Development of Deep Convolutional Neural Network Techniques for Edge Detection in Images

Abdullah Mohammad Al-Amaren

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

June 2022

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: Abdullah Mohammad Al-Amaren

Entitled: Development of Deep Convolutional Neural Network Techniques for Edge Detection in Images

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Wen-Fang Xie

_____ External Examiner
Dr. Tokunbo Ogunfunmi

_____ External to Program
Dr. Rajamohan Ganesan

_____ Examiner
Dr. Wei-Ping Zhu

_____ Examiner
Dr. Chunyan Wang

_____ Thesis Co-Supervisor
Dr. M. Omair Ahmad

_____ Thesis Co-Supervisor
Dr. M.N.S. Swamy

Approved by _____
Dr. Jun Cai, Graduate Program Director

August 22, 2022 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Development of Deep Convolutional Neural Network Techniques for Edge Detection in Images

Abdullah Mohammad Al-Amaren, Ph.D.
Concordia University, 2022

Edge detection plays a very crucial role in many image processing, and computer vision applications. Several edge detection methods have been proposed in the literature, which can be categorized into two groups: non learning based methods and learning based methods. The performance of the methods in the first category is not as good as of the methods in the second category. Use of deep convolutional neural networks (DCNNs) has significantly advanced the performance of image edge detection techniques. Most of the existing DCNN techniques are based on the ResNet architecture or on the VGG-16 architecture. The ResNet based techniques exhibit a good edge detection performance at the expense of an extremely high computational complexity due to the use of very large number of convolutional layers. The VGG-16 based techniques have much lower complexity, however, their performance is inferior to that of ResNet based techniques. This inferior performance is due to the fact that the spatial resolution of the feature maps in the last layer is very small. Restoring this small spatial resolution to the original image size would result in a blurred output and a poor localization of the edges. In addition, the deeper layers of the VGG16-based techniques are not able to learn some of the information captured by the initial layers. In this thesis, deep convolutional neural networks based on the VGG-16 architecture, with a focus on a reduced complexity and a performance that is comparable or superior to those of the other existing edge detection techniques, are developed. The thesis has two parts.

In the first part, the idea of residual learning is introduced for the first time in a VGG-16 architecture for the task of edge detection with a view to improve the performance of the existing VGG-16 based networks and also to reduce the complexity. The idea of residual learning enables the network to progressively increase the spatial resolution of the maps

as the features extraction process is moved from the shallow layers to the deeper layers of the network through an appropriate use of transposed convolutions, and the use of smaller number and larger size filters in the deeper layers. The proposed network is experimented on different datasets and is shown to outperform most of the existing techniques. At the same time, the complexity of the proposed network in terms of the number of parameters is lower than that of all the other existing edge detection techniques.

Even though the complexity of the technique proposed in the first part is lower than that of all of the other networks, it is still high and its performance is lower than that of a couple of the other existing techniques, one of which is VGG-16 based and the other is ResNet based. Hence, it is important to develop an edge detection convolutional network having a complexity that is still lower than that of the network proposed in the first part, but without lowering its performance. With this objective in mind, in the second part, we develop deep residual convolutional neural networks based on the VGG-16 architecture. The objective of reduced complexity of the networks is achieved through the use of fire modules which results in increasing the depth of the proposed networks. Also, the use of residual learning allows to maintain or even improve the performance of the networks. The objectives of the proposed networks are validated by conducting experiments employing two different datasets and the proposed networks are shown to outperform all the existing techniques in terms of the edge detection accuracy and complexity.

Acknowledgments

I would like to express my deepest gratitude and appreciation to my supervisors, Prof. M. Omair Ahmad and Prof. M.N.S. Swamy, for their constant support, encouragement, patience, and insightful guidance during my studies and writing of this thesis. I also extend my thanks and appreciation to the committee members for the useful suggestions. I am very grateful to my supervisors and Concordia University for the financial support that I received, which was very crucial for completing this research work.

I sincerely thank all the members of my thesis examining committee for their feedback and support.

I am thankful to all my fellow colleagues at at the Center for Signal Processing and Communications, Concordia University. I would like to thank the administration of the university for providing such a wonderful and well-equipped laboratory for the purpose of study and research.

I would like to thank all of my Jordanian friends in Canada, specially, Dr. Mufleh Al-Shatnawi for his unlimited support.

Special Thanks go to my dearest friend, Dr. Saed Alrabaee, for his help, encouragement, support, and priceless advise he gave to me during my studies.

I would like to thank my best friend Dr. Abdelrahman Idries for being the one I can always rely on.

I would like to express my deep love and appreciation for my lovely son, Mohammad (Hamodeh), whose presence has given me strength and peace of mind.

I am very thankful to my father, Mohammad Alamaren and my mother Basma Alamaren, For their continues support and encouragement to excel in my studies, life and career. Thank you both for giving me strength to reach for the stars and chase my dreams.

Contents

List of Figures

List of Tables

List of Abbreviations

The next list describes several abbreviations that will be later used within the body of the document

Adam adaptive moment estimation

BDCN Bidirectional Cascade Network

BFENet Bio-inspired feature enhancement network

BSDS500 Berkeley Segmentation Dataset and Benchmark

CED Crisp Edge Detection

CNN Convolutional Neural Network

Conv Convolutional Layer

Dexi dense extreme inception subnetwork

DexiNed Dense Extreme Inception Network

ELBP extended local binary pattern

FC Fully Connected

FPS Frame Per Second

HED Holistically nested Edge Detection

HFL High for Low

L_i Loss Function

LRDNN Low-complexity Residual Deep Neural Network

M-LRDNN A modified version of the proposed LRDNN network

N^4-Fields Neural Network Nearest Neighbor Fields

NAO-Multi-scale a Multi-scale decoder Architecture

NYUD NYU Depth

ODS Optimal Dataset Scale

OIS Optimal Image Scale

PiDiNet Pixel Difference Networks

RCF Richer Convolutional Features

RDS Relaxes Deep Supervision

ReLU Rectified Linear Unit

ResNet Residual Neural Network

RHN Residual Holistic Neural Network

S_i Side Output

SGD stochastic gradient descent

SR Squeeze Ratio

UB up-sampling block

List of Symbols

$\mathcal{L}$	Loss Function
Θ	Set of Parameters
y	Input to Network Being Trained
S_{i}	Side Output
L_{i}	Balanced Cross Entropy Loss Function
S	Height and Width of Filter
D	Depth of Filter
†	Network Trained Using Mix of BSDS500 and PASCAL-Context datasets
††	Network Trained Using Mix of BSDS500 and PASCAL-Context Datasets and Tested in Multi-Scale Strategy

Chapter 1

Introduction

1.1 General

Edge detection plays a very important role in several applications, such as fingerprint recognition [1–3], image mosaicing [4, 5], object detection and recognition [6–11], tracking of moving objects [12, 13], motion estimation [14] and medical imaging [15–17]. The edge detection process aims to simplify the analysis of images by reducing the amount of data to be processed, while simultaneously conserving their useful structural information. An example of edge map extracted from the Lena gray image is shown in Figure 1.1.

Several edge detection methods have been discussed in the literature, and they can be divided into two classes: learning-based and non- learning-based methods. The nonlearning-based methods can be further classified into two categories: gradient-based methods such as those in [18], [19], [20], and [21], which, respectively, use the Sobel, Prewitt, Robert's, and Canny operators, and the one proposed in [22]; and Laplacian-based methods that use the Laplacian filter [23] or the Laplacian of Gaussian filter [24]. The gradient-based methods detect the edges by determining the minimum and maximum of the first-order derivatives of an image. The Laplacian-based methods, on the other hand, make use of the zero-crossing of the second-order derivatives of the image to recognize the edges. It should

(a)

(b)

Figure 1.1: (a) Lena gray image, (b) Its edge map.

be noted that in all the non- learning-based methods, the edge detection is based on finding the discontinuities in the image using pixel intensity values.

In recent years, several deep learning edge detection methods have been presented in the literature [25–37]. In these methods, the edge features are automatically learned by using a large number of convolutional layers. They need a large amount of training data as well as the corresponding ground-truth label maps, but the performance of these methods is much

superior to that of the non-learning-based techniques.

1.2 A Brief Review of Learning-Based Methods for Edge Detection

In this section, a review of some of the learning-based methods for edge detection is presented.

The first edge detection technique using deep learning, referred to as the N^4-Fields technique, was proposed in 2014 [25]. The network used in this technique essentially consists of the first three layers of AlexNet network [38], which was proposed for the task of image classification. At the training stage, this technique creates a dictionary of the network outputs obtained by passing a sample of input patches randomly chosen from the images of the training dataset. Then, at the test stage, the output of a patch of the test image is matched with the dictionary items using the nearest neighbor search [39]. The performance of this technique is much superior to that provided by any of the non-learning techniques. However, the N^4-Fields network is a shallow network, since it uses only three convolutional layers of AlexNet. This network is unable to detect all the edges of an image in view of its limited feature extraction capability by using only three layers.

In order to enhance the performance of the N^4-Fields technique, in [26] a network referred to as the DeepEdge network has been constructed for edge detection by employing all the five convolutional layers of the AlexNet network and adding a few additional layers that are fully connected. This network uses the Canny edge detector to extract the candidate edge points. Then, at each candidate point patches with different scales are extracted and passed through the five convolutional layers. The performance of the DeepEdge network is superior to that of the N^4-Fields method, but with an increased computational time.

In [27], the authors have presented a deep CNN scheme in order to learn discriminative

features of natural images for the task of edge map extraction, and have referred to their network as DeepContour network. The DeepContour technique outperforms the DeepEdge technique with a lower computational speed.

The aforementioned techniques [25–27] can be regarded as patch-level prediction techniques, since the features are extracted for each patch of the input image. As the edge detection task is to determine as to whether a pixel is an edge or a non-edge pixel, one could also generate the features for each pixel of the input image. In [28], a pre-trained architecture is used to construct a per-pixel feature learner for contour detection. In this technique, a feature vector for every pixel is extracted by using the first five convolutional layers of AlexNet. This technique outperforms the patch-based techniques of [25–27].

In [29], a technique, referred to as High for Low (HFL) technique, has been developed in which a pre-trained VGG16 [40] classification network is fine-tuned and used to perform edge detection. This technique is similar to that of the DeepEdge technique of [26], in that they both use semantic high level features in order to obtain the final edge map, without using the complicated multi-scale and bifurcated architecture of [26]. The HFL architecture uses the structured edge (SE) detector [41] to extract a set of candidate edge points in order to speed up the computation and outperforms the network of [28].

In [30], the authors proposed a method for edge detection, called the holistically nested edge detection (HED), and implemented it in a deep network based on the architecture of VGG-16. This network has been divided into a number of blocks, each of which consists of convolutional layers and a max pooling layer, except for the last block which contains only fully connected convolutional layers. The outputs, referred to as the side outputs, of the last convolutional layers of all the blocks are then exploited in a nested multiscale feature learning framework for a deep supervision of the network. The performance of the HED network is superior to that provided by the CNNs reported previously [25–29], and has a lower computational complexity.

Based on the idea of deep supervision of [30], a new edge detection technique called the relaxed deep supervision (RDS) has been developed in [31], wherein the authors have used relaxed labels for the supervision of the network, without increasing its complexity over that of [30].

The authors in [32] have proposed a network referred to as DeepBoundaries to improve the performance of HED technique by incorporating the idea of multi-resolution learning by using three parallel stages of the HED network with shared parameters and by carefully designing a loss function for boundary detection training. Each of the three stages operates on the same input image but at a different resolution. The DeepBoundaries technique outperforms both the HED and RDS techniques.

In [33], a technique, called crisp edge detection (CED) network, was proposed for edge detection. This technique has a forward propagation pathway and a backward refining pathway. The former, which uses the VGG-16 architecture as its backbone, generates low-resolution high-dimensional feature maps, and the latter is designed to gradually increase the resolution of the feature maps produced by the former by using an efficient up-sampling method. The feature maps resulting from the backward refining pathway are finally fused to generate an edge map of the input image. The authors of [33] have demonstrated that their CED network has a performance that is superior to that of the networks of [30], [31] and [32]. By replacing VGG-16 in the forward propagation pathway by ResNet-50, the authors have provided another version of their network whose performance is even better than that of CED that uses VGG-16; however, this is achieved at the expense of a much higher computational complexity. In order to distinguish the two versions of the network proposed in [33], we refer to these two CEDs as CED-VGG16 and CED-ResNet50.

In [34], the authors have developed a VGG16-based network, referred to as richer convolutional features (RCF) network. This network employs the idea of deep supervision

by using the outputs of all the convolutional layers for training the network. The performance of this network was shown to be superior to that of CED-VGG16, but inferior to that of CED-ResNet50. The authors of [34] have also provided two other versions of their network using ResNet50 and ResNet101 as backbones. We refer to these three versions of RCF as RCF-VGG16, RCF-ResNet50, and RCF-ResNet101, respectively. It has been shown in [34] that the performance of RCF-ResNet50 is inferior to that of CED-ResNet50. On the other hand,it has been shown that RCF-ResNet101 outperforms all versions of the CED and RCF networks, but with an extraordinarily large computational complexity.

In [36] a new edge detection network, referred to as bidirectional cascade network (BDCN), has been designed based on the architecture of VGG-16. In this network, the features are extracted by the various blocks of the network each at a different scale. Since this network also uses a deep supervision for its training, and the resolutions of the edge maps produced by the various blocks are different, the resolution of the label used for calculating the error at the output of a block is matched with that of the edge map produced by the block. The performance of this network, which we refer to as BDCN-VGG16, was shown to be superior to that of RCF-VGG16, but at the expense of a higher computational complexity. In [36], another version of their network using ResNet50 instead of VGG-16 as backbone has also been developed. We refer to this version of their network as BDCN-ResNet50 whose performance is even better than that of BDCN-VGG16, but at a much higher complexity.

Very recently, a new network referred to as BFENet has been proposed for edge detection [37]. This network consists of three parts: pre-enhancement network, encoding network, and decoding network. The pre-enhancement network simulates the visual information processing and transmission mechanism of the retina/lateral geniculate nucleus in order to enhance the ability of the encoding network to extract details and local features. The encoding network is the VGG-16 network. The decoding network is a new feature

fusion network that fuses the feature maps resulting from the encoding network to generate an edge map of the input image. We refer to this technique as BFENet-VGG16. The performance of this technique is superior to that of all the aforementioned VGG16- based techniques, however with a large computational complexity, except for BDCN-VGG16.

There are some other recently proposed edge detection techniques [35, 42, 43] that are neither VGG-16 based nor ResNet based architectures. In [35], the authors have presented a novel architecture, referred to as dense extreme inception (DexiNed) network, to generate thin edge-maps. This network has two parts, a dense extreme inception (Dexi) subnetwork and an up-sampling blocks (UB) subnetwork. The output from each of the Dexi main blocks is fed to an up-sampling block to generate an intermediate edge map. All of the intermediate edge maps are fused to generate the final edge map. In [42], a multi-scale decoder architecture, referred to as NAO-Multi-scale network, is presented. This architecture uses mathematical morphology to perform morphological operations using standard convolution layers. These two heavy networks provide good edge detection performance. However, their complexity is very high, in that the complexity of [42] is higher than those of the networks designed based on VGG-16 or ResNet architectures and that of [35] is higher than that of all the VGG-16 based techniques. In [43], the authors have developed a lightweight convolutional neural network, referred to as pixel difference network (PiDiNet), to achieve a trade-off between accuracy and efficiency for the task of edge detection. This network uses an ELBP edge detector to capture the gradient information of the input image and uses this information to generate the edge map of the image through the powerful learning ability of the deep convolutional layers of the network. However, the performance of the network is limited in view of the lightweight nature of the network.

1.3 Motivation and Objectives

It is seen from Section 1.2 that the deep residual learning methods, such as CED-ResNet50 [33], RCF-ResNet50 [34], RCF-ResNet101 [34], and BDCN-ResNet50 [36], provide very good edge detection performance. The high performance of these networks results from their capability of extracting a richer set of features resulting from their residual learning and use of a large number of convolutional layers. However, the complexity of these networks is extremely high. For example, the number of parameters employed by BDCN-ResNet50 is 28.7 M. there are some other edge detection networks, such as HED-VGG16 [30], CED-VGG16 [33] and RCF-VGG16 [34], which have been designed based on the VGG-16 architecture, and have relatively a much lower complexity. For example, the number of parameters employed by HED-VGG16 [30] and RCF-VGG16 [34] networks is around 14.7 M, and the number of parameters employed by the latest technique, namely, BDCN-VGG16 [36], is 16.3 M. Although these networks greatly benefit from deep supervision, their performance is not as good as that provided by the networks developed based on the ResNet architecture.

The objective of this thesis is to develop edge detection convolutional networks that have a complexity comparable or lower than that of the VGG-16 based convolutional neural networks and a performance that is comparable or superior to that of the ResNet based convolutional neural networks. This thesis attempts to achieve the aforementioned objective by developing VGG-16 architecture based edge detection neural networks, which have the capabilities of deep supervision, residual learning, and rich feature generation while using small number of parameters.

This thesis has two parts. In the first part, an edge detection convolutional neural network, based on the VGG-16 architecture, is developed by introducing in it deep supervision and residual learning in such a way that the overall complexity of the network proposed is lower than that of all the VGG-16 based networks and its performance is very close to that of the

ResNet based networks. In the second part, deep neural networks, also based on the VGG-16 architecture, are developed for the task of edge detection whose complexity are lower than that of the edge detection network of part one and the performance superior to that of all the existing edge detection networks. As in the network of part one, these networks also use deep supervision and residual learning in their architectures; however, the significant improvements in performance and complexity of these networks are achieved by introducing fire modules in their architectures.

1.4 Organization of the Thesis

The thesis is organized as follows. Chapter 2 provides the fundamental concepts of deep learning and convolutional neural networks. In addition, we present a brief overview on the VGG-16 and ResNet architectures. Chapter 3 starts with the presentation of the design and implementation of the first edge detection neural network proposed in this thesis. Experimental results are then provided to demonstrate the performance and complexity of the network proposed. A comparison of the performance and complexity of the proposed network with that of the other edge detection networks is provided. In Chapter 4, the design and implementation of the baseline edge detection neural network is first presented. The results obtained by performing experiments on the baseline network are then provided and compared with the other networks commonly used for edge detection. Two modified versions of the baseline network are also presented and discussed. Finally, in Chapter 5, some concluding remarks on the work undertaken in this thesis are made and some suggestions given for further research based on the findings of the thesis.

Chapter 2

Background Material

In this chapter, a brief review of convolutional neural networks, as a background material useful to understand the work carried out in this thesis, is given. First, the motivation behind the development of convolutional neural networks is discussed. The architecture of a typical convolutional neural network and its various components are next presented. Then, the two schemes commonly used for the training of a convolutional neural network are described. Finally, two specific architectures, namely, VGG-16 and ResNet-34, are described, since the architectures of the edge detection neural networks developed in this thesis are based on the architecture of the former by introducing in them the concept of residual learning originally proposed in the architecture of the latter to address the gradient vanishing problem of deep convolutional networks.

2.1 Convolutional Neural Networks

Convolutional neural network (CNN), also called, ConvNet, was first introduced in 1998 by LeCun, et.al [44] for recognizing hand written digits by proposing a CNN architecture called LeNet. Through this architecture the authors introduced the concept of end-to-end

learning, in which the features of the network inputs are automatically learned by using convolutional layers. A convolutional layer is a layer of filters carrying out convolutional operations on the same input by using different filters each of which is characterized by a different set of learnable coefficients or kernel values. A convolutional neural network provides a data driven platform for extracting useful features of the input signal automatically. Therefore, it is capable of providing a performance superior to that of a scheme that uses hand-crafted features. However, the capability of a CNN for extracting useful features is very much dependent on the number of coefficients employed by it and the availability of large training dataset. Both these requirements became bottlenecks for using CNNs in various applications. In the following period, as more powerful hardware resources and large training datasets became available, CNN became a very powerful tool for various applications. As a result, since the beginning of 2010, different CNN architectures have been developed as they differ in the capabilities of extracting good features and in their stabilities for different applications. AlexNet is the first large CNN (with 62 million parameters) that was proposed in 2012 for the task of image classification [38]. In subsequent years, many other powerful CNN architectures, each having its own strength and limitation, have been proposed [40,45–53] for a wide variety of applications such as computer vision, healthcare, entertainment, and machine translation.

