No-Reference Image Quality Assessment using Deep Convolutional Neural Networks

Master's Thesis

Amritpal Singh Gill

No-Reference Image Quality Assessment using Deep Convolutional Neural Networks

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

Embedded Systems

by

Amritpal Singh Gill

Multimedia Computing Group
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://mmc.tudelft.nl

No-Reference Image Quality Assessment using Deep Convolutional Neural Networks

Author: Amritpal Singh Gill
Student id: 4419820
Email: amritpal.singh@live.com

Abstract

No-reference image quality assessment (NR-IQA) is a challenging field of research that, without making use of reference images, aims at predicting the image quality as it is perceived by the human visual system (HVS). Many NR-IQA methods have been proposed over time but recently proposed convolutional neural network (CNN) based approaches, through their powerful feature learning capabilities, have outperformed all previously existing NR-IQA methods. But these CNN based approaches are perceptually incorrect in assuming distortions to be homogeneously distributed across images. They operate on very small image portions while considering all of them to have identical perceptual quality, whereