjee, H.S. Dutta, and S. Mukhopadhyay. Hardware based analysis on automated early detection of diabetic-retinopathy. Procedia Technology, 4:256–260, 2012. ISSN 2212-0173. doi: https://doi.org/10.1016/j.protcy.2012.05.039. URL https://www.sciencedirect.com/science/article/pii/S2212017312003180. 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[23] Etienne Decencière, Xiwei Zhang, Guy Cazuguel, Bruno Lay, Béatrice Cochener, Caroline Trone, Philippe Gain, Richard Ordonez, Pascale Massin, Ali Erginay, Béatrice Charton, and Jean-Claude Klein. Feedback on a publicly distributed image database: The messidor database. Image Analysis Stereology, 33(3):231–234, 2014. ISSN 1854-5165. doi: 10.5566/ias.1155. URL https://www.ias-iss.org/ojs/IAS/article/view/1155.

[24] Javier del Valle, Juan Ramirez, Marcelo Rozenberg, and Ivan Schuller. Challenges in materials and devices for resistive-switching-based neuromorphic computing. Journal of Applied Physics, 124:211101, 12 2018. doi: 10.1063/1.5047800.

[25] Suvajit Dutta, Bonthala Manideep, Muzamil Basha, Ronnie Caytiles, and N Ch Sriman Narayana Iyenger. Classification of diabetic retinopathy images by using deep learning models. International Journal of Grid and Distributed Computing, 11:89–106, 01 2018. doi: 10.14257/ijgdc.2018.11.1.09.

[26] Mehdi Torabian Esfahani, Mahsa Ghaderi, and Raheleh Kafiyeh. Classification of diabetic and normal fundus images using new deep learning method. Leonardo Electron. J. Pract. Technol, 17:233–248, 2018.

[27] Rishab Gargeya and Theodore Leng. Automated identification of diabetic retinopathy using deep learning. Ophthalmology, 124(7):962–969, 2017. ISSN 0161-6420. doi: https://doi.org/10.1016/j.ophtha.2017.02.008. URL https://www.sciencedirect.com/science/article/pii/S0161642016317742.

[28] Arfan Ghani, Chan H. See, Vaisakh Sudhakaran, Jahanzeb Ahmad, and Raed Abd-Alhameed. Accelerating retinal fundus image classification using artificial neural networks (anns) and reconfigurable hardware (fpga). Electronics, 8(12), 2019. ISSN 2079-9292. doi: 10.3390/electronics8121522. URL https://www.mdpi.com/2079-9292/8/12/1522.

[29] Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C. Nelson, Jessica L. Mega, and Dale R. Webster. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. JAMA, 316(22):2402–2410, 12 2016. ISSN 0098-7484. doi: 10.1001/jama.2016.17216. URL https://doi.org/10.1001/jama.2016.17216.

[30] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, page 1135–1143, Cambridge, MA, USA, 2015. MIT Press.

[31] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2261–2269, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.243. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.243.

[32] Hongyang Jiang, Kang Yang, Mengdi Gao, Dongdong Zhang, He Ma, and Wei Qian. An interpretable ensemble deep learning model for diabetic retinopathy disease classification. In 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 2045–2048, 2019. doi: 10.1109/EMBC.2019.8857160.

[33] Pilsung KANG and Jongmin JO. Benchmarking modern edge devices for ai applications. IEICE Transactions on Information and Systems, E104.D:394–403, 03 2021. doi: 10.1587/transinf.2020EDP7160.

[34] Mahmut Karakaya and Recep Hacisoftaoglu. Comparison of smartphone-based retinal imaging systems for diabetic retinopathy detection using deep learning. BMC Bioinformatics, 21, 07 2020. doi: 10.1186/s12859-020-03587-2.

[35] Jonathan Krause, Varun Gulshan, Ehsan Rahimy, Peter Karth, Kasumi Widner, Greg Corrado, Lily Peng, and Dale Webster. Grader variability and the importance of reference standards for evaluating machine learning models for diabetic retinopathy. Ophthalmology, 125, 10 2017. doi: 10.1016/j.ophtha.2018.01.034.