2.2 Architecture of a typical CNN

The architecture of a typical CNN has three types of layers [54]: convolutional layers, pooling layers, and fully connected layers, as shown in Figure 2.1. A convolutional layer uses several filters of kernel size SxSxD, where S represents the height or width of the filter and D the filter depth, which is equal to the number of channels in the volumetric data input to the layer. The weights (coefficients) of the filter are generally randomly initialized. A

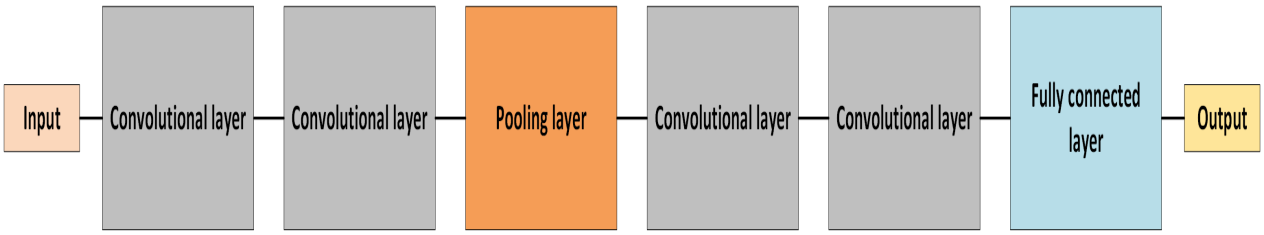

Figure 2.1: A typical architecture of a convolutional neural network.

filter slides over the feature tensor input to the layer and at each position of the window, a convolutional operation is performed followed by an activation operation, to produce a single pixel of a map of the output of the convolutional layer. Different filters are used in a layer to produce different maps of its output.

A convolutional layer may be followed by a pooling layer, the purpose of which is to reduce the spatial resolution of the feature maps produced by a convolutional layer so as to allow the succeeding convolutional layers to employ a larger number of filters in order to extract additional robust features without adversely affecting the complexity of the network. There are two types of pooling layers, average pooling layer and max pooling layer, commonly used in convolutional neural networks. For a pooling operation, a window of a given size is slid over a feature map and at each position of the window, the pixels within the window are replaced by a single pixel, where the value of the single pixel is the average of the pixel values within the window in the case of an average pooling layer, whereas it is the maximum value of the pixels within the window in the case of a max pooling layer. Figures 2.3 and 2.2 illustrate examples of the two types of pooling operations.

The focus in the design of a convolutional neural network is to develop an architecture that is able to extract a set of feature maps, which are very rich from the perspective of the intended application of the network. Once such a rich set of feature maps is extracted, the network needs to have some additional layers, which can be used to finally produce the

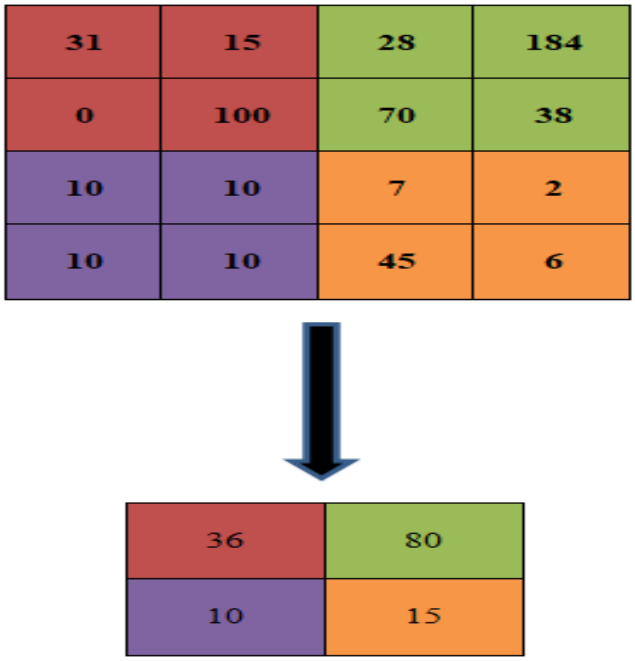

Figure 2.2: Example of the average pooling operation

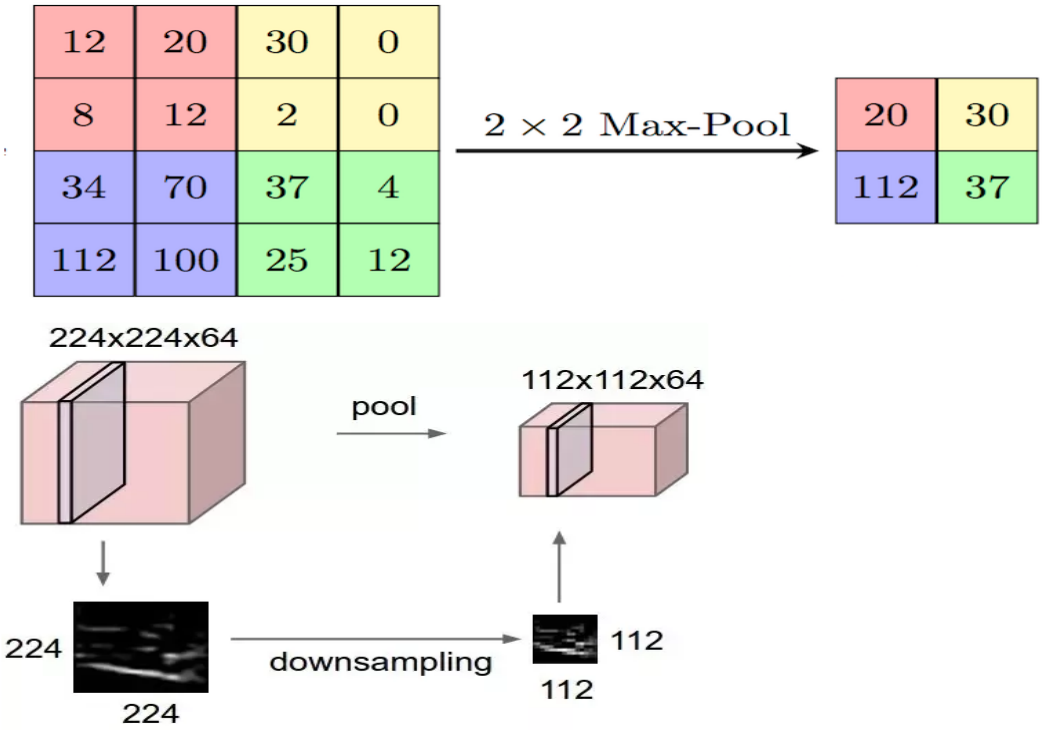

Figure 2.3: Example of the max pooling operation

network's desired objective, such as to denoise, detect edges or classify input images using these maps. For example for denoising and edge detection of gray level images, these final layers could be 1x1 convolutional layers. On the other hand, for the task of image classifications, these final layers are fully connected. Some other examples of applications in which the network require fully connected layers as additional layers are speech recognition [55] and image clustering [56]. The purpose of fully connected layers is to determine the label of the input data from its maps extracted by the convolutional layers of the network. In a fully connected layer, at a time, a 1D convolution is performed with all the pixels of the entire set of the maps produced by the previous layer. Figure 2.4 illustrates an example of how fully connected layers are used for the task of image classification wherein two fully connected layers are employed. It is seen from this figure that a set of M feature maps each of size NxN corresponding to an input image that needs to be classified are first flattened into a single MNx1 size feature vector and then convolved using successively K 1D filters, each of kernel size MNx1, of the first fully connected layer (FC1). The output of FC1 is further processed by the filters in the second fully connected layer (FC2). Since in this example the second fully connected layer is the last layer of the network, the number of filters C used in FC2 needs to be the same as the number of possible classes. The entries of the final vector produced by FC2 are the probabilities of the input image belonging to the various classes.

2.3 Training of a typical CNN

As seen in Section 2.2, convolutional neural network consists of several convolutional layers that employ a large number of filters. In order to make a network operational, that is, for it to extract features of a given input and use them to achieve the desired objective, the values of the parameters of these filters must be known. The process of determining

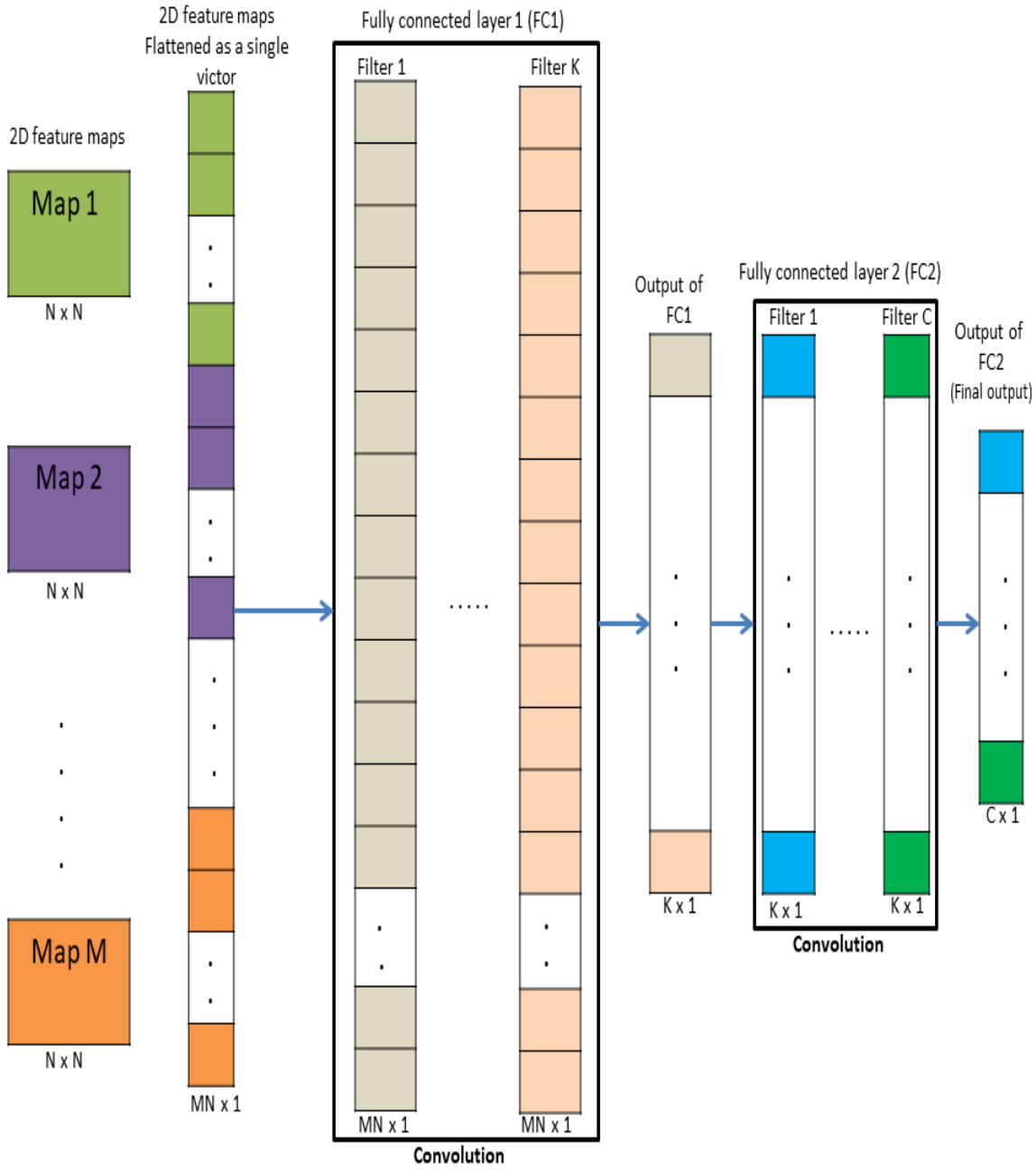

Figure 2.4: An example of how fully connected layers are used for the task of image classification.

suitable values of these parameters is referred to as training the network. The network can be trained through a supervised learning or unsupervised learning.

In supervised learning assuming that a training set containing a sufficient number of labeled data is available, the values of the network parameters are determined by making the network to map the data in the training set to a set of labels that are as close to the actual labels of the data in the training set as possible.

In a supervised learning the training process starts by feeding in a single input data from the training set to the network with its parameter values initialized randomly. The network computes an output corresponding to this input data. Then, a loss (prediction error) between this predicted output and the given label corresponding to the input data is obtained. The idea of training the network is to adjust the parameter values of the network using this prediction error so that in the next iteration, in which a new training data is fed into the network with the adjusted parameters, the prediction error would be lower than the current prediction error. Backpropagation is a mechanism used to update the parameter values for the next iteration using the current prediction error. The gradient descent optimization algorithm and its variants, such as stochastic gradient descent (SGD) or adaptive moment estimation (Adam), are the optimization algorithms that are commonly used to iteratively adjust the values of the network parameter in order to minimize the loss function. In the gradient descent algorithm, the value of the gradient of the loss function evaluated using the values of the parameters and prediction error at the current iteration is used to determine the adjustment in the parameter values for the next iteration. A forward pass and a backward pass together using a single input data is counted as one pass. One cycle of all the passes using the entire dataset is called one epoch. Generally, a network requires several epochs using the same training set in each epoch to train it satisfactorily. The performance of the

trained model is finally tested by examining the accuracy of its output when it is fed with the input from the test dataset comprising the data not seen by it during the training. In summary, in supervised learning, a network is trained by minimizing a loss function given by

$$E = \mathcal{L}(f_\Theta(y), x) \tag{2.1}$$

where y is the input to the network being trained and x is the given label corresponding to the input y, $f_\Theta(y)$ represents the network operation employing a set of parameters Θ, by using a gradient descent optimization algorithm, and thus, the final set of the parameter values of the trained network is obtained.

In unsupervised learning, the network needs to be trained, that is, its parameter values are to be determined, without the knowledge of annotations or labels of the data in the training set. In this case, the input data y from the training set itself is used for the label x in the loss function given by (1). Also, depending on the application of the network, the training data itself may be modified and used for y in (1). Minimization of the loss function thus modified is carried out by applying an optimization algorithm in the same way as done in the case of supervised learning.

2.4 VGG-16 Network

The VGG-16 convolutional neural network was developed by Karen Simonyan and Andrew Zisserman for the task of image classification [40]. The architecture of VGG16 network consists of six blocks, as shown in Figure 2.5 Each of the first two blocks contains two convolutional layers and one pooling layer, each of the next three blocks contains three convolutional layers and one pooling layer, and finally, the last block contains three fully

connected layers. Therefore, the network, in total, has 13 convolutional layers, 5 pooling layers, and 3 fully connected layers. Each of the convolutional operations in this network is performed using filters of kernel size 3x3 and followed by an activation operation using the RelU function. Even though the VGG16 architecture was developed for the task of image classification, its modified versions have been successfully used in many other applications, such as edge detection, object recognition [57], and medical image segmentation [58], in view of the richness of the features extracted by the architecture of VGG16.

2.5 ResNet-34 Network

ResNet-34, which was developed originally also for the task of image classification, is a very deep convolutional neural network where skip connections between the layers are used to avoid the vanishing gradients problem of deep networks [47]. As a result, this network is able to use a very large number of convolutional layers to produce a very rich set of hierarchal features. The architecture of ResNet34 network has 33 convolutional layers, 2 pooling layers, and 1 fully connected layer, as shown in Figure 2.6. Each of the convolutional operations in this network is performed using filters of kernel size 3x3 followed by an activation operation using the RelU function, except for the first convolutional layer in which a kernel size 7x7 is used in its filters. There are three other ResNet architectures, ResNet-50, ResNet-101, and ResNet-152, with the only difference in each from the original one being in the number of convolutional layers in the architecture. Specifically, the numbers of layers used by these three architectures are 50, 101, and 152, respectively. There are several modified versions of ResNet architectures that have been developed for applications other than classification, such as the architectures of [33], [59], and [60] which have been developed for the tasks of edge detection, image compression, and speech recognition, respectively.

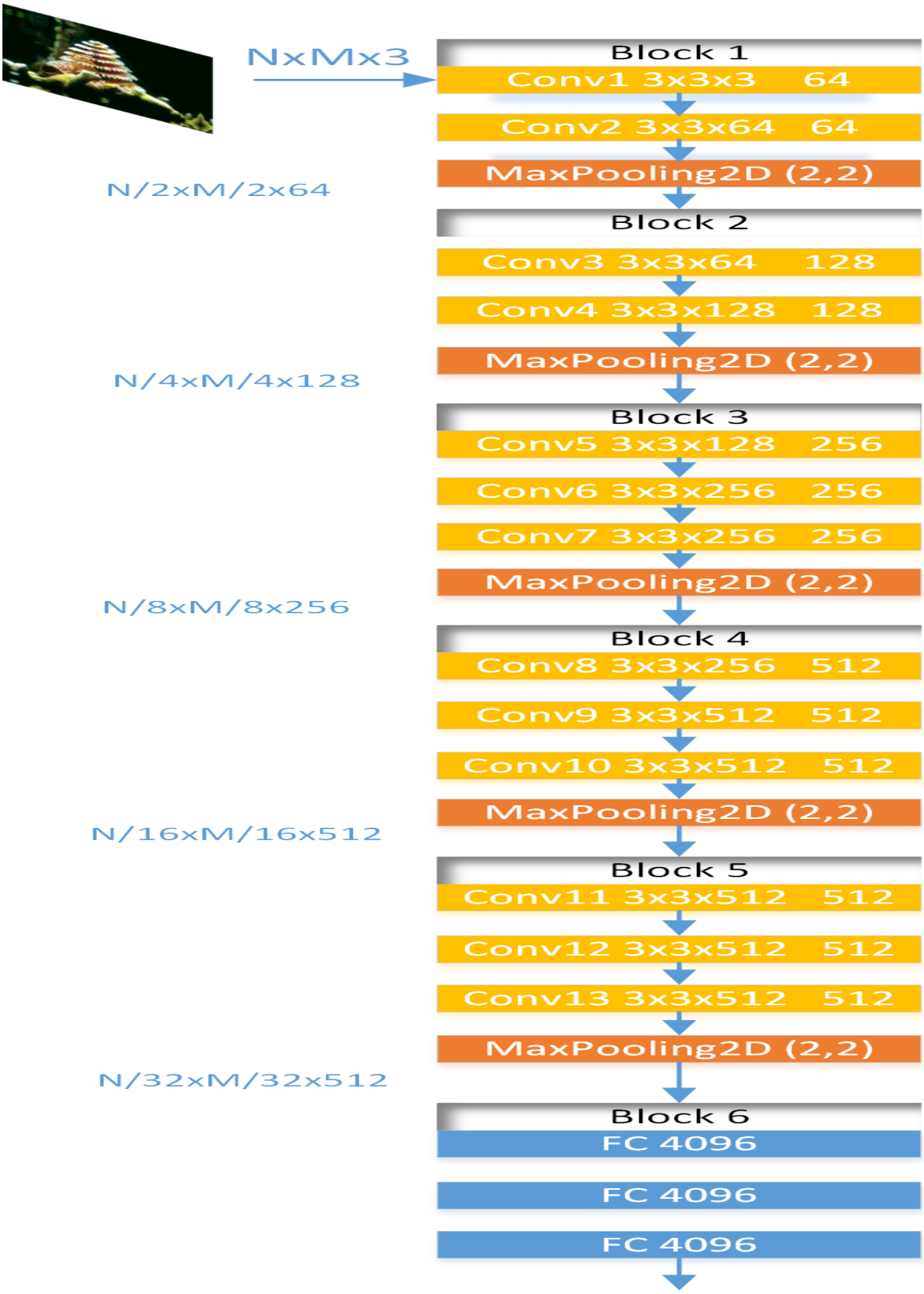

Figure 2.5: The VGG-16 network.

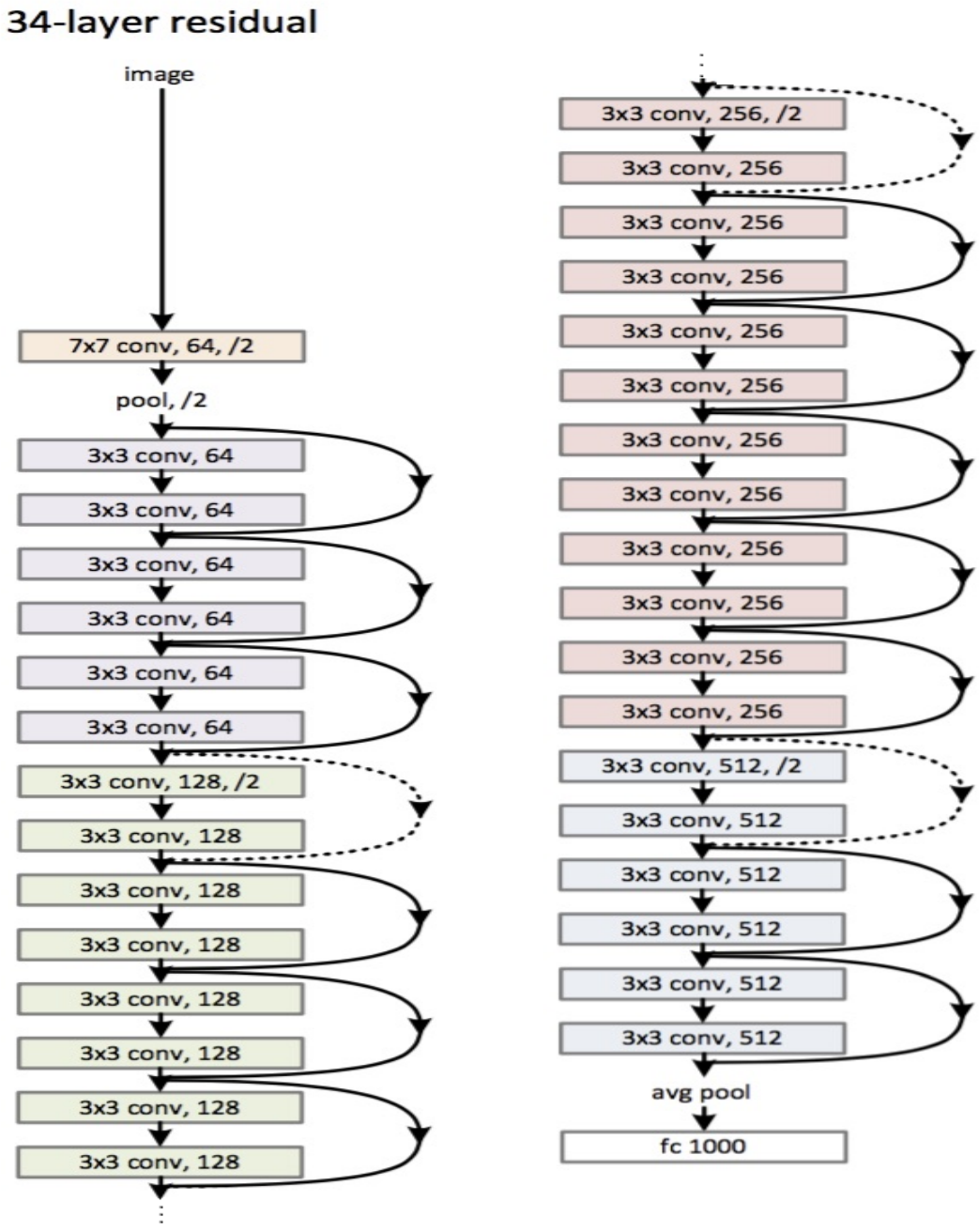

Figure 2.6: ResNet-34 Architecture.

2.6 Summary

This chapter has presented the background material that is relevant and necessary to understand the research work carried out in this thesis. First, a brief introduction to convolutional neural networks has been given, and the architecture and components of a typical convolutional neural network have been explained. Then, the schemes for training convolutional neural networks are described. Finally, the architectures of the two specific networks, VGG-16 and ResNet-34, have been briefly described.

Chapter 3

A Residual Neural Network for Edge Detection

3.1 Introduction

Existing deep residual learning based techniques for edge detection provide good performance but at the expense of an extremely high computational complexity. On the other hand, the complexity of the VGG16-based edge detection techniques is relatively much lower than that of the residual learning based techniques but their performance is low. The reasons for the low performance of the VGG-16 based networks are as follows. (i) The deeper layers of the VGG-16 architecture are not able to learn some of the information captured by the initial layers. (ii) The spatial resolution of N x M of the feature maps in the convolutional layers of the first block finally decreases to N/32 x M/32 in the convolutional layers of the fifth block. Restoring these feature maps to the original input size is necessary in some of the applications such as image edge detection. However, a sudden restoration of the spatial resolution of N/32 x M/32 of the maps in the fifth block to the original N x M resolution would result in a blurred output. (iii) Increasing the number of layers in the VGG-16 architecture in order to improve the performance, even in applications where a

somewhat larger complexity is acceptable, would potentially risk the vanishing gradients problem. In this chapter, we develop a new VGG-16 based edge detection architecture [61] that is capable of addressing these limitations of the existing VGG-16 based edge detection networks by introducing in it a mechanism of residual learning. In Section 3.2, the proposed architecture is presented. In Section 3.3, the performance and complexity of the proposed edge detection network are provided and compared with those of the state-of-the-art edge detection networks. Finally, the conclusion of this study is provided in Section 3.4.

3.2 The Proposed Method

The proposed edge detection network is built upon the VGG-16 architecture. In order to better understand the various concepts used in building the proposed edge detection network and to contrast it with VGG-16 that was originally designed for image classification, the architecture of the latter is dedicated in Figure 2.5 and that of proposed network in Figure 3.1.

The proposed network, referred to as residual holistic network (RHN), for edge detection consists of elven blocks, blocks 1 – 11, of which only the first one is exactly the same as that used in VGG16. The second block contains two convolutional layers, one max pooling layer, and one transposed convolutional layer. The third and fifth blocks have three convolutional layers, and in addition, the former has a max pooling layer and the later a feature map pooling layer. The fourth block is similar to the second one with a feature map pooling layer and an additional convolutional layer. Block 6 and block 11 have a convolutional layer and a loss function, whereas each of blocks 7-10 has in addition a transposed convolutional layer.

Each of the convolutional layers in the first five blocks has filters of size SxSxD, where S represents the height and width of the filter and D is the filter depth that is equal to the

number of channels in the volumetric data input to the layer. For example, in the first convolutional layer (Conv1) the value of D is 3, since image input to the network is a color image and thus has 3 channels.

The number of filters in each of the two layers of block 1 is chosen to be 64 for achieving a good balance between the complexity and variety of features to be extracted by the network. A max pooling layer follows these two convolutional layers, which produces a lower resolution version of the feature maps input to this layer by selecting the most salient feature in each of the 2x2 windows of each of the channels of the input. The use of the max pooling layer reduces the amount of data that is input to block 2. This reduction allows the use of a larger number of filters in the layers of block 2 in order to extract more robust features without adversely affecting the complexity. We use 128 filters in each of the three convolutional layers of block 2. Since residual learning schemes have been successfully used to extract richer set of features, we now apply this idea first to block 2. In order to accomplish this, a transposed convolutional layer is used after the max pooling layer of this block to up-sample the feature maps resulting from the max pooling layer and to decrease their number to match the spatial resolution and the depth of the data from block 1. As a result of applying the residual learning to block 2, the output of this block is the sum of the outputs from its transposed convolutional layer and that of block 1. The role of block 3, whose architecture is similar to that of block 1 with one additional convolutional layer and the use of 128 filters rather than 64 in each of the convolutional layers, is simply to improve the network performance.

The size of the filters used in all of the convolutional layers in the first 3 blocks is 3x3, since a small size filter is more appropriate to extract low-level features. However, the use of a larger size filter enables a denser connection between the feature maps and thus can help in learning the global features for the task of edge detection. For this reason, the size of the filters used in blocks 4 and 5 are increased, respectively, to 5x5 and 7x7. To deal with the

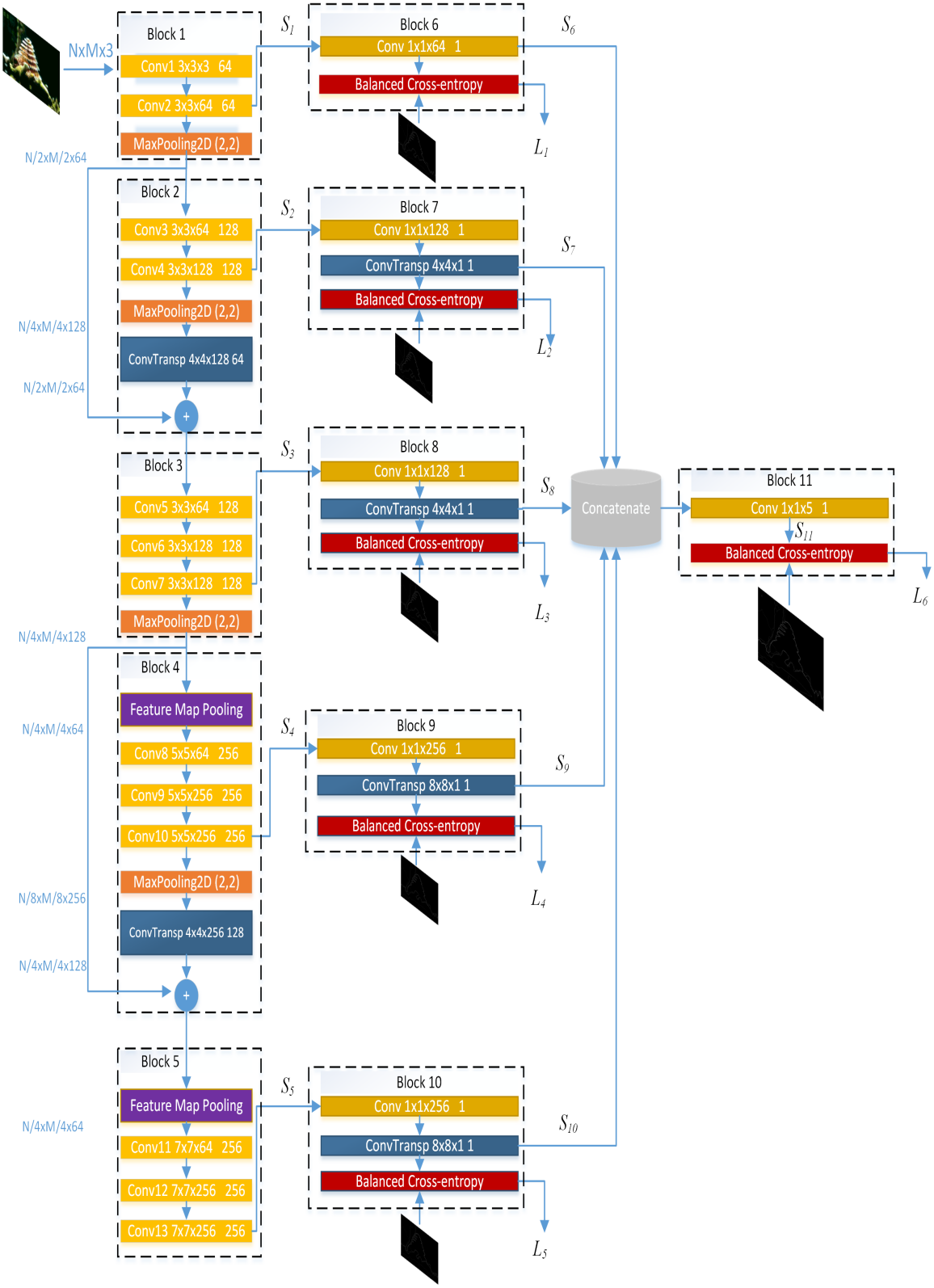

Figure 3.1: The proposed edge detection architecture.

problem of higher number of arithmetic operations associated with the increased number of parameters by the use of larger size and number of filters in the convolutional layers of block 4, a smaller amount of data is used for processing by this block. This is achieved through the reduction of the spatial resolution by the max pooling layer of block 3 and the number of channel reduction by a feature map pooling layer at the beginning of block 4 by using only 64 filters. The residual learning is also applied to this block, which then requires the use of a transposed convolutional layer as well. In view of using even a larger size of filters in block 5, we again use a feature map pooling layer with 64 filters at the beginning of the block in order to keep the number of operations low by making the size of the data to be processed by its three convolutional layers to be smaller.

The proposed network is further developed in order for it to benefit from deep supervision learning by augmenting it with blocks 6 – 11. These six blocks generate six different loss functions, L_1 – L_6. The loss functions L_1 – L_5 are generated using the side feature maps S_1 – S_5, respectively. Figure 3.2 shows some examples of the network side feature maps after up-sampling (side outputs).

It is to be noted that these five side feature maps have, 64, 128, 128, 256, and 256 feature maps with spatial resolutions of NxM, N/2xM/2, N/2xM/2, N/4xM/4 and N/4xM/4, respectively. While the existing algorithms such as HED and RCF, have spatial resolutions of NxM, N/2xM/2, N/4xM/4, N/8xM/8 and N/16xM/16. In order to construct the loss functions L_1 – L_5, each of these sets of feature maps has to be converted into a single map of spatial resolution NxM. This is accomplished by the convolutional and transpose convolutional layers of the blocks 6-10 producing, respectively, the feature maps S_6 – S_{10}. For example, the input to block 9 has 256 feature maps of size N/4xM/4 resulting from block 4. Therefore, block 9 first by using one convolutional filter of kernel size 1x1x256 is able to produce a single feature map of size N/4xM/4, and then by using a transposed

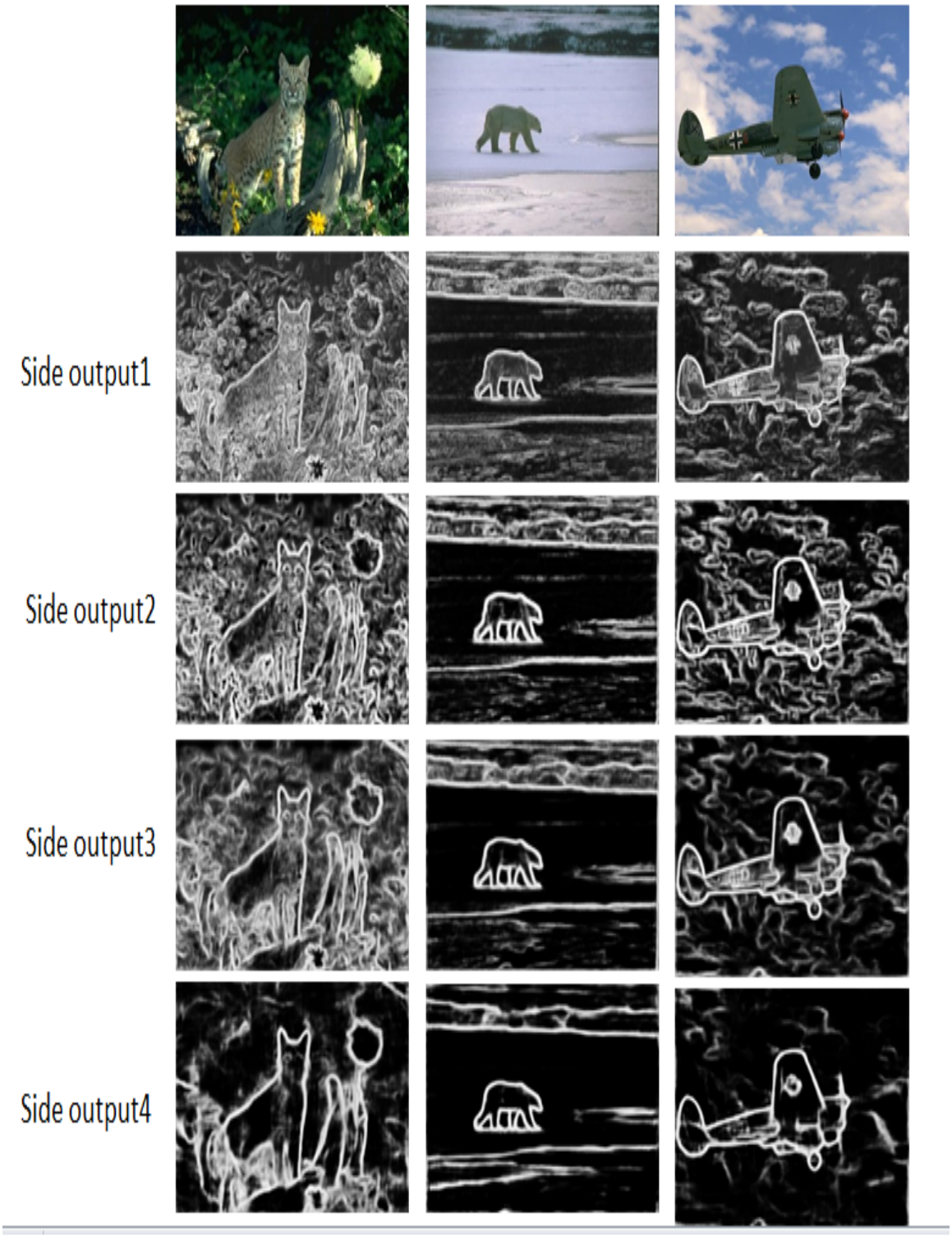

Figure 3.2: Examples of the network side outputs

convolutional layer of size 8x8x1 up-samples this feature map to the spatial resolution of NxM. Note that block 1 does not need a transposed convolutional layer, since its input maps already have the desired spatial resolution of NxM resulting from block 1. The feature maps S_6, ..., S_{10} are concatenated and then the resulting feature maps S_{11} are combined into a single map, S_{12}, by using a convolutional filter in block 13. Six balanced cross entropy loss functions denoted by L_i, i = 1, ..., 6, are produced using the edge maps S_1, ..., S_5 and S_{11}, respectively, and the ground truth, [62]

$$\begin{aligned} L_i(W, w^{(i)}) = -\beta \sum_{j \in Y_+} logPr(y_j = 1 \mid X; W, w^{(i)}) \\ -(1-\beta) \sum_{j \in Y_-} logPr(y_j = 0 \mid X; W, w^{(i)}) \end{aligned} \tag{3.1}$$

where W denotes the set of all the network parameters, $w^{(i)}$ denotes the classifier weights corresponding to the side output S_i, i = 1, ..., 5, 11, $X = (x_j, j = 1, ..., \mid X \mid)$ and $Y = (y_j, j = 1, ..., \mid Y \mid)$, respectively, denote a training image and the corresponding ground-truth image, Y_+ and Y_-, respectively, denote the set of indices in the edge and non-edge ground-truth labels, and $\beta = \mid Y_- \mid / \mid Y_+ + Y_- \mid$ represents the fraction of the non-edge pixels in Y. β and $1 - \beta$ in (1) are used to overcome the problem of inequitable distribution of the edge and non-edge pixels in the ground-truth. $Pr(y_j = 1 \mid X; W, w^{(i)})$ is computed using the ReLU function on the activation value at pixel j. The six loss functions, L_1, ..., L_6, are finally combined to obtain a single cost function [62], given by

$$\mathcal{L} = \sum_{i=1}^{6} L_i(W, w^{(i)}) \tag{3.2}$$

This single cost function is used for a deep supervised training of the network.

3.3 Experimental Results and Comparisons

In this section, experiments on the proposed RHN are performed using three datasets, namely, the Berkeley Segmentation Dataset and Benchmark (BSDS500) [63], NYU Depth (NYUD) [64], and Multicue [65] datasets. The first two datasets provide ground-truth images which may not necessarily be considered as an edge map or a boundary map of the images in the dataset, whereas the third dataset has two ground-truth images for each image in the dataset, the first one being an edge map and the second one a boundary map of the image. The performance of the proposed network in terms of ODS and OIS , and complexity in terms of the average number of frames produced by the network per second (FPS) and the number of network parameters are studied and compared with that of the other existing state-of-the-art networks for edge detection. Localization tolerance is a parameter used to compute the ODS and OIS metrics for evaluating the performance of an edge detection method [66,67]. This parameter is related to the maximum distance (in terms of the number of pixels) between the corresponding pixels of the predicted and ground truth edge maps, and controls the crispness of the predicted edge map (i.e. the values of the performance metrics). The value of this parameter is empirically determined in order to provide the best edge detection performance on the images of a given dataset [66, 67].

For the training of the proposed network, the mini-batch-size is set to 10. The adaptive moment estimation [68] is used, instead of stochastic gradient descent [69] because it is computationally efficient and needs less memory requirements. The weight decay is set to $2\text{x}10^{-4}$ and the learning rate to 10^{-6}. The size of the training images is set to 320 x 480, 480 x 480, and 500 x 500 for the BSDS500, NYUD and Multicue datasets, respectively. Furthermore, data augmentation is used to increase the sample size in each dataset. Each convolutional layer is followed by a ReLU activation unit. Batch normalization is used after each ReLU function in order to improve the network performance and stability by normalizing the distribution of the features produced by the activation operation of the

previous layer [70–72]. The training is terminated after 35 epochs. The proposed algorithm is implemented using Keras [73] that is back ended by the TensorFlow package [74]. The training procedure for the proposed networks is conducted on a machine with Intel(R) Core(TM) i7-7700K CPU @4.2 GHz, 32 GB RAM and NVIDIA GeForce GTX 1080 GPU.

3.3.1 Quantitative Performance and Comparison

The performance of the proposed deep learning technique in terms ODS and OIS F-measures as well as the complexity of the network in terms of FPS and number of network parameters are obtained using the different datasets. All the results are compared to that of the state-of-the-art- networks. As in other deep learning methods, the standard non-maximal suppression (NMS) technique [75] is used for thinning the detected edges for the evaluation of the different edge maps resulting from the various methods.

Performance Using the BSDS500 Dataset

The BSDS500 dataset consists of 200 training, 100 validation, and 200 test images. Each image in this dataset is labeled by 4 to 9 annotators to generate the ground truth edge maps, where if a pixel is assigned a positive label by at least three annotators then it is considered as belonging to an edge. Similar to the previous works, we mix the augmented data of BSDS500 with the Pascal-Context dataset [76] as training data. The PASCAL Context dataset has 10K labeled images. For the network training, we augment the images of the BSDS500 dataset and their ground-truths, by applying only the operations of scaling by 0.5, 1 and 1.5, and rotation by 0°, 22.5°, 45°, ..., 315° and 337.5°, and flipping. For the training, Pascal-Context dataset is also augmented but only through the operation of flipping its images.