[36] Ryan Lee, Tien Y. Wong, and Charumathi Sabanayagam. Epidemiology of diabetic retinopathy, diabetic macular edema and related vision loss. Eye and vision (London, England), 2:17, 2015.

[37] Tao Li, Yingqi Gao, Kai Wang, Song Guo, Hanruo Liu, and Hong Kang. Diagnostic assessment of deep learning algorithms for diabetic retinopathy screening. Information Sciences, 501:511 – 522, 2019. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2019.06.011. URL http://www.sciencedirect.com/science/article/pii/S0020025519305377.

[38] Xiaogang Li, Tiantian Pang, Biao Xiong, Weixiang Liu, Ping Liang, and Tianfu Wang. Convolutional neural networks based transfer learning for diabetic retinopathy fundus image classification. In 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pages 1–11, 2017. doi: 10.1109/CISP-BMEI.2017.8301998.

[39] Yi-Peng Liu, Zhanqing Li, Cong Xu, Jing Li, and Ronghua Liang. Referable diabetic retinopathy identification from eye fundus images with weighted path for convolutional neural network. Artificial Intelligence in Medicine, 99:101694, 2019. ISSN 0933-3657. doi: https://doi.org/10.1016/j.artmed.2019.07.002. URL https://www.sciencedirect.com/science/article/pii/S0933365718307747.

[40] Katrine B. Nielsen, Mie L. Lautrup, Jakob K.H. Andersen, Thiusius R. Savarimuthu, and Jakob Grauslund. Deep learning–based algorithms in screening of diabetic retinopathy: A systematic review of diagnostic performance. Ophthalmology Retina, 3(4):294–304, 2019. ISSN 2468-6530. doi: https://doi.org/10.1016/j.oret.2018.10.014. URL https://www.sciencedirect.com/science/article/pii/S2468653018304871.

[41] José Ignacio Orlando, Elena Prokofyeva, Mariana Del Fresno, and Matthew B Blaschko. An ensemble deep learning based approach for red lesion detection in fundus images. Computer methods and programs in biomedicine, 153:115–127, 2018.

[42] Nishtha Panwar, Philemon Huang, Jiaying Lee, Pearse A. Keane, Tjin Swee Chuan, Ashutosh Richhariya, Stephen Teoh, Tock Han Lim, and Rupesh Agrawal. Fundus photography in the 21st century–a review of recent technological advances and their implications for worldwide healthcare. Telemedicine journal and e-health : the official journal of the American Telemedicine Association, 22:198–208, Mar 2016.

[43] Sam Philip, Alan D Fleming, Keith A Goatman, Sofia Fonseca, Paul Mcnamee, Graham S Scotland, Gordon J Prescott, Peter F Sharp, and John Alexander Olson. The efficacy of automated “disease/no disease” grading for diabetic retinopathy in a systematic screening programme. British Journal of Ophthalmology, 91(11):1512–1517, 2007.

[44] Amit Prakash, Debanjan Jana, and Siddheswar Maikap. Taox-based resistive switching memories: prospective and challenges. Nanoscale Research Letters, 8(1):418, 2013. ISSN 1556-276X. doi: 10.1186/1556-276X-8-418. URL https://doi.org/10.1186/1556-276X-8-418.

[45] Gwenolé Quellec, Katia Charrière, Yassine Boudi, Béatrice Cochener, and Mathieu Lamard. Deep image mining for diabetic retinopathy screening. Medical Image Analysis, 39:178–193, 2017. ISSN 1361-8415. doi: 10.1016/j.media.2017.04.012. URL https://www.sciencedirect.com/science/article/pii/S136184151730066X.

[46] Gayathri S., Varun P. Gopi, and P. Palanisamy. A lightweight cnn for diabetic retinopathy classification from fundus images. Biomedical Signal Processing and Control, 62:102115, 2020. ISSN 1746-8094. doi: https://doi.org/10.1016/j.bspc.2020.102115. URL https://www.sciencedirect.com/science/article/pii/S1746809420302676.

[47] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pages 14–26, 2016. doi: 10.1109/ISCA.2016.12.