Table 3.1: Comparison of the proposed network with state-of-art networks on the BSDS500 dataset. The networks trained using a mix of the BSDS500 and PASCAL Context datasets and tested only on the images of the BSDS500 dataset

Method	ODS F-measure	OIS F-measure	No. of parameters in Millions	FPS
Human	0.80	0.80	-	-
Canny [21]	0.611	0.695	-	28
Pb [77]	0.672	0.695	-	-
SE [41]	0.743	0.763	-	2.5
OEF [79]	0.746	0.770	-	2/3
DeepContour [27]	0.757	0.776	-	1/30
DeepEdge [26]	0.753	0.772	-	1/1000
HFL [29]	0.767	0.788	-	5/6
N^4-Fields [25]	0.753	0.769	-	1/6
DexiNed-f [35]	0.729	0.745	>>18	-
CSCNN [80]	0.756	0.775	-	-
CED-ResNet50 [33]	0.810	0.829	-	-
RCF-ResNet101 [34]	0.812	0.829	>>44	12
BDCN-ResNet50 [33]	**0.826**	**0.840**	28.7	-
HED [30]	0.788	0.804	14.7	30
RDS [31]	0.792	0.810	14.7	30
CED-VGG16 [33]	0.803	0.820	-	-
RCF-VGG16 [34]	0.806	0.823	14.8	30
BDCN-VGG16 [34]	0.820	0.838	16.3	-
RHN	0.817	0.833	**11.5**	**33**

Results in terms of ODS F-measure, OIS F-measure and FPS of the proposed technique for the BSDS500 dataset are given in Table 3.1. For the purpose of comparison, the results from different classical edge detection techniques, Canny [21], Pb [77], SE [78], and OEF [79], and also those obtained of the recent deep learning techniques, namely, DeepContour [27], DeepEdge [26], HFL [29], N^4-Fields [25], DexiNed-f [35], CSCNN [80], HED [30], RDS [31], CED-VGG16 [33], CED-ResNet50 [33], RCF-VGG16 [34], RCF-ResNet101 [34], BDCN-VGG16 [36], and BDCN-ResNet50 [36] are listed in this table.

It is seen from this table that the performance of the proposed RHN in terms of ODS F-measure and OIS F-measure is superior to that of all the other VGG-16 based techniques except for BDCN-VGG16 and superior or comparable to that of the ResNet based techniques. It is seen from this table that the values of the ODS and OIS F-measures provided

by the proposed RHN are, respectively, 0.011 and 0.010 higher than that of RCF-VGG16 technique. This improved performance of the proposed technique is achieved with 11.5 M parameters and a rate of 33 frames per second, as compared to the 14.8 M parameters and a rate of 30 frames per second provided by RCF-VGG16 technique. It is also to be noted that even though the performance of the proposed technique is inferior to that of the BDCN-VGG16 and BDCN-ResNet50 techniques, the number of parameters employed by these techniques are 16.3 M and 28.7 M, which are significantly higher compared to 11.5 M parameters utilized by the proposed RHN. Figure 3.3 presents the precision-recall curves of the proposed RHN and some of the other methods used in our comparison on the BSDS500 dataset.

Performance Using the NYUD Dataset

The NYUD dataset has 1449 pairs of input images recorded by Microsoft Kinect cameras from indoor scenes. The first image in each pair of this dataset is captured by an RGB camera, whereas the second one by a depth camera. Figure 3.4 shows two such pairs of images from this data set. In our experiments, the NYUD dataset is spilt into 381 training, 414 validation and 654 test pairs of images. For the network training, we augment the images of the NYUD dataset and their ground-truths, by applying only the operations of scaling by 0.5, 1 and 1.5, and rotation by 0°, 90°, 180°, and 270°, and flipping. For the purpose of evaluation, the maximum tolerance allowed for correct matches of edge predictions is 0.011 for the images of this dataset instead of 0.0075 that was used for BSDS500 dataset, since the size of the images in the former is larger. The network is trained separately for the RGB and depth training sets resulting in two trained models. The final predicted edge map is obtained by averaging the outputs from the RGB and depth models.

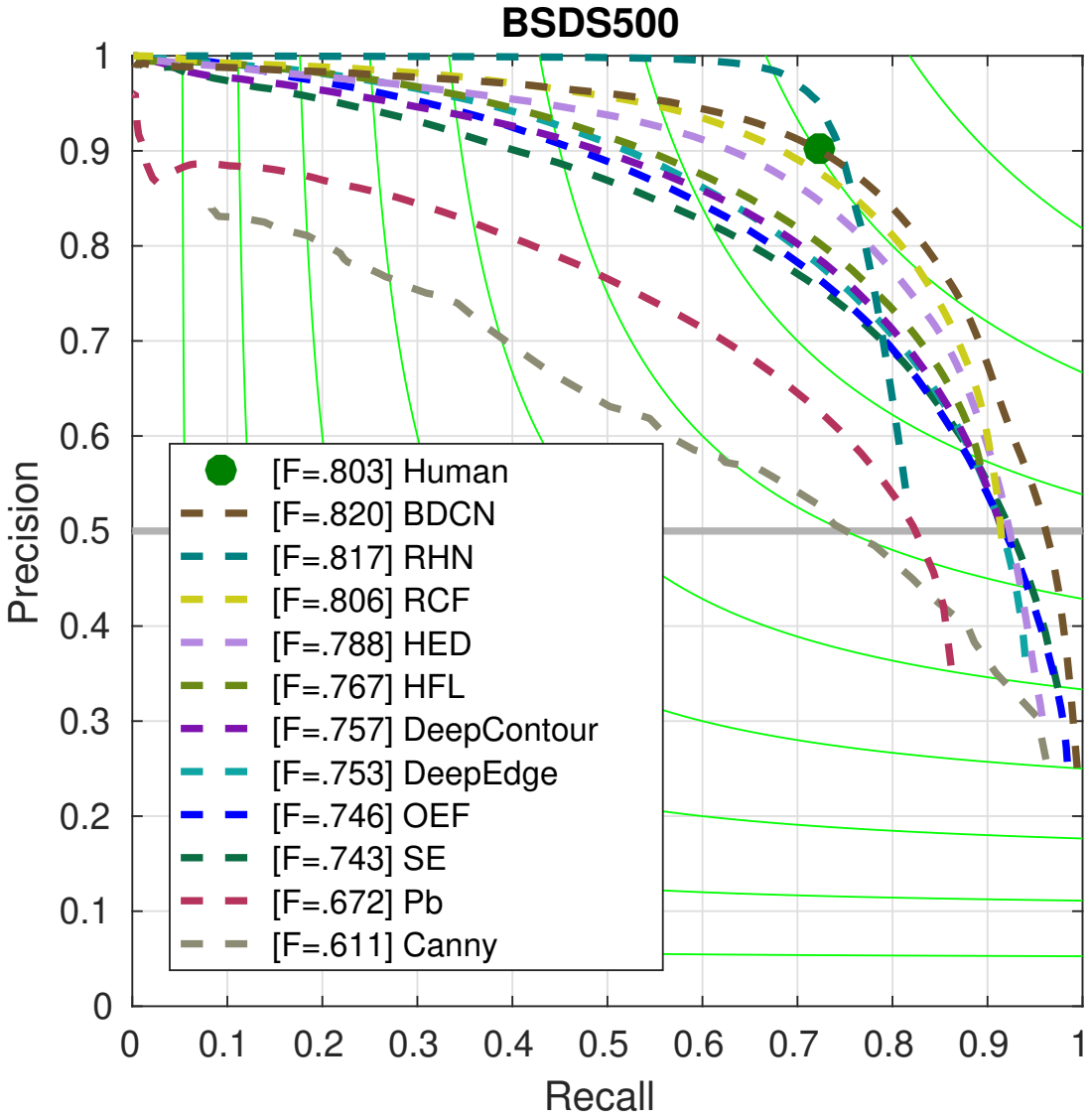

Figure 3.3: precision/recall on the BSDS500 dataset.

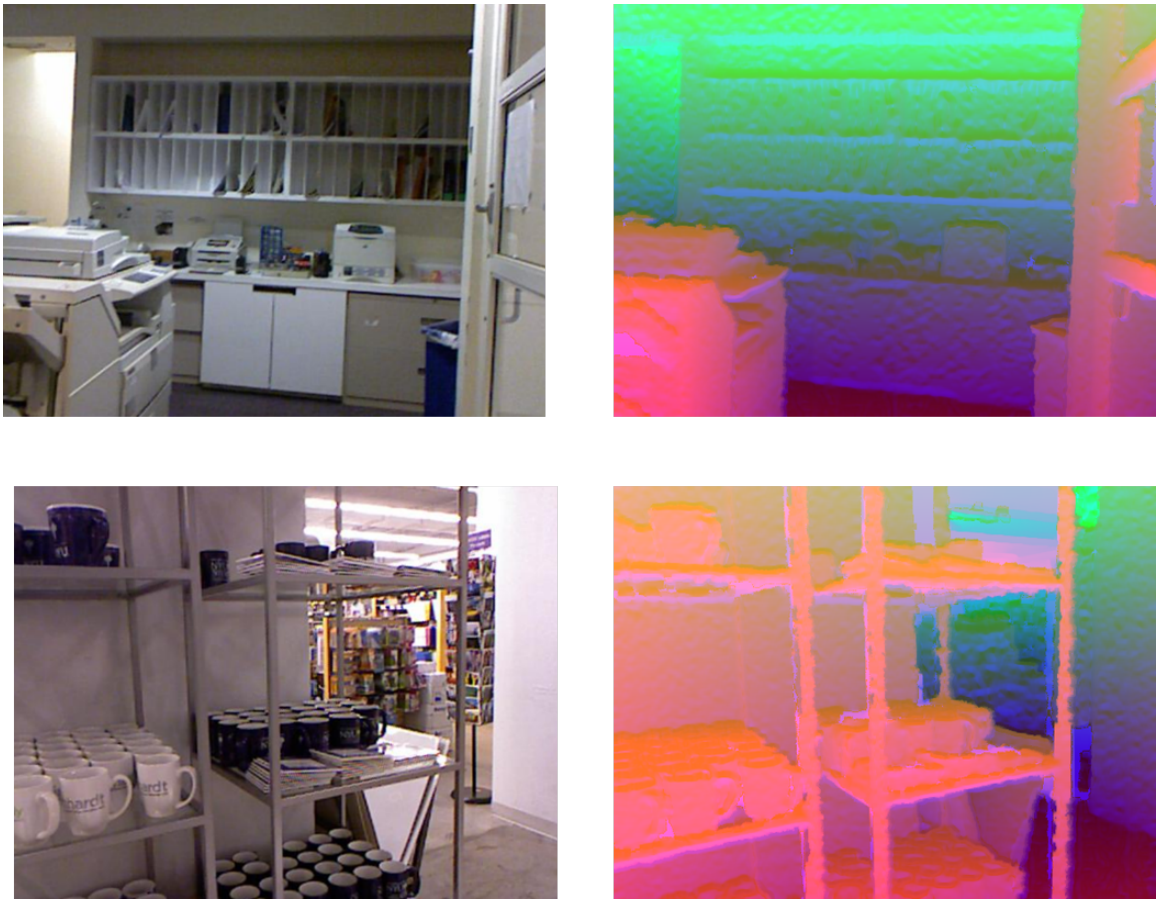

Figure 3.4: Two examples of pairs of RGB and depth images from the NYUD dataset.

Results in terms of ODS F-measure, OIS F-measure and FPS of the proposed RHN technique for the NYUD dataset are given in Table 3.2. For the purpose of comparison, results of the edge detection techniques, SE [78], OEF [79], gPb+NG [81], SE+NG+ [82], DexiNed-f [35], HED [83], RCF-VGG16 [34], RCF-ResNet50 [34], BDCN-VGG16 [36], and BDCN-ResNet50 [36], are also listed in this table.

First of all, it is observed from this table that regardless of the technique used, both the ODS F and OIS F measures are improved when the maps obtained from the RGB and HHA models are averaged over that using either RGB or HHA model alone, and hence, we use these improved measures in our comparison of the various methods.

It is seen from this table that the values of the ODS and OIS F-measures provided by the proposed RHN (RGB + HHA) are, respectively, 0.007 and 0.009 higher than that provided by the RCF-VGG16 technique and they are 0.005 and 0.006 higher than that provided by the BDCN-VGG16 technique. This improved performance of the proposed technique is achieved with 11.5 M parameters, as compared to 14.8 M and 16.3 M parameters provided respectively by the RCF-VGG16 and BDCN-VGG16 techniques. It is also to be noted that even though the performance of the residual learning-based techniques is superior to that of the proposed technique, the number of parameters employed by the proposed technique is much lower than that provided by the RCF-ResNet50 and BDCN-ResNet50 techniques. Figure 3.5 presents the precision-recall curves of the proposed RHN and some of the other methods used in our comparison on the NYUD dataset.

Edges Versus Boundaries: Multicue Dataset

In the case of the previous two datasets, BSDS500 and NYUD, all the networks including the proposed one were trained using a single set of ground-truth images. In these two datasets, the ground-truths are, in fact, neither fully edge maps nor fully boundary maps.

Table 3.2: Comparison to The State-of-The-Art Methods on NYUD

Method	ODS F-measure	OIS F-measure	No. of parameters in Millions	FPS
SE [78] (RGB+HHA)	0.695	0.708	-	5
OEF [79] (RGB+HHA)	0.651	0.667	-	1/2
gPb+NG [81] (RGB+HHA)	0.687	0.716	-	1/375
SE+NG+ [82] (RGB+HHA)	0.706	0.734	-	1/15
DexiNed-f [35] (RGB+HHA)	0.658	0.674	-	-
RCF-ResNet50 [34] (RGB+HHA)	0.781	0.793	-	7
BDCN-ResNet50 [36] (RGB+HHA)	**0.788**	**0.802**	28.7	-
HED [30] (HHA)	0.681	0.695	14.7	20
HED [30] (RGB)	0.717	0.732	14.7	20
HED [30](RGB+HHA)	0.741	0.757	14.7	10
RCF-VGG16 [34] (HHA)	0.703	0.717	14.8	20
RCF-VGG16 [34](RGB)	0.743	0.757	14.8	20
RCF-VGG16 [34] (RGB+HHA)	0.765	0.780	14.8	10
BDCN-VGG16 [36] (HHA)	0.708	0.720	16.3	-
BDCN-VGG16 [36](RGB)	0.748	0.763	16.3	-
BDCN-VGG16 [36] (RGB+HHA)	0.767	0.783	16.3	-
RHN (HHA)	0.711	0.721	**11.5**	**24**
RHN (RGB)	0.751	0.762	**11.5**	**24**
RHN (RGB+HHA)	0.772	0.789	**11.5**	12

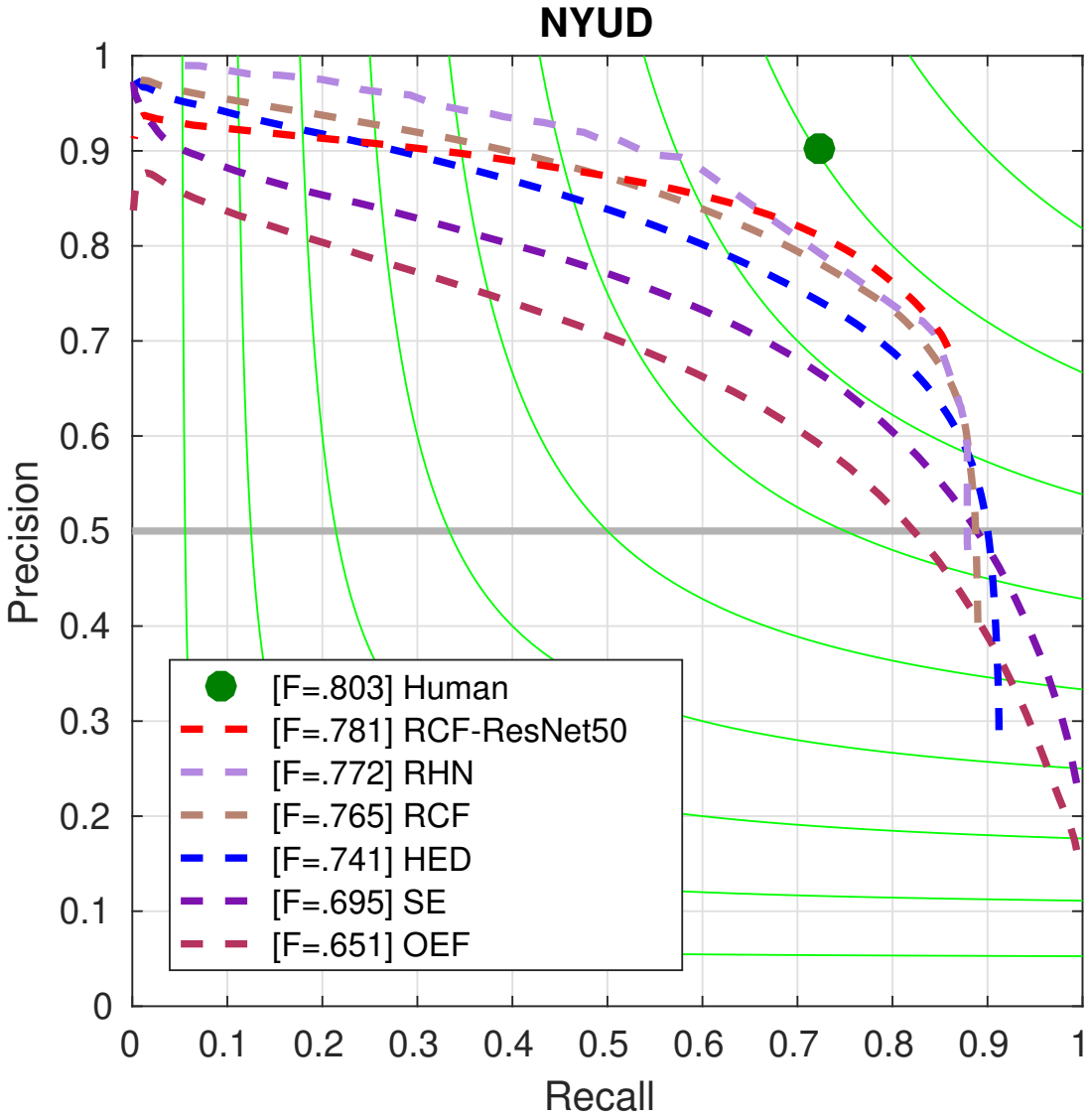

Figure 3.5: precision/recall on the NYUD dataset.

Hence, the trained networks are capable of detecting only the maps of the objects in the input image guided by the ground-truth maps. The Multicue dataset [65] has two sets of hand-annotations: one containing the entire edge map and the other one containing only the boundary map corresponding to each image in the dataset. Hence, the deep edge detection networks can have two trained networks depending on the set of ground-truth maps used for training. Hence, the Multicue dataset enables us to use one of the two trained networks depending on whether we want to determine a boundary map or an edge map for a given image. The Multicue dataset contains 100 video sequences of different scenes with the frame spatial resolution of 1280 by 720 pixels. Each sequence contains 10 frames of the left view and 10 frames of the right view of the scene. A set containing the training and test images was then formed by taking the last frame of the left views from each of the sequences [65]. Thus, this set has a total of 100 images to be used for the training of the networks and testing of the algorithms. Each of the 100 images in the set are manually labeled for two annotations, one giving a ground-truth boundary map and the other a ground-truth edge map.

In our experiments, the training and test set is randomly split into 80 images for training and 20 images for testing. We augment the images and the corresponding ground-truth labels in the training set by scaling them by 75% and 125% and rotating them by 90°, 180°, and 270°. Since the resolution of the image in this set is very high, we randomly crop them into 500 x 500 sub-images. The process of obtaining the training and test sets as described in this paragraph is repeated three times in order to have three different training and test sets. Each training set is used individually to obtain three different models of the proposed network trained to extract an edge map and again three different models to extract a boundary map. The ODS and OIS scores are obtained for each of the three models trained to obtain an edge (boundary) map for each of the images in the test set. The final values of these metrics are then obtained by averaging the results obtained from the three trained models

over all the 20 images in the test set.

Results in terms of ODS F-measure and OIS F-measure of the proposed RHN technique are given in Tables 3.3 and 3.4. For the purpose of comparing the performance of the proposed network with that of the three techniques, namely, HED [30], RCF-VGG16 [34], and BDCN-VGG16 [36], the results for the measures of these techniques are also listed in these tables. It is seen from these tables that the proposed RHN outperforms the other three networks in terms of the two measures both in edge detection and boundary detection.

Table 3.3: Performance comparison with state-of-the-arts for boundary detection using Multicue dataset

Method	ODS F-measure	OIS F-measure
HED [30]	0.814	0.822
RCF- VGG16 [34]	0.817	0.825
BDCN-VGG16 [36]	0.836	0.846
RHN	**0.841**	**0.856**

Table 3.4: Performance comparison with state-of-the-arts for edge detection using Multicue dataset

Method	ODS F-measure	OIS F-measure
HED [30]	0.851	0.864
RCF- VGG16 [34]	0.857	0.862
BDCN-VGG16 [36]	0.891	0.898
RHN	**0.896**	**0.905**

3.3.2 Qualitative Performance and Comparison

In this section, we give some examples of the visual quality of the edge maps obtained from using the proposed scheme and compared with the visual quality of the edge maps obtained by using the RCF technique [34]. We also illustrate using the proposed network

the significance of suitable training of an edge detection network depending on whether the objective of the network is to obtain a boundary map or an edge map.

Figure 3.6 shows the maps of the images obtained by using the RCF-VGG16 [34] and proposed RHN networks trained separately on BSDS500 leading two different trained models. In this figure, the first row show two different images from the BSDS500 dataset and the corresponding. The maps in row b of the figure are the ground-truth maps, whereas those in rows c and d are the ones obtained by using, respectively, the RCF-VGG16 and proposed RHN networks. It is seen from this figure that the edge maps obtained by using the proposed network are closer to the ground-truth maps than the maps obtained by the RCF-VGG16 technique are. For example, it is seen from the maps of the first column of this figure that the children's eyes and chin of the middle child are better represented in the map obtained by the proposed scheme. Also, it is seen from the maps of the second column of this figure that the edge of the shadow of the barn roof on its wall is sharper in the map obtained by using the proposed scheme.

Figure 3.7 shows the maps of the images obtained by using the RCF-VGG16 [34] and proposed RHN networks trained separately on NYUD datasets. In this figure, the first row show two different images from the NYUD dataset and the corresponding maps. The maps in row b of the figure are the ground-truth maps, whereas those in rows c and d are the ones obtained by using, respectively, the RCF-VGG16 and proposed RHN networks. It is also seen from this figure that the edge maps obtained by using the proposed RHN are closer to the ground-truth maps than the maps obtained by the RCF-VGG16 technique are. For example, it is seen from the maps of the first column that the left edge of the curtain in the deeper background of the image and the ring on the wall of the foreground do appear in the map provided by the proposed scheme, whereas they are completely missed out in the map

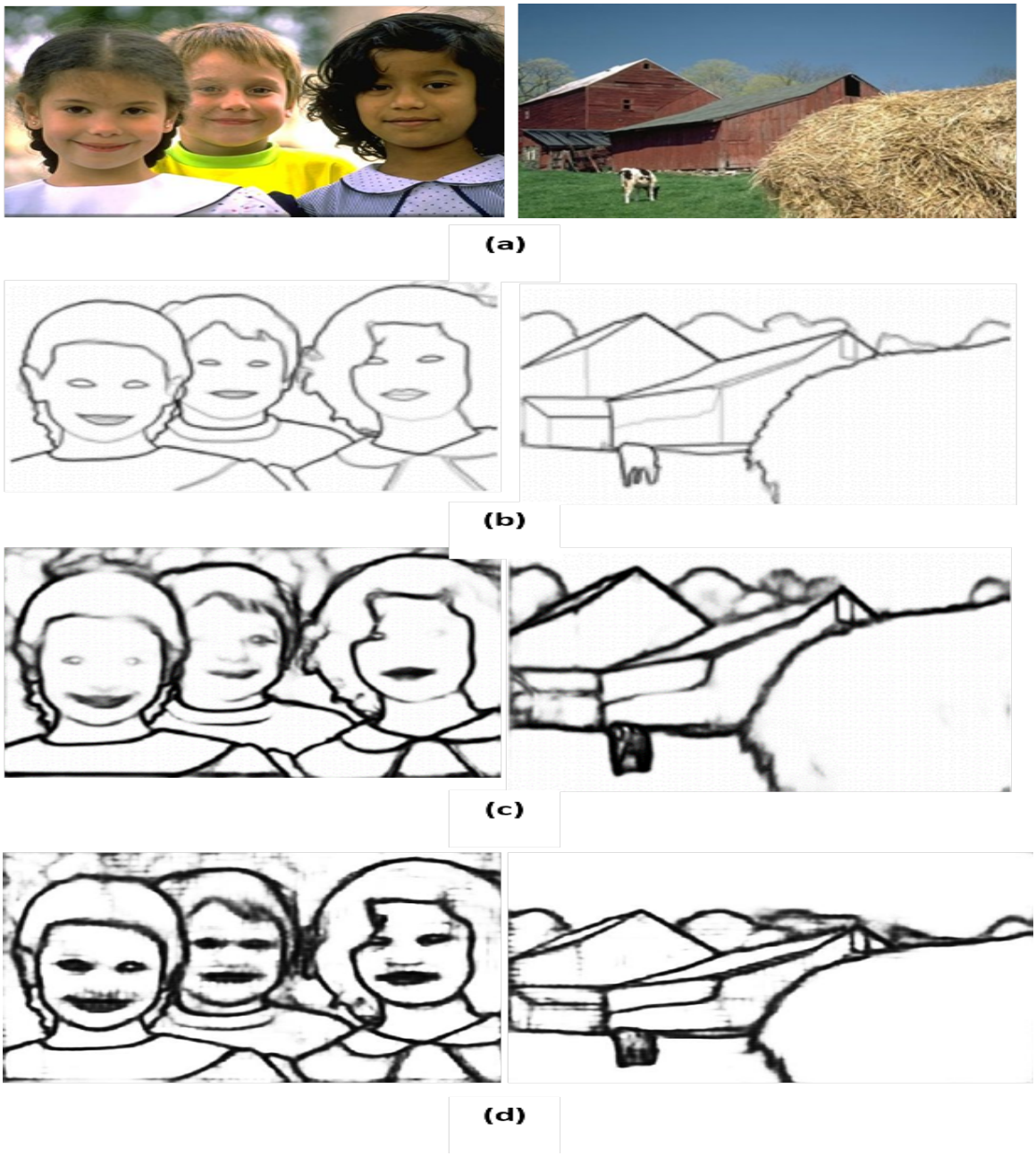

Figure 3.6: The edge maps of two of the images from the BSDS500. (a) Original images. (b) Ground-truth edge maps. (c) Edge maps using RCF. (d) Edge maps using the proposed RHN.

produced by RCF-VGG16 network. Finally, it is seen from the maps of the second column that the edges in the lamp pole are cleaner and sharper in the map obtained by using the proposed RHN.

We now illustrate the impact of a suitable network training on the subjective quality of the maps produced. For this purpose we train the proposed RHN network on the two training sets of the Multicue dataset and the BSDS500 dataset providing three different trained models. Figure 3.8 shows the maps of three images obtained by using the three trained models. In this figure, each of the three columns relate to a different image. The original images are shown in row a, whereas the corresponding ground-truth boundary and edge maps are shown, respectively, in rows b and c. Row d shows the maps obtained from the RHN network trained using the Multicue training set to detect boundaries of the images, whereas row e shows the maps obtained from the RHN network trained using the Multicue training set to detect edges of the images. Row f shows the maps obtained from the RHN network trained using the BSDS500 training set which is not constructed to detect specifically either the boundary map or the edge map entirely. It is seen from this figure that the maps in row f are neither as good as the boundary maps in row d nor are they as good as the edge maps in row e.

3.4 Summary

Several techniques have been developed for the task of edge detection based on the VGG16 network, since the convolutional layers of the networks of such schemes have fewer parameters than those of the existing residual networks. However, their performance is inferior to that of the residual techniques. This reduced performance is due to the fact that the spatial resolutions of the feature maps in this architecture significantly decrease between the

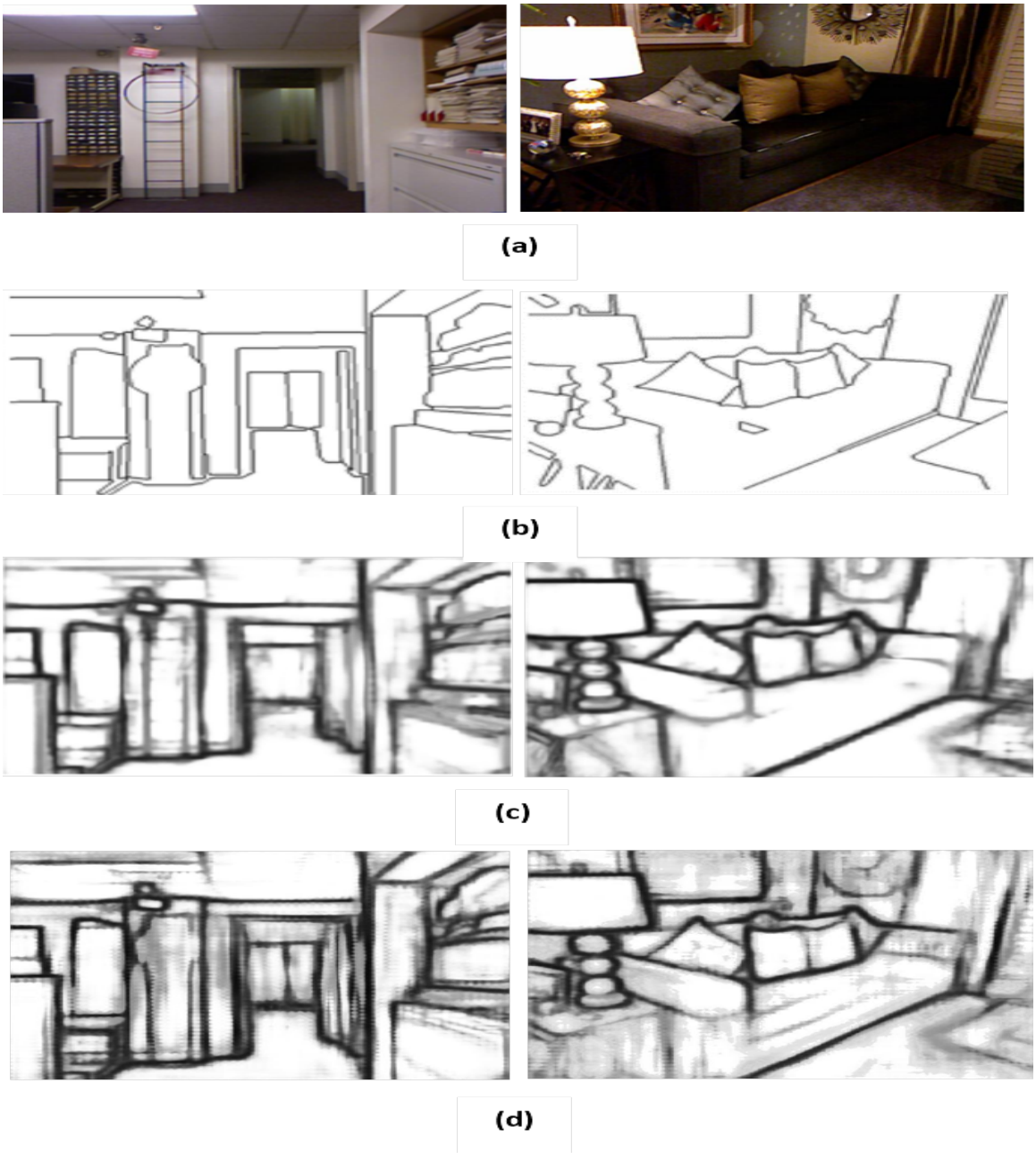

Figure 3.7: The edge maps of two of the images from the NYUD datasets. (a) Original images. (b) Ground-truth edge maps. (c) Edge maps using RCF. (d) Edge maps using the proposed RHN.

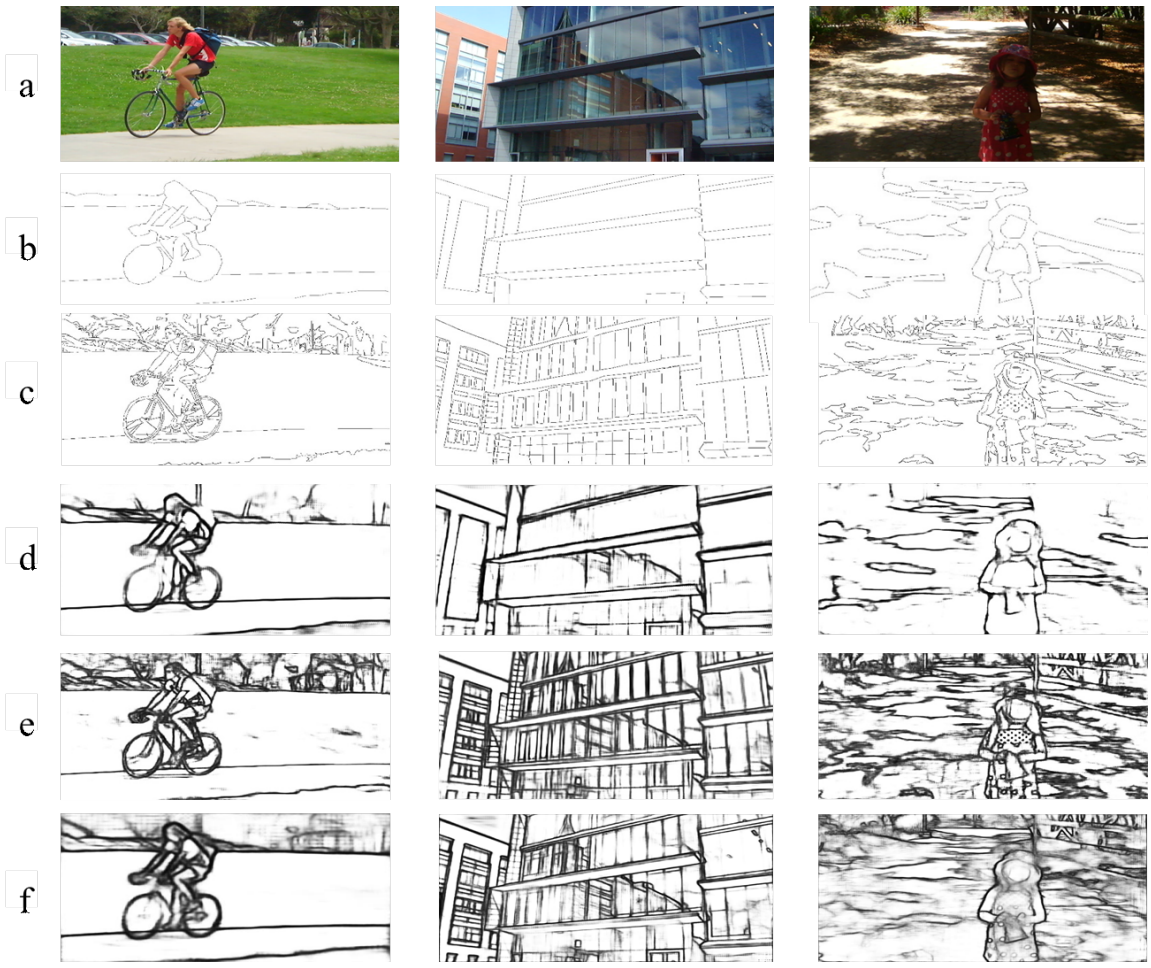

Figure 3.8: The boundary and edge maps of three images. (a) Original images. (b) Ground-truth boundary maps. (c) Ground-truth edge maps. (d) Boundary maps from the proposed RHN trained by using the Multicue dataset. (e) Edge maps from the proposed RHN trained by using the Multicue dataset. (f) Maps from the proposed RHN trained by using the BSDS500 dataset.

convolutional layers of the first block and that of the last block. Further, the deeper layers cannot learn some of the information captured by the shallower layers. In this chapter, a mechanism of residual learning has been introduced in the VGG16 architecture to obtain a deep edge detection convolutional neural network to provide superior performance while still preserving its lower complexity character. Extensive experiments have been carried out on the proposed edge detection network using several benchmarks datasets to evaluate its performance and complexity. It has been shown that the performance of the proposed network outperforms most of the existing techniques in terms of ODS F-measure and OIS F-measure and provides edge maps of superior visual quality. In terms of the network complexity, the number of parameters used by the proposed RHN is lower than that of all the existing techniques.

Chapter 4

A Reduced Complexity Residual Deep Neural Network for Edge Detection

4.1 Introduction

Even though the complexity of RHN, which proposed in Chapter 3, is lower than that of other VGG-16 based networks, it is still high in view of employing a large number of parameters; further, its performance is lower than that of the BDCN in view of its limited capacity in extracting a rich set of features. The objective of this chapter is to build further on the work of Chapter 3 and develop VGG-16 based residual deep convolutional neural networks having complexity that are not only lower than that of ResNet-based networks, but also substantially lower than those of the existing VGG16-based networks and a performance, which surpasses that of all the edge detection networks [84]. Generally, deep convolutional neural networks are made to yield improved performance by designing them so as to extract richer and more useful features from the input data by increasing the depth, width, receptive fields, and multiresolution capability of the networks. However, this course of action results in substantially increased complexity of the networks. The main idea of this chapter that has not been utilized earlier in the design of edge detection

networks is the use of fire modules in the convolutional layers in the design of the proposed networks. In a fire module, which was originally introduced in [85] for compressing the AlexNet architecture for the task of image classification, the number of parameters is decreased without losing the accuracy of the network by first squeezing and then expanding the number of feature maps in a convolutional layer of AlexNet. The rest of this chapter is organized as follows: In Section 4.2, the proposed architecture of the baseline network is presented. In Section 4.3 experimental results providing the performance evaluation and complexity analysis of the proposed edge detection network, along with a comparison of these results with that of some of the other networks commonly used for edge detection, are presented. Two other modified versions of the proposed baseline network are also presented and discussed. Finally, some concluding remarks are made in Section 4.4.

4.2 The Proposed Method

In this section, we develop our proposed residual edge detection network based on the VGG-16 architecture. The VGG16 architecture was originally developed for the task of image classification, and it composed of a total of 13 convolutional layers, 5 pooling layers, and 3 fully connected layers. This network can be divided into six blocks, as shown in Fig. 2.5. The size of the filters used in all the convolutional layers is 3x3. Each of the first two blocks contains two convolutional layers and one pooling layer. Each of the succeeding three blocks contains three convolutional layers and one pooling layer. The last block of the VGG16 network contains three fully connected layers, which perform the classification task. Hence, in the development of our proposed edge detection network, we will use the first five blocks as a founding architecture. Since the fire module [85] constitutes a very important component in the development of our edge detection network, we start this section with a brief review of this module in Section 4.2.1 The architecture of the

proposed network is developed in Section 4.2.2 The details on the numbers and sizes of the filters used in the various layers of the network are also given in this subsection. Finally, the training details of the proposed network are provided in Section 4.2.3.

4.2.1 Fire module

The schematic of a fire module is shown in Fig. 4.1. This module consists of two convolutional layers, a squeeze convolutional layer containing only 1x1 filters and an expand convolutional layer containing a mix of 1x1 and a larger size filters. Since a convolutional layer cannot accommodate filters of mixed sizes, the expand layer of the fire module is composed of two convolutional layers, the first of which uses 1x1 filters and the second one uses filters that are of larger size. The outputs of these two layers are then concatenated to obtain the output of the fire module. The number of filters in the squeeze layer is kept low making the number of channels input to the expand layer small so that the complexity of processing in the expand layer resulting from its use of larger size filters is not adversely affected. This small number of channels input to the expand layer allows for the use of a large number of filters in the expand layer, which results in a large number of feature maps. The ratio of the number of filters in the squeeze layer to that in the expand layer, called the squeeze ratio (SR), is an important parameter whose value is chosen so as to have a trade-off between the performance accuracy and the complexity of the network using the module.