[48] Kaushal Solanki, Chaithanya Ramachandra, Sandeep Bhat, Malavika Bhaskaranand, Muneeswar Gupta Nittala, and Srinivas R. Sadda. Eyeart: Automated, high-throughput, image analysis for diabetic retinopathy screening. Invest. Ophthalmol. Vis. Sci., 56(7):1429–1429, June 2015. ISSN 1552-5783.

[49] S. D. Solomon and M. F. Goldberg. Etdrs grading of diabetic retinopathy: Still the gold standard?, 2019. ISSN 0030-3747. URL https://www.karger.com/DOI/10.1159/000501372.

[50] Bhavana Sosale, Aravind R. Sosale, Hemanth Murthy, Sabyasachi Sengupta, and Muralidhar Naveenam. Medios- an offline, smartphone-based artificial intelligence algorithm for the diagnosis of diabetic retinopathy. Indian Journal of Ophthalmology, 68(2), 2020. ISSN 0301-4738. URL https://journals.lww.com/ijo/Fulltext/2020/68020/Medios__An_offline,_smartphone_based_artificial.30.aspx.

[51] Jameel Syed Amin and Mohamed Shanavas. Fundus imaging techniques for fundus images. 09 2016. doi: 10.13140/RG.2.2.22836.86407.

[52] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2818–2826, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society. doi: 10.1109/CVPR.2016.308. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.308.

[53] Ankur Taly, Anthony Joseph, Arjun Sood, Arun Narayanaswamy, Dale Webster, David Devoud Coz, Derek Wu, Ehsan Rahimy, Greg Corrado, Jesse Smith, Jonathan Krause, Katy Blumer, Lily Peng, Michael Shumski, Naama Hammel, Rory Abbott Sayres, Scott Barb, and Zahra Rastegar. Using a deep learning algorithm and integrated gradient explanation to assist grading for diabetic retinopathy. Ophthalmology, 2019.

[54] Zhen Ling Teo, Yih-Chung Tham, Marco Yu, Miao Li Chee, Tyler Hyungtaek Rim, Ning Cheung, Mukharram M. Bikbov, Ya Xing Wang, Yating Tang, Yi Lu, Ian Y. Wong, Daniel Shu Wei Ting, Gavin Siew Wei Tan, Jost B. Jonas, Charumathi Sabanayagam, Tien Yin Wong, and Ching-Yu Cheng. Global prevalence of diabetic retinopathy and projection of burden through 2045: Systematic review and meta-analysis. Ophthalmology, 128(11):1580–1591, 2021. ISSN 0161-6420. doi: https://doi.org/10.1016/j.ophtha.2021.04.027. URL https://www.sciencedirect.com/science/article/pii/S0161642021003213.

[55] Parshva Vora and Sudhir Shrestha. Detecting diabetic retinopathy using embedded computer vision. Applied Sciences, 10:7274, 10 2020. doi: 10.3390/app10207274.

[56] Shaohua Wan, Yan Liang, and Yin Zhang. Deep convolutional neural networks for diabetic retinopathy detection by image classification. Computers Electrical Engineering, 72:274–282, 2018. ISSN 0045-7906. doi: https://doi.org/10.1016/j.compeleceng.2018.07.042. URL https://www.sciencedirect.com/science/article/pii/S0045790618302556.

[57] Furqan Zahoor, Tun Zainal Azni Zulkifli, and Farooq Ahmad Khanday. Resistive random access memory (rram): an overview of materials, switching mechanism, performance, multilevel cell (mlc) storage, modeling, and applications. Nanoscale Research Letters, 15(1):90, 2020. ISSN 1556-276X. doi: 10.1186/s11671-020-03299-9. URL https://doi.org/10.1186/s11671-020-03299-9.

[58] Y. Zhou, B. Wang, L. Huang, S. Cui, and L. Shao. A benchmark for studying diabetic retinopathy: Segmentation, grading, and transferability. IEEE Transactions on Medical Imaging, 40(3):818–828, 2021. doi: 10.1109/TMI.2020.3037771.