4.2.2 The architecture of the proposed network

The architecture of the proposed network for edge detection, referred to as low-complexity residual deep neural network (LRDNN), is depicted in Fig. 4.2. The network consists of 13 blocks. Blocks 1, 2, 4, 5, and 7 of the proposed network employ fire modules, instead of

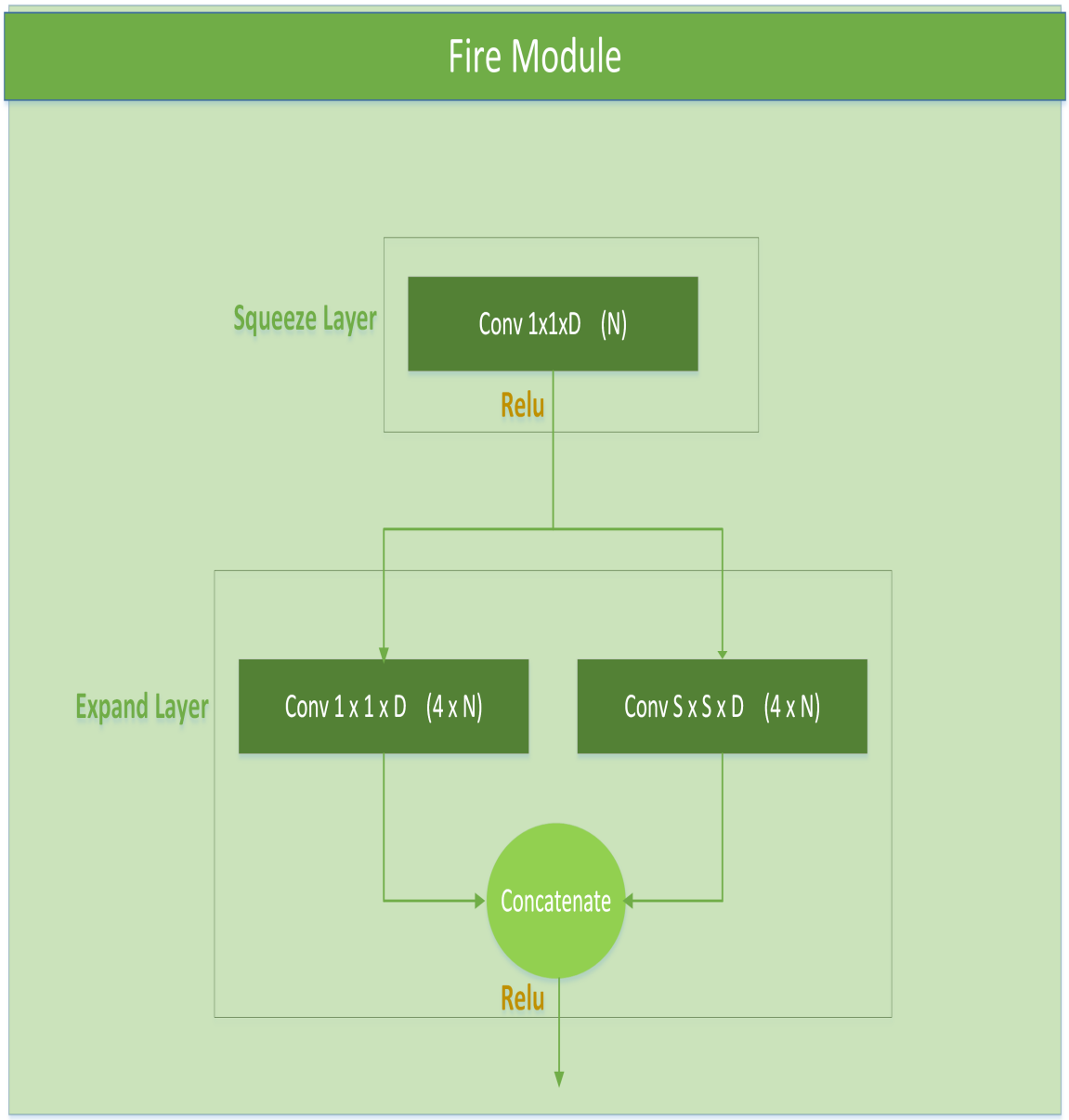

Figure 4.1: Organization of convolution filters in the fire module.

the regular convolutional layers as used in the original VGG-16 architecture, except for the first block in which only the second convolutional layer is replaced by a fire module. Our main motivation of using fire modules in the proposed edge detection network is that they increase the depth of the network, and at the same time, reduce the network complexity. In the proposed network, similar to [85] and [86], the value of the squeeze ratio for the fire module is chosen to be 1/8, and the numbers of 1x1 and the larger size filters in the expand layer are chosen to be equal. Each of the blocks 1, 2, 4, and 5 also use a max pooling layer. As in many other convolutional networks, the purpose of using a max pooling layer is to reduce the spatial resolution of the feature maps produced by a convolutional layer so as to allow the succeeding convolutional layers to employ a larger number of filters in order to extract additional robust features without adversely affecting the complexity of the network. Blocks 3 and 6 of the proposed architecture are the information fusion blocks in which the maps resulting from two different blocks are resolution-wise matched using a transposed convolution layer and then added. Blocks 8 to 13 are used to facilitate a deep supervision of the network for its training, as to be explained in Section 4.2.3. Each of these blocks has a reconstruction convolutional layer and a balanced cross-entropy loss layer. With the exception of blocks 8 and 13, each of these blocks also has a transposed convolutional layer. The purpose of each of the blocks 8-12 is to construct a loss function between the labeled ground-truth and the edge map constructed from the feature maps (S_1, S_2, S_3, S_4, and S_5) generated by each of the blocks, 1, 2, 4, 5, and 7. However, the purpose of block 13 is to construct a loss function between the ground-truth and the edge map obtained by concatenating the edge maps generated by blocks 8-12. The convolutional operations in each of the convolution layers of the proposed architecture are followed by a ReLu activation function.

In block 1, the numbers of filters in the convolutional layer (Conv1), the squeeze layer

and the expand layer of Fire Module 1, are chosen to be 64, 8, and 64, respectively. In block 2, the numbers of filters used in the squeeze and expand layers are 16 and 128, respectively. Since residual learning schemes have been used to extract richer sets of features, this idea is also employed in blocks 3 and 6. To accomplish this, a transposed convolutional layer is used to match the spatial resolution and the number of maps of the outputs of blocks 1 and 2 (blocks 4 and 5) and then to add the first output with the resulting second output.In block 4, the numbers of filters used in the squeeze and expand layers are 32 and 256, respectively. The size of the filters in the second convolutional layers of the expand layers used in all of the fire modules of blocks 1, 2, and 4 is 3x3. Since the purpose of the network is, to finally provide edge maps, that is, maps containing global low level features (edges and corners of the objects in the image), the sizes of the filters in the second convolutional layers of the expand layers in blocks 5 and 7 are, respectively, increased to 5x5 and 7x7. The use of larger size filters in these blocks would facilitate extraction of such global low level features by providing denser connections between the feature maps that are input to these blocks. In blocks 5 and 7, the numbers of filters used in the squeeze and expand layers are 64 and 512, respectively.

4.2.3 Training of the proposed network through deep supervision

The blocks 8, 9, 10, 11, 12, and 13 generate six loss functions, L_1, L_2, L_3, L_4, L_5 and L_6, respectively. The loss functions L_1, L_2, L_3, L_4, and L_5 are generated using the side feature maps S_1, S_2, S_3, S_4, and S_5, generated by blocks, 1, 2, 4, 5 and 7, respectively, and the labeled ground-truth. It should be noted that these 5 side feature maps have 64, 128, 256, 512, and 512 maps with spatial resolutions of H x W, H/2 x W/2, H/2 x W/2, H/4 x W/4 and H/4 x W/4, respectively. To construct the loss functions L_2, ..., L_5, each of the feature maps, S_2, ..., S_5, must be converted into a single map with a spatial resolution of H x W. This is

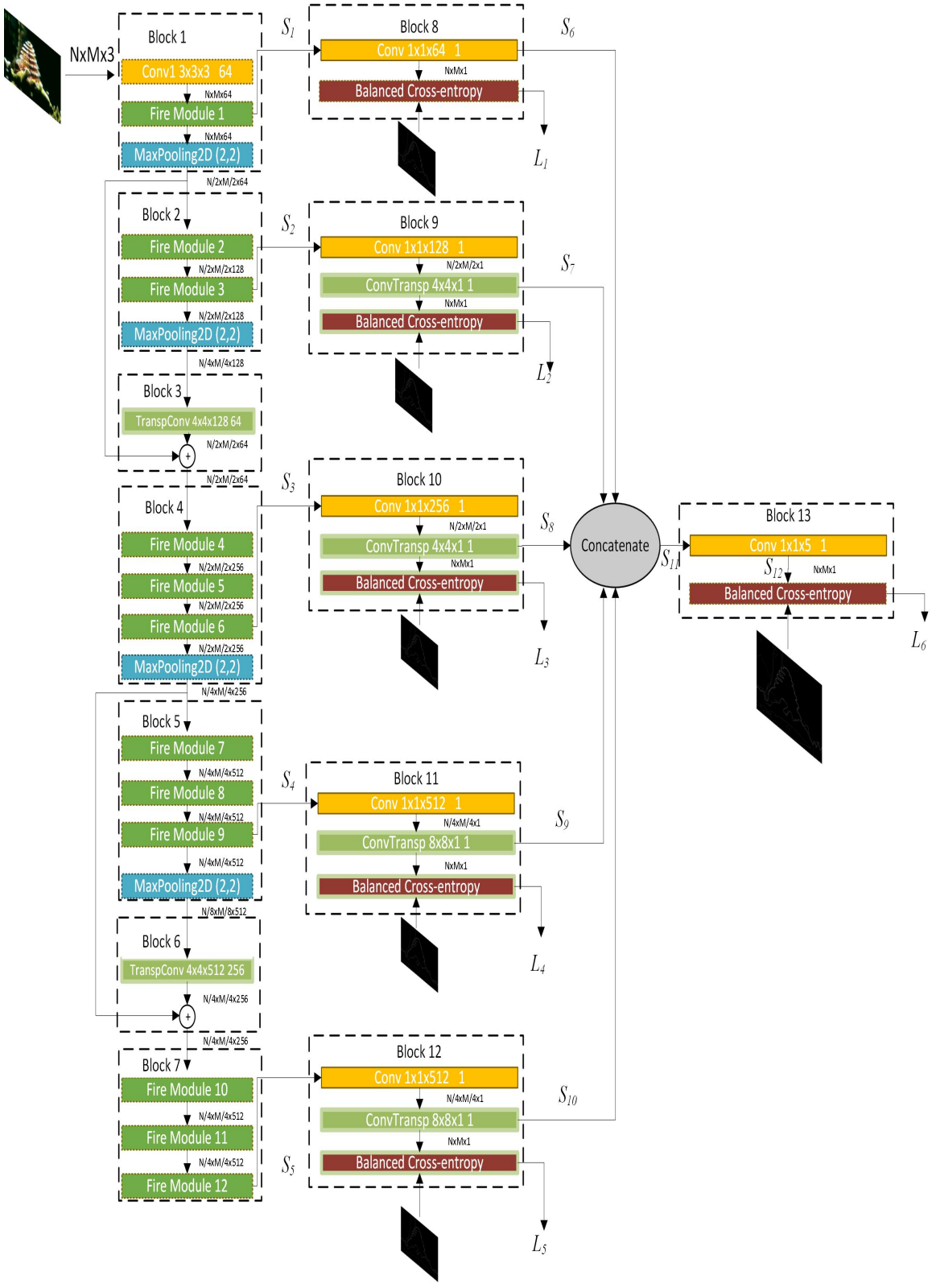

Figure 4.2: Architecture of the proposed edge detection convolution network.

accomplished by a convolutional filter and the transposed convolutional layer of the blocks 9-12, which produce the feature maps S_7, S_8, S_9 and S_{10}, respectively. However, since the resolution of the maps, S_1, is already H x W, these maps are combined into a single map by using only a convolutional filter to generate the feature map S_6. Finally, the feature maps S_6, ..., S_{10} are concatenated and then the resulting feature maps S_{11} are combined into a single map, S_{12}, by using a convolutional filter in block 13. As in Section 3.2 of Chapter 3, six balanced cross entropy loss functions denoted by L_i, i = 1, ..., 6, are produced using the edge maps S_1, ..., S_5 and S_{11}, respectively, and the ground truth. The six loss functions L_1 , ..., L_6 are combined to obtain a single cost function as given by Eqn. (3.2). This single cost function is then used for the deep supervision of the training of the network given in Figure 4.2.

4.3 Results and comparison with the state-of-the-art edge detection schemes

In this section, experiments are performed on the scheme proposed in this chapter to establish its effectiveness. The performance of the scheme is measured in terms of the optimal dataset scale (ODS) and optimal image scale (OIS) metrics and the complexity of the scheme in terms of the network parameters. For training the networks, BSDS500 [63], a mix of BSDS500 and PASCAL-Context datasets, and NYUD [64] datasets are used. For the training of all the networks, we augment the images of the datasets and their ground-truths, by applying only the operations of scaling, rotation, and flipping as done in Chapter 3.

The adaptive moment estimation technique [68] is used to update the network parameters. The value of the weight decay is set to $2x10^{-4}$, the learning rate to 10^{-6} and a batch size of 10 is chosen in all of the experiments. In order to improve the performance and stability

of the network, a batch normalization is carried out on the feature maps obtained after the operation of each ReLU [70–72]. Since the loss function used for training does not have appreciable change in its value after 45 epochs, we set the maximum number of epochs to be 45. The Keras library [73], back ended by the TensorFlow package [74], is used for implementation on a hardware platform that uses an Intel(R) Core(TM) i7-7700K CPU @4.2 GHz, 32 GB of RAM and an NVIDIA GeForce GTX 1080 GPU.

As in all other deep learning methods, the standard non-maximal suppression (NMS) technique is used to thin the edges detected for evaluating the estimated edge maps.

In Section 4.3.1, an ablation study is carried out by performing experiments using LRDNN and its two variants, which do not use the fire modules, on the images of the BSDS500 and NYUD datasets. In Section 4.3.2, experiments are performed on the proposed LRDNN using the BSDS500 dataset and the performance results are compared with that of some of the other edge detection networks. In Section 4.3.3, we developed a modified version of LRDNN in order to improve its performance. Experiments are also performed on the modified network using the BSDS500 dataset to evaluate its performance. In Section 4.3.4, the performance of the LRDNN and M-LRDNN networks are evaluated by carrying out experiments using the NYUD dataset. In Section 4.3.5, the qualitative performance is presented. Finally, in Section 4.3.6, we developed an ultra-lightweight version of the proposed LRDNN network and compare its performance with another lightweight network in the literature by conducting experiments using the BSDS500 dataset.

4.3.1 Ablation study on the proposed baseline architecture

In this subsection, we conduct an ablation study to examine the impact of the main idea, i.e. the use of the fire modules in the design of our proposed baseline architecture, LRDNN, on its performance. We carry out this study in two parts. For the first part, we construct a

variant M1 of LRDNN in which instead of using the fire modules in the various blocks of the network, we simply use the standard convolutional layers. However, in order to keep the complexity, in terms of the number of parameters, of M1 to be about the same as that of LRDNN, the number of filters used by the convolutional layers of blocks 2, 4, 5, and 7 are reduced to 64, 128, 128 and 128, respectively; whereas the number of filters in the convolutional layers of block 1 is retained to be 64. For the second part, we construct another variant of LRDNN, namely M2, in which we increase the number of filters of the convolutional layers in M1 so that the performance of the resulting M2 is as close as possible to that of the baseline LRDNN. Through an empirical study, we determined that by using 64, 128, 256, 512 and 512 filters in the convolutional layers of blocks 1, 2, 4, 5, and 7, respectively, this goal is achieved. Each of these two variants are trained and tested individually on two datasets, namely, BSDS500 and NYUD. The performance results and the number of parameters of the two variants, along with that of our proposed LRDNN baseline architecture are given in Table 4.1. It is seen from this table that the performance of M1 (which has the same number of parameters as that of LRDNN) drops down from that of the baseline architecture. It is also seen from this table that when the fire modules are replaced by the standard convolutional layers, the number of parameters in the resulting variant network M2 has to be raised by more than 10 folds in order to give a performance that is still somewhat lower than that of the baseline architecture. Therefore, the results of this ablation study clearly demonstrate the effectiveness of our main idea in the design of the proposed edge detection architecture, i.e., the use of the fire modules.

4.3.2 Performance of the LRDNN network on the images of BSDS500 dataset

In this subsection, we obtain two sets of results on the LRDNN network. The results of the first set are obtained, when the network is trained using the BSDS500 dataset and that

Table 4.1: Experiments on BSDS500 and NYUD datasets to study the impact of fire modules on the proposed network.

Method	ODS F-measure		OIS F-measure		No. of parameters
	BSDS500	NYUD	BSDS500	NYUD	
Variant M1	0.791	0.767	0.811	0.779	**4.3 M**
Variant M2	0.806	0.778	0.825	0.793	50.1 M
LRDNN	**0.808**	**0.784**	**0.827**	**0.797**	**4.3 M**

of the second set obtained, when it is trained using a mix of the BSDS500 and PASCAL Context datasets [76]. All the edge detection schemes that we have used in our comparison have chosen the value of the localization tolerance parameter to be 0.0075 for the BSDS500 dataset [30–37, 61, 65, 70] for the calculation of the ODS and OIS F-measures. We also use the same value of this parameter for the computation of the two measures.

The results in terms of the ODS F-measure, OIS F-measure and the number of parameters obtained by the proposed architecture (LRDNN) are given in Table 4.2. For the purpose of comparison, the results obtained by using some of the recently developed deep learning techniques, namely, the HED-VGG16 network [30], RDS-VGG16 [31], CED-VGG16 [33], CED-ResNet50 [33], RCF-VGG16 [34], RCF-ResNet101 [34], RHN-VGG16 [61], BDCN-VGG16 [36], BDCN-ResNet50 [36], DexiNed-f [35], NAO-Multi-scale [42], and BFENet-VGG16 [37], are also listed in this table. It should be pointed out that all the networks in this table are tested using the images from the BSDS500 dataset, regardless of whether the network to have been trained using only the BSDS500 dataset or a mix of both the datasets. The networks trained using the mixed dataset are marked with a dagger (†). It is observed from this table that the performance of the proposed LRDNN network trained using only the BSDS500 dataset, in terms of the two metrics, is superior to that of all the other VGG16-based techniques, irrespective of whether these other networks are trained using only the BSDS500 dataset or the mixed dataset, except for BDCN-VGG16 when it is trained with the mixed dataset. However, it is also seen that when the proposed

LRDNN is trained using the mixed dataset, its performance becomes superior even with respect to BDCN-VGG16 trained with the mixed dataset. In terms of the network complexity, the number of parameters used by the proposed LRDNN is only 26.4 percent of the number of parameters used by this best performing network. It is also to be noted that the complexity of the proposed network is 37.4 percent of the number of parameters used by RHN-VGG16, the VGG16-based network that employs the lowest number of parameters. It is also seen from this table that the proposed LRDNN outperforms all the ResNet-based networks both in terms of the performance measures and complexity, except for BDCN-ResNet50 compared to which the performance of the proposed LRDNN is lower by 0.1 percent in terms of ODS F-measure only. However, the number of parameters employed by this technique is 28.7 M, which is significantly higher compared to 4.3 M parameters utilized by the proposed LRDNN.

In order to further improve the performance of the proposed LRDNN trained using the mixed dataset and tested individually at three different scales namely, 0.5, 1, and 1.5 on the various test images of the BSDS500 dataset. The three edge maps thus obtained for a test image are restored to the original scale and the final predicted edge map is obtained by fusing the three edge maps. This single fused map is then used to obtain the ODS and OIS values corresponding to this edge map. Table 4.3 gives the values of these metrics averaged over all the test images for the proposed LRDNN and those of the networks that also use multi-scale inputs for testing them. By comparing the results of this table with those of Table 4.2, it is seen that fusing the maps at the three different scales helps in improving the performance by increasing the values of ODS and OIS F-measures. It is also seen that the proposed LRDNN still remains the best performing network with the exception of BDCN-ResNet50††. It should be pointed out that this improvement in the performance is achieved by all the networks individually at the cost of increasing computational time by slightly

Table 4.2: Comparison of the proposed network with state-of-art methods on the BSDS500 dataset with and without the PASCAL-Context dataset. The networks trained with both the datasets are marked by †. All the networks are tested only on the images of the BSDS500 dataset.

Method	ODS F-measure	OIS F-measure	No. of parameters (in Millions)
RCF-ResNet101†	0.812	0.829	>>44
RCF-ResNet50†	0.808	0.825	>>23
BDCN-ResNet50	0.809	0.828	28.7
BDCN-ResNet50†	**0.826**	**0.840**	28.7
RDS-VGG16†	0.792	0.810	14.7
HED-VGG16†	0.788	0.808	14.7
CED-VGG16	0.794	0.811	21.4
CED-VGG16†	0.803	0.820	21.4
RCF-VGG16	0.798	0.815	14.8
RCF-VGG16†	0.806	0.823	14.8
BDCN-VGG16	0.806	0.826	16.3
BFENet-VGG16†	0.809	0.829	>>20
RHN-VGG16	0.801	0.820	11.5
RHN-VGG16†	0.817	0.833	11.5
BDCN-VGG16†	0.820	0.838	16.3
DexiNed-f†	0.729	0.745	>>18
NAO-Multi-scale†	0.814	0.831	>>44
LRDNN	0.808	0.827	**4.3**
LRDNN†	0.825	**0.840**	**4.3**

Table 4.3: Comparison of the proposed network with the state-of-art methods employing multi-scale testing strategy, where the networks are trained using a mix of BSDS500 and PASCAL-Context datasets. The networks in this table are marked by †† to indicate their training using the mixed dataset and the multi-scale testing strategy.

Method	ODS F-measure	OIS F-measure	No. of parameters (in Millions)
CED-ResNet50††	0.817	0.834	>>23
RCF-ResNet101††	0.819	0.836	>>44
BDCN-ResNet50††	**0.832**	**0.847**	28.7
CED-VGG16††	0.815	0.833	21.4
RCF-VGG16††	0.811	0.830	14.8
BFENet-VGG16††	0.822	0.843	>>20
RHN-VGG16††	0.824	0.843	11.5
BDCN-VGG16††	0.828	0.844	16.3
LRDNN††	0.830	0.846	**4.3**

more than three times.

4.3.3 A modified version of the proposed LRDNN network and its performance on the images of the BSDS500 dataset

In this subsection, we now study the impact of suitably increasing the number of layers and filters in our proposed LRDNN network, while keeping its complexity still much lower than that of the other schemes, on its performance. Specifically, we increase the depth of the proposed network, the baseline network, by modifying it to have one additional block (block 8) for extracting more features. To facilitate the extraction of global low level features in the proposed baseline network, LRDNN, the receptive fields of the filters were increased in its deeper layers. Specifically, 5x5 and 7x7 filters, instead of 3x3 filters, were used in the blocks 5 and 7, respectively. Therefore, in the modified network the larger size filters, namely the 5x5 and 7x7 filters, are now used for blocks 7 and 8, respectively, and the size of the filters in all the other convolutional blocks remains 3x3 as in the baseline architecture. Further, the numbers of filters in the squeeze layers of blocks 4, 5, and 7 are

increased to 64, 96, and 128, respectively. Thus, the number of filters in the corresponding expand layers now need to be increased to 512, 768, and 1024. The number of filters used in the squeeze and expand layers of the new block (block 8) of the modified network are, respectively, 32 and 256. Since in this chapter, the deep supervision is used for network training, we utilize the feature maps produced by block 8 by constructing one additional loss function L6 in the same way as was done for constructing each of the other loss functions, $L_1, \ldots, L_5$, were produced by introducing another block, block 14. The architecture of the modified edge detection network, which we refer to as M-LRDNN, is shown in Fig. 4.3.

In view of adding additional blocks, namely, blocks 8 and 14, and the use of increased number of filters in blocks 4, 5, and 7, the complexity of M-LRDNN becomes almost double of that of the baseline architecture LRDNN, but still significantly lower (almost one quarter) than that of the BDCN-ResNet50, the best performing edge detection architecture in the literature.

Our M-LRDNN network is trained using BSDS500 and PASCAL-Context datasets and tested using the 200 images of BSDS500. Table 4.4 gives the performance of M-LRDNN, along with that of BDCN-ResNet50, BFENet-VGG16, and our baseline LRDNN network, in terms of the ODS and OIS F-measures as well as the number of parameters. It is seen from this table that the modifications in the baseline network have resulted in improving the values of the ODS and OIS F-measures by 0.009. It was seen from Table 4.3 that the performance of BDCN-ResNet50 on images with a single scale or multi scales was superior to that of our baseline network on images with a single scale. However, it is now seen from Table 4.4 that the performance of M-LRDNN on images with a single scale is superior even to that of BDCN-ResNet50 on images with a single scale or multi scales. Thus, M-LRDNN on images with single scale outperforms all the networks in the literature

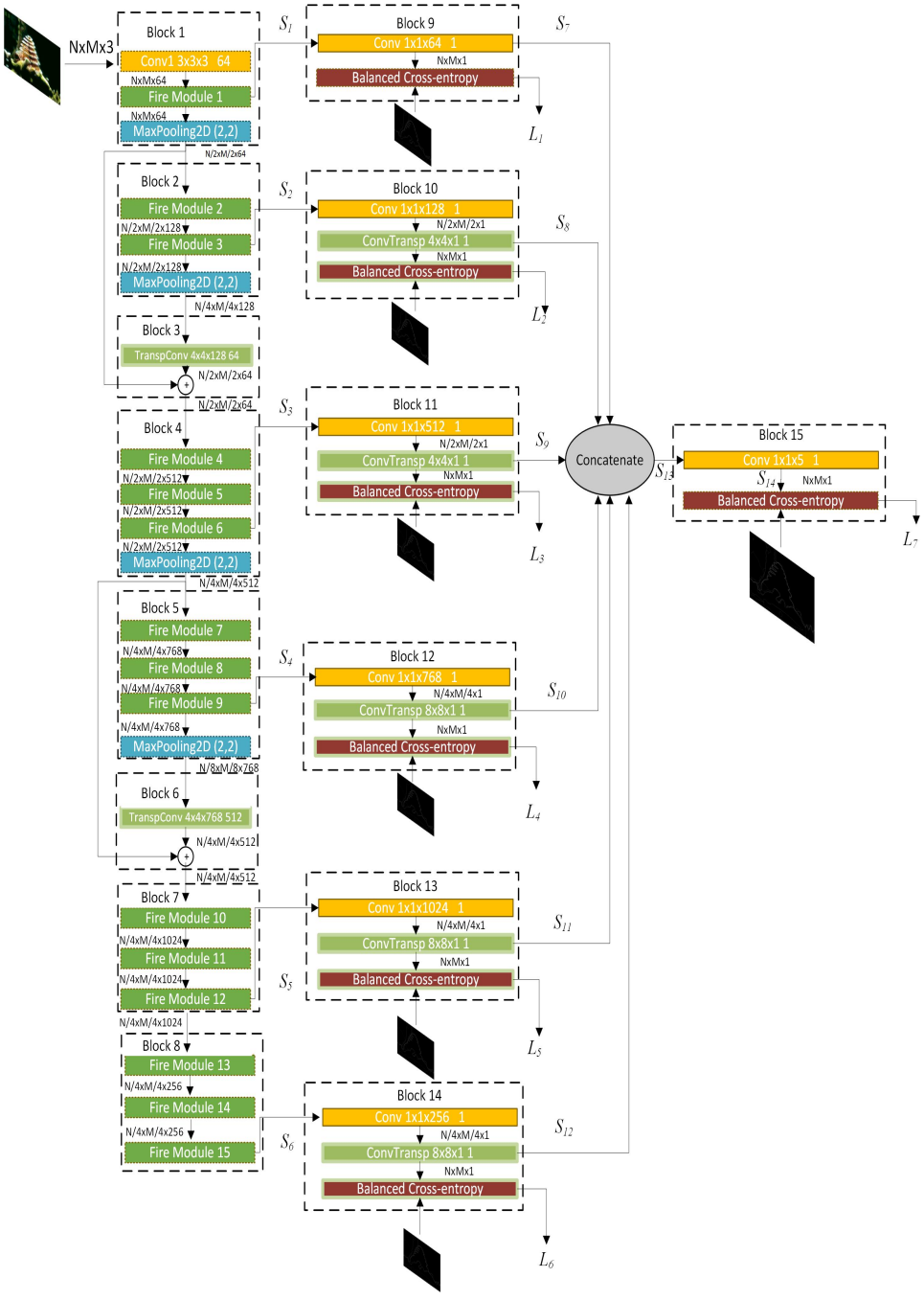

Figure 4.3: Architecture of the the modified network M-LRDNN.

whether these other networks are tested with a single scale or multi scales. It is especially noteworthy from this table that even though the number of network parameters used in M-LRDNN is almost two times that used in our baseline network, it is less than one-half of

Table 4.4: Comparison of the modified network M-LRDNN with state-of-art methods on the BSDS500 dataset when the networks are trained using a mix of the BSDS500 and PASCAL-Context datasets. The networks marked with † indicate that they are trained using a mixed of the two datasets.

Method	ODS F-measure	OIS F-measure	No. of parameters (in Millions)
BDCN-ResNet50†	0.826	0.840	28.7
LRDNN†	0.825	0.840	**4.3**
M-LRDNN†	**0.834**	**0.849**	8

that used by BDCN-VGG16 (the best performing VGG16-based architecture), and almost one-quarter of that of BDCN-ResNet50, the best performing architecture in the literature. It goes without saying that the performance of M-LRDNN can be expected to improve further if it is tested on images with multi scales. It is worth mentioning that the number of frames per second produced by the proposed techniques, LRDNN, and M-LRDNN, in the testing stage are 94, and 49, respectively.

Fig. 4.4 presents the precision-recall curves of the proposed methods, namely, the baseline LRDNN and M-LRDNN, and that of the other methods used in our comparison, using the BSDS500 dataset. It is seen from this figure that the M-LRDNN network provides the best performance in terms of the precision-recall curves as well.

4.3.4 Performance of LRDNN and M-LRDNN on the images of the NYUD dataset

In the previous two subsections, we evaluated the performance of our LRDNN and M-LRDNN networks using the images of the BSDS500 dataset. In this subsection, the performance of these two networks is carried out using the images of the NYUD dataset [64]. For this dataset, the value of the localization tolerance parameter is set to 0.011 compared to 0.0075 used for the BSDS500 dataset in view of the larger size images in the NYUD

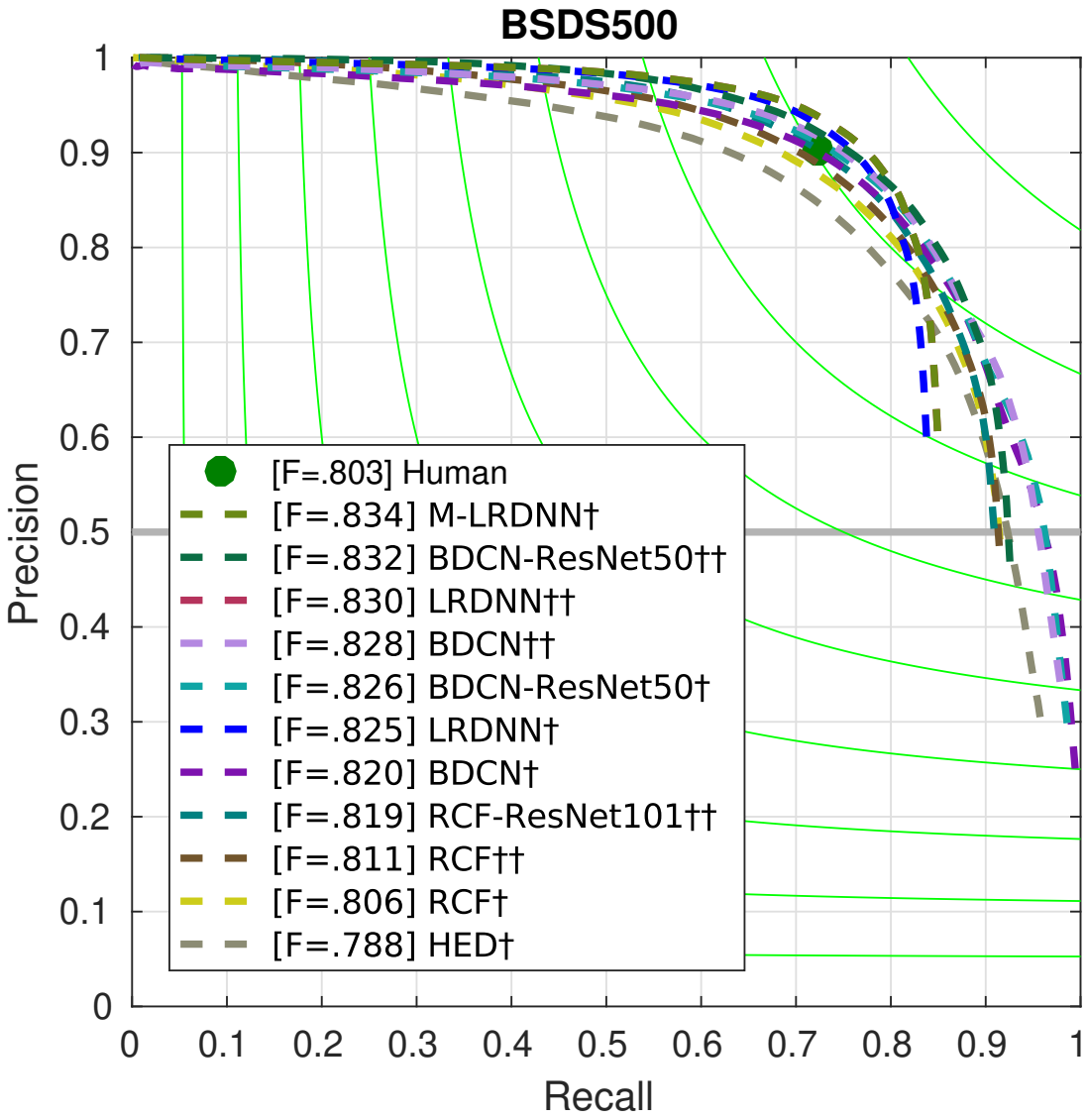

Figure 4.4: Precision/Recall curves of different methods on the images of the BSDS500 dataset.

dataset. It is noted that the value 0.011 for the tolerance parameter is the same as has been used by the other networks for this dataset. The network is trained separately on the RGB and depth training sets, resulting in two trained models referred to as RGB model and HHA model, respectively. The final predicted edge map is obtained by fusing the outputs from the two models.

The results in terms of the ODS F-measure, OIS F-measure and the number of parameters of the proposed LRDNN technique for the NYUD dataset are given in Table 4.5. For comparison, the results obtained by the HED-VGG16 [30], RCF-VGG16 [**?**], RCF-ResNet50 [34], RHN-VGG16 [61], BDCN-VGG16 [36], BDCN-ResNet50 [36], DexiNed-f [35], and BFENet-VGG16 [37] edge detection techniques are also listed in this table. It is seen from this table that, irrespective of the technique used, the values of both the metrics are enhanced, when the maps obtained from the RGB and HHA models are fused in contrast to those obtained using either the RGB or HHA model alone. It is also to be seen from this table if only one of the RGB and HHA models is to be used, it is the first model that provides superior results to that provided by the second one.

It is also seen from Table 4.5 that the values of the ODS and OIS F-measures provided by the proposed LRDNN (RGB + HHA) are, respectively, 0.012 and 0.008 larger than those provided by the RHN-VGG16 technique [61] and they are 0.011 and 0.010 larger than those provided by the recently published BFENet-VGG16 technique [37]. This improvement in the performance is achieved with only 4.3 M parameters utilized by our LRDNN (RGB + HHA) compared to the 11.5 M parameters employed by the RHN-VGG16 technique and greater than 20 M parameters provided by BFENet-VGG16 technique [37]. It is to be pointed out that, as in the case of the BSDS500 dataset, the performance of the proposed LRDNN technique is slightly inferior to that of the BDCN-ResNet50 technique, but the number of parameters employed by this latter technique is very much larger (28.7 M). However, it is to be seen from this table that the performance of the proposed M-LRDNN (RGB + HHA) in terms of the two metrics, as well as in terms of the number of parameters, is superior even to that of the BDCN-ResNet50 network.

Fig. 4.5 presents the precision-recall curves of the proposed methods, namely, the baseline LRDNN and M-LRDNN, and that of the other methods used in our comparison,

Table 4.5: Comparison of the proposed network with state-of-art methods on the NYUD dataset.

Method	ODS F-measure	OIS F-measure	No. of parameters (in Millions)
RCF-ResNet50 (RGB+HHA)	0.781	0.793	>>23
BDCN-ResNet50 (HHA)	0.717	0.730	28.7
BDCN-ResNet50 (RGB)	0.760	0.773	28.7
BDCN-ResNet50 (RGB+HHA)	0.788	0.802	28.7
HED-VGG16 (RGB+HHA)	0.741	0.757	14.7
DexiNed-f (RGB+HHA)	0.658	0.674	>>18
RCF-VGG16 (HHA)	0.703	0.717	14.8
RCF-VGG16 (RGB)	0.743	0.757	14.8
RCF-VGG16 (RGB+HHA)	0.765	0.780	14.8
BDCN-VGG16 (HHA)	0.708	0.720	16.3
BDCN-VGG16 (RGB)	0.748	0.763	16.3
BDCN-VGG16 (RGB+HHA)	0.767	0.783	16.3
BFENet-VGG16 (HHA)	0.719	0.732	>>20
BFENet-VGG16 (RGB)	0.752	0.766	>>20
BFENet-VGG16 (RGB+HHA)	0.773	0.787	>>20
RHN-VGG16 (HHA)	0.711	0.721	11.5
RHN-VGG16 (RGB)	0.751	0.762	11.5
RHN-VGG16 (RGB+HHA)	0.772	0.789	11.5
LRDNN (HHA)	0.721	0.734	**4.3**
LRDNN (RGB)	0.759	0.770	**4.3**
LRDNN (RGB+HHA)	0.784	0.797	**4.3**
M-LRDNN (HHA)	0.724	0.738	8
M-LRDNN (RGB)	0.767	0.779	8
M-LRDNN (RGB+HHA)	**0.792**	**0.805**	8

using the NYUD dataset. It is seen from this figure that the M-LRDNN network, as in the case of the BSDS500 dataset, provides the best performance in terms of the precision-recall curves.

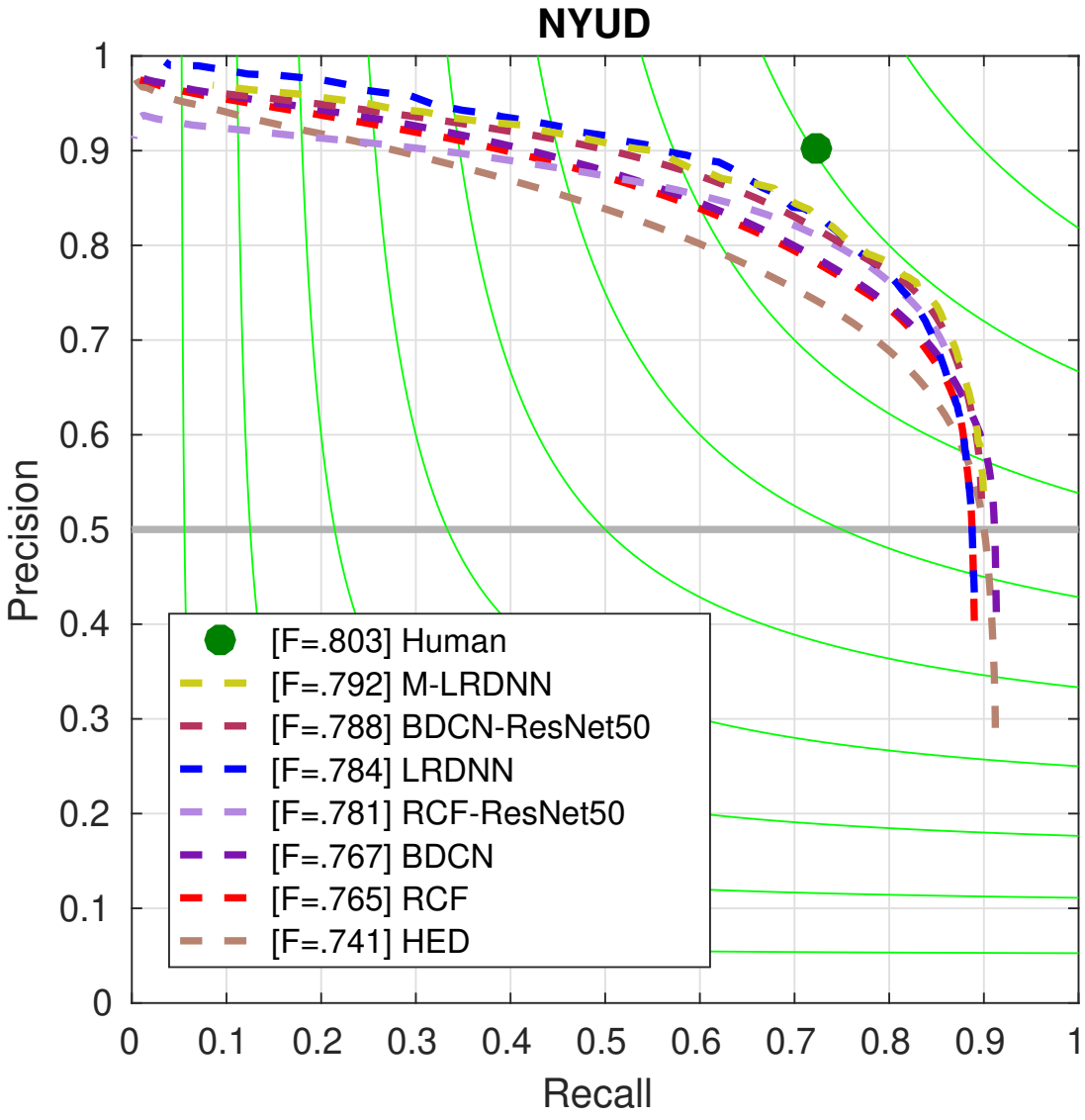

Figure 4.5: Precision/Recall curves of different methods on the images of the NYUD dataset.

4.3.5 Qualitative Performance and Comparison

In this section, we give some examples of the visual quality of the edge maps obtained from using the proposed LRDNN and M-LRDNN networks and compared with the visual quality of the edge maps obtained by using some of the other methods used in our comparison on the BSDS500 dataset.

Fig. 4.6 shows the edge maps of the images obtained by using the HED-VGG-16, CED-VGG16, RCF-VGG-16, RHN-VGG-16, BDCN-ResNet50, LRDNN, and M-LRDNN networks. In this figure, the first row shows five different images selected from the BSDS500 dataset, and the second row shows the corresponding ground truth maps. The remaining seven rows show, respectively, the maps yielded by HED-VGG-16, CED-VGG16, RCF-VGG-16, RHN-VGG-16, BDCN-ResNet50, LRDNN, and M-LRDNN after performing NMS. It is seen from this figure that the edge maps obtained from our two networks, LRDNN and M-LRDNN, are generally sharper and more complete compared to the maps obtained from the other techniques, with the maps of the M-LRDNN having a slight superiority over that obtained by using LRDNN.

4.3.6 An ultra-lightweight version of the LRDNN network and its performance on the images of the BSDS500 dataset

In this subsection, we propose an ultra lightweight version of the proposed LRDNN. In this lightweight version, we decrease the number of filters in blocks 2, 4, 5, and 7 of the baseline network. Specifically, the numbers of filters in the squeeze layers of blocks 2, 4, 5, and 7 are decreased to 8, 16, 16, and 16, respectively. Thus, the number of filters in the corresponding expand layers now need to be decreased to 64, 128, 128, and 128. In order to distinguish the lightweight network from the baseline network, we refer to the former as LW-LRDNN.

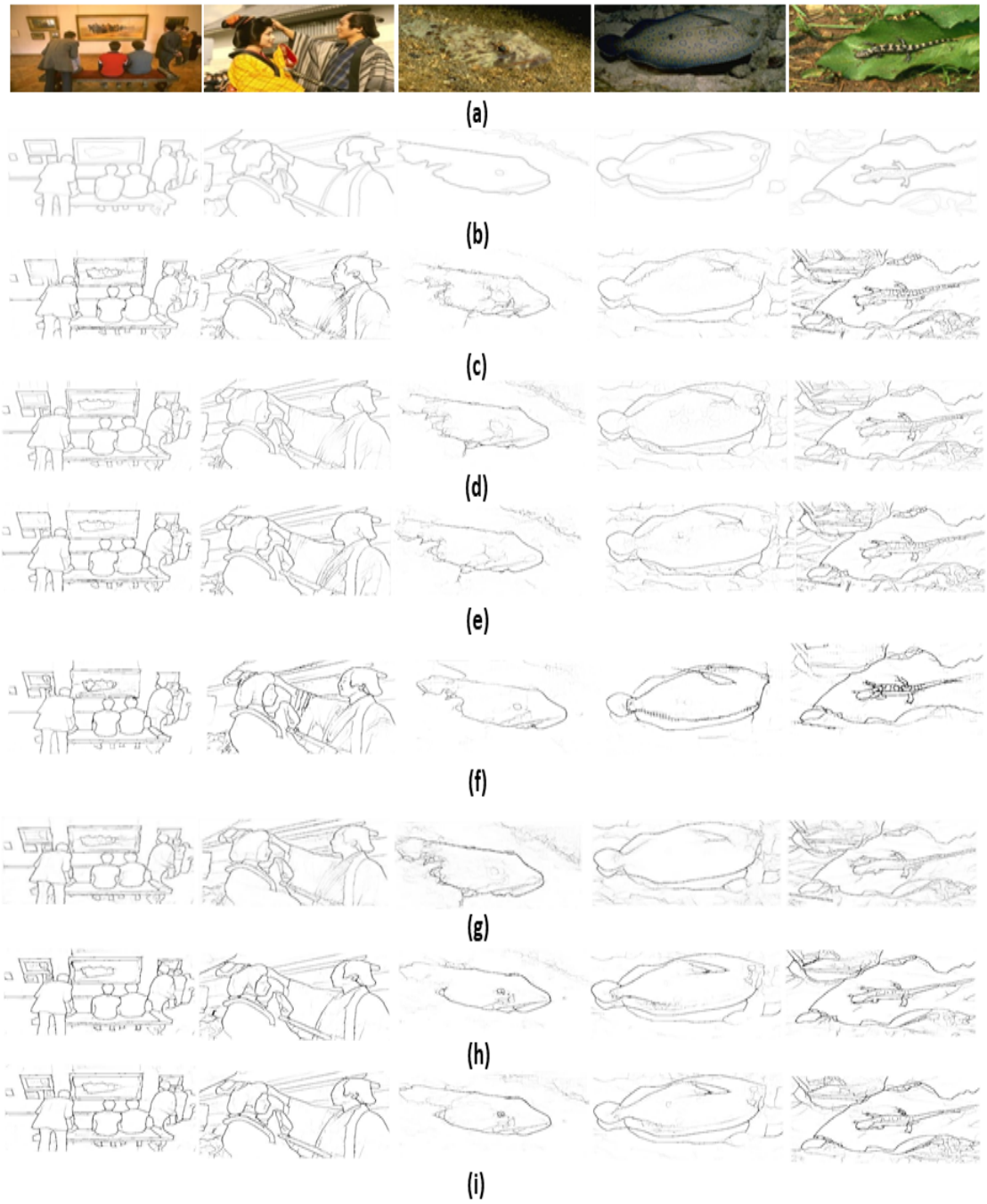

Figure 4.6: Visual quality of the edge maps obtained by seven different methods. (a) Original images from BSDS500 dataset. (b) Corresponding ground truth edge maps. (c) Edge maps obtained using HED-VGG16. (d) Edge maps obtained using CED-VGG16. (e) Edge maps obtained using RCF-VGG16. (f) Edge maps obtained using RHN-VGG16 (g) Edge maps obtained using BDCN-ResNet50. (h) Edge maps obtained using LRDNN. (i) Edge maps obtained using M-LRDNN.

Our LW-LRDNN network is trained using the mix of BSDS500 and PASCAL-Context datasets and tested using the 200 images of BSDS500. Table 4.6 gives the performance in terms of the ODS and OIS F-measures as well as the number of parameters of LW-LRDNN along with that of the recently published lightweight PiDiNet technique [43]. It is seen from this table that the performance of the proposed LW-LRDNN is about the same or slightly better than that of PiDiNet by using the number of parameters that is about 25 percent lower.

Table 4.6: Comparison of the proposed lightweight network with the lightweight PiDiNet network where the networks are trained using a mix of BSDS500 and PASCAL-Context datasets.

Method	ODS F-measure	OIS F-measure	No. of parameters (in Millions)
PiDiNet	0.807	**0.823**	0.71
LW-LRDNN	**0.808**	**0.823**	**0.53**

4.4 Summary

Existing edge detection deep networks based on the VGG-16 and ResNet architectures have extremely high complexity because the convolutional layers of such networks use large number of parameters. In this chapter, the idea of fire module along with residual learning has been used in a VGG16 architecture to propose a baseline edge detection network LRDNN. Since the complexity of this network is so much lower than that of all the other existing networks, we have been motivated to improve its performance by proposing a somewhat higher complexity version of LRDNN, M-LRDNN. In order to show the effectiveness of the idea used for designing LRDNN, we have also proposed in this chapter its ultra-lightweight version, LW-LRDNN. All the three proposed networks have been extensively experimented on two publicly available datasets. It has been shown that the proposed

LRDNN network has a complexity that is not only lower than that of ResNet-based networks, but also substantially lower than those of all the existing VGG16-based networks and it has a performance, which surpasses that of all the VGG-16 edge detection networks and all the other networks except that it is slightly lower in terms of ODS F-measure than that of BDCN-ResNet50. The second proposed edge detection network, M-LRDNN, provides a performance that is superior to all the existing edge detection networks in the literature. Even though the complexity of M-LRDNN is almost twice that of the baseline architecture, LRDNN, this complexity is still only a quarter of that of BDCN-ResNet50, the best performing edge detection network in the literature. The third proposed edge detection network, LW-LRDNN, is an ultra-lightweight version of the baseline LRDNN network and has been shown to outperform the only other existing ultra-lightweight edge detection network, PiDiNet [43], in the literature both in terms of the edge detection accuracy and complexity.

Chapter 5

Conclusion and Future Work

5.1 Concluding Remarks

Existing deep residual learning based techniques for edge detection provide good performance but at the expense of an extremely high computational complexity. The reason for this large complexity in these networks is the use of very large number of convolutional layers, which cannot be reduced without degrading their performance. On the other hand, the complexity of the VGG16-based edge detection techniques is relatively much lower than that of the residual learning based techniques but with reduced performance. This reduced performance is due to the fact that the spatial resolution of the feature maps in the last block is very small. Restoring the spatial resolution of the feature maps in the last block to the original image size is necessary for the edge detection task. However, such a sudden increase in spatial resolution of the maps results in a blurred output and a poor localization of the edges. Further, the deeper layers of the VGG16-based techniques are not able to learn some of the information captured by the initial layers. In this thesis, new VGG16-based DCNN techniques for edge detection with the capability of deep supervision and residual learning have been developed.

In the first part of the thesis, a mechanism of residual learning has been introduced in the

VGG16 architecture to obtain a deep edge detection convolutional neural network to provide superior performance while still preserving its lower complexity character. It has been shown that despite the larger spatial resolution of the feature maps produced by the convolutional layers, necessitated to introduce the residual learning idea, the overall complexity of the network is not increased. This has been achieved through a judicious choice of the number of filters and the kernel size used in each of the convolutional layers of the network. Extensive experiments have been carried out on the proposed edge detection network using several datasets, namely, BSDS500, NYUD, and Multicue, to evaluate its performance and complexity. It has been shown that the proposed network outperforms all the existing techniques except for BDCN-VGG16 and BDCN-ResNet50 in terms of ODS F-measure and OIS F-measure. At the same time, the complexity of the proposed network in terms of the number of parameters is lower than that of all the other existing edge detection techniques, and specifically, it is 4.8 M parameters lower than that of the BDCN-VGG16 network.

Even though the complexity of the edge detection network, RHN, proposed in the first part of the thesis is lower than that all of the other networks, it is still high. Further its performance is limited by its capability in extracting rich features. Therefore, in the second part of the thesis, VGG16-based edge detection networks that requires extremely low number of parameters and yet have edge detection performance that is superior to that of RHN network proposed in the first part have been proposed and that of other edge detection networks. This has been made possible by using fire modules in the convolutional layers of the VGG-16 architecture. Through the use of fire modules, the number of parameters of a VGG-16 based edge detection network is made so much lower as to make it possible to maintain or even to improve the performance of the resulting networks by make them deeper, while at the same time, drastically reducing the computational complexity. Specifically, three edge detection networks, namely, LRDNN, M-LRDNN, and LW-LRDNN. The M-LRDNN network has been built upon the architecture of LRDNN by increasing the

number of layers and filters in the convolutional layers of the latter. The network LW-LRDNN is an ultra-lightweight version of the baseline model LRDNN and is obtained by decreasing the number of filters in the latter network. Extensive experiments have been performed on all the three networks proposed in the second part. The proposed LRDNN is shown to outperform all of the edge detection networks, except the BDCN-ResNet50 network, in terms of edge detection accuracy with a complexity that is substantially smaller than that of any edge detection network in the literature. The M-LRDNN is shown to provide an edge detection performance that is superior to that of all edge detection networks. Even though the complexity in terms of number of parameters of M-LRDNN is twice that of the baseline LRDNN network, it is only about one-quarter of that of BDCN-ResNet50, the best performing edge detection network in the existing literature and one-half of that of BDCN-VGG16. Finally, the proposed LW-LRDNN has been shown to maintain the detection performance of the only ultra-lightweight edge detection network, PiDiNet, existing in the literature by using a number of parameters that is about 25 percent lower than that of the latter.

5.2 Scope for Future Investigation

The following are a couple of examples for further investigating the edge detection networks proposed in this thesis.

(i) It has been shown in Chapters 3 and 4 that the use of the images from the NYUD dataset, which includes images with a representation in RGB as well as that describing the depth of the pixels, in training the proposed networks has enhanced their performance. Therefore, it would be worth exploring the performance of the proposed edge detection networks trained using images with multiple representations.

(ii) An architectural unit called, squeeze-and-excitation (SE) is a network unit that can be used to perform feature recalibration to selectively emphasize useful features and suppress the less useful features and thus can make the network to learn more efficiently the useful features. The use of SE has been shown [43] to enhance the performance of classification networks. A study could be undertaken to investigate the performance of the networks proposed in this thesis by adding SE units in their blocks.

References

[1] L. Coetzee and E.C. Botha, "Fingerprint recognition in low quality images," *Pattern recognition*, vol. 26, no. 10, pp. 1441–1460, 1993.

[2] Y. Xu, G. Lu, Y. Lu, and D. Zhang, "High resolution fingerprint recognition using pore and edge descriptors," *Pattern Recognition Letters*, vol. 125, pp. 773–779, 2019.

[3] W. Cui, G. Wu, R. Hua, and H. Yang, "The research of edge detection algorithm for fingerprint images," in *IEEE World automation congress*, pp. 1–5, 2008.

[4] I. Aljarrah, A. Al-Amareen, A. Idries, and O. Al-Khaleel, "Image mosaicing using binary edge detection," in *The International Conference on Computing Technology and Information Management (ICCTIM)*, p. 186, Society of Digital Information and Wireless Communication, 2014.

[5] A. Alamareen, O. Al-Jarrah, and I. Aljarrah, "Image mosaicing using binary edge detection algorithm in a cloud-computing environment," *International Journal of Information Technology and Web Engineering (IJITWE)*, vol. 11, no. 3, pp. 1–14, 2016.

[6] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[8] F. Gao, Y. Li, and S. Lu, "Extracting moving objects more accurately: a cda contour optimizer," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[9] A. Manno-Kovacs, "Direction selective contour detection for salient objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 375–389, 2018.

[10] Z. Tu, Y. Ma, C. Li, J. Tang, and B. Luo, "Edge-guided non-local fully convolutional network for salient object detection," *IEEE Transactions on circuits and systems for video technology*, vol. 31, no. 2, pp. 582–593, 2020.

[11] L. Sun and T. Shibata, "Unsupervised object extraction by contour delineation and texture discrimination based on oriented edge features," *IEEE transactions on circuits and systems for video technology*, vol. 24, no. 5, pp. 780–788, 2013.

[12] L.-W. Tsai, J.-W. Hsieh, and K.-C. Fan, "Vehicle detection using normalized color and edge map," *IEEE transactions on Image Processing*, vol. 16, no. 3, pp. 850–864, 2007.

[13] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 10–20, 2018.

[14] A. Al-Amaren, M.O. Ahmad, and M.N.S. Swamy, "A very fast edge map-based algorithm for accurate motion estimation," *Signal, Image and Video Processing*, pp. 1–8, 2021.

[15] W.-C. Lin and J.-W. Wang, "Edge detection in medical images with quasi high-pass filter based on local statistics," *Biomedical Signal Processing and Control*, vol. 39, pp. 294–302, 2018.

[16] B. Ramamurthy and K. Chandran, "Content based image retrieval for medical images using canny edge detection algorithm," *International Journal of Computer Applications*, vol. 17, no. 6, pp. 32–37, 2011.

[17] F. Orujov, R. Maskeliūnas, R. Damaševičius, and W. Wei, "Fuzzy based image edge detection algorithm for blood vessel detection in retinal images," *Applied Soft Computing*, vol. 94, p. 106452, 2020.

[18] F.G. Sobel, "A 3x3 isotropic gradient operator for image processing," presented at the Stanford Artificial Intelligence Project, 1968.

[19] J. MS, "Prewitt."object enhancement and extraction"," *Picture Processing and Psychopictorics*, vol. 10, pp. 15–19, 1970.

[20] L. Roberts, "Machine perception of 3-d solids-series. optical and electro-optical information processing, mit press, usa," 1965.

[21] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[22] A. Al-Amaren, M.O. Ahmad, and M.N.S. Swamy, "Edge map extraction of an image based on the gradient of its binary versions," in *64th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 279–283, 2021.

[23] R.M. Haralick, "Digital step edges from zero crossing of second directional derivatives," in *Readings in Computer Vision*, pp. 216–226, Elsevier, 1987.

[24] A. Huertas and G. Medioni, "Detection of intensity changes with subpixel accuracy using laplacian-gaussian masks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 651–664, 1986.

[25] Y. Ganin and V. Lempitsky, "N4-fields: Neural network nearest neighbor fields for image transforms," in *Asian Conference on Computer Vision*, pp. 536–551, Springer, 2014.

[26] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4380–4389, 2015.

[27] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3982–3991, 2015.

[28] J.-J. Hwang and T.-L. Liu, "Pixel-wise deep learning for contour detection," *ArXiv Preprint ArXiv:1504.01989*, 2015.

[29] G. Bertasius, J. Shi, and L. Torresani, "High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 504–512, 2015.

[30] S. Xie and Z. Tu, "Holistically-nested edge detection," *International Journal of Computer Vision*, vol. 125, no. 1-3, pp. 3–18, 2017.

[31] Y. Liu and M.S. Lew, "Learning relaxed deep supervision for better edge detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 231–240, 2016.

[32] I. Kokkinos, "Pushing the boundaries of boundary detection using deep learning," *ArXiv Preprint ArXiv:1511.07386*, 2015.

[33] Y. Wang, X. Zhao, Y. Li, and K. Huang, "Deep crisp boundaries: From boundaries to higher-level tasks," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1285–1298, 2018.

[34] Y. Liu, M. Cheng, X. Hu, J. Bian, L. Zhang, X. Bai, and J. Tang, "Richer convolutional features for edge detection.," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, p. 1939, 2019.

[35] X.S. Poma, E. Riba, and A. Sappa, "Dense extreme inception network: Towards a robust cnn model for edge detection," in *Proceedings of the IEEE/CVF Winter Conference on Applicatio ns of Computer Vision*, pp. 1923–1932, 2020.

[36] J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang, "BDCN: Bi-directional cascade network for perceptual edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[37] C. Lin, Z. Zhang, and Y. Hu, "Bio-inspired feature enhancement network for edge detection," *Applied Intelligence*, pp. 1–16, 2022.

[38] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[39] L. Cayton, "Fast nearest neighbor retrieval for bregman divergences," in *Proceedings of the 25th international conference on Machine learning*, pp. 112–119, 2008.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[41] P. Dollár and C.L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, 2014.

[42] Y. Hu, N. Belkhir, J. Angulo, A. Yao, and G. Franchi, "Learning deep morphological networks with neural architecture search," *arXiv preprint arXiv:2106.07714*, 2021.

[43] Z. Su, W. Liu, Z. Yu, D. Hu, Q. Liao, Q. Tian, M. Pietikäinen, and L. Liu, "Pixel difference networks for efficient edge detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5117–5127, 2021.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] M.D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *ArXiv Preprint ArXiv:1311.2901*, 2013.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *ArXiv Preprint ArXiv:1409.4842*, 2014.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[48] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[49] S.K. Pandey and R.R. Janghel, "Recent deep learning techniques, challenges and its applications for medical healthcare system: a review," *Neural Processing Letters*, vol. 50, no. 2, pp. 1907–1935, 2019.

[50] C.-H. Chou, Y.-S. Su, C.-J. Hsu, K.-C. Lee, and P.-H. Han, "Design of desktop audiovisual entertainment system with deep learning and haptic sensations," *Symmetry*, vol. 12, no. 10, p. 1718, 2020.

[51] H.K. Hamarashid, S.A. Saeed, and T.A. Rashid, "A comprehensive review and evaluation on text predictive and entertainment systems," *Soft Computing*, pp. 1–22, 2022.

[52] M. Ali, N. Yousuf, M. Rahman, J. Chaki, N. Dey, K. Santosh, *et al.*, "Machine translation using deep learning for universal networking language based on their structure," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 8, pp. 2365–2376, 2021.

[53] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin, "Natural language to visualization by neural machine translation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 217–226, 2021.

[54] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[55] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[56] C.-C. Hsu and C.-W. Lin, "Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 421–429, 2017.

[57] M. Zhang, C. Tseng, and G. Kreiman, "Putting visual object recognition in context," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12985–12994, 2020.

[58] D. Lin, Y. Li, T.L. Nwe, S. Dong, and Z.M. Oo, "Refineu-net: Improved u-net with progressive global feedbacks and residual attention guided local refinement for medical image segmentation," *Pattern Recognition Letters*, vol. 138, pp. 267–275, 2020.

[59] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep residual learning for image compression.," in *CVPR Workshops*, p. 0, 2019.

[60] S. Zhou, Y. Zhao, S. Xu, B. Xu, *et al.*, "Multilingual recurrent neural networks with residual learning for low-resource speech recognition.," in *INTERSPEECH*, pp. 704–708, 2017.

[61] A. Al-Amaren, M.O. Ahmad, and M.N.S. Swamy, "RHN: A residual holistic neural network for edge detection," *IEEE Access*, vol. 9, pp. 74646–74658, 2021.

[62] P. Sharma, M. Diwakar, and N. Lal, "Edge detection using moore neighborhood," *International Journal of Computer Applications*, vol. 61, no. 3, 2013.

[63] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[64] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision*, pp. 746–760, Springer, 2012.

[65] D. A. Mély, J. Kim, M. McGill, Y. Guo, and T. Serre, "A systematic comparison between visual cues for boundary detection," *Vision Research*, vol. 120, pp. 93–107, 2016.

[66] D.R. Martin, C.C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[67] P. Isola, D. Zoran, D. Krishnan, and E.H. Adelson, "Crisp boundary detection using pointwise mutual information," in *European conference on computer vision*, pp. 799–814, Springer, 2014.

[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[69] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes," in *International Conference on Machine Learning*, pp. 71–79, 2013.

[70] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ArXiv Preprint ArXiv:1502.03167*, 2015.

[71] C. Jaron, "Glossary of deep learning: Batch normalisation, accessed: May. 10, 2021. [online]. available: https://medium.com/deeper-learning/glossary-of-deep-learning-batch-normalisation-8266dcd2fa82," 2017.

[72] R. Shashank, "A guide to an efficient way to build neural network architectures-part ii: Hyper-parameter selection and tuning for convolutional neural networks using hyperas on fashion-mnist, accessed: May. 10, 2021. [online]. available: https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7," 2017.

[73] F. Chollet *et al.*, "keras, github, accessed: May. 10, 2021. [online]. available: https://keras.io," 2015.

[74] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems. 2015. software available from tensorflow. org," *URL https://www. tensorflow. org*, 2019.

[75] P. Dollár and C.L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, 2014.

[76] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 891–898, 2014.

[77] D.R. Martin, C.C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[78] P. Dollár and C.L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, 2014.

[79] S. Hallman and C.C. Fowlkes, "Oriented edge forests for boundary detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1732–1740, 2015.

[80] J.-J. Hwang and T.-L. Liu, "Pixel-wise deep learning for contour detection," *ArXiv Preprint ArXiv:1504.01989*, 2015.

[81] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 564–571, 2013.

[82] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision*, pp. 345–360, Springer, 2014.

[83] J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning," 2018.

[84] A. Al-Amaren, M.O. Ahmad, and M.N.S. Swamy, "A low-complexity residual deep neural network for image edge detection," to appear in *Applied Intelligence*, 2022.

[85] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[86] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," in *8th IEEE Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 169–175, 2018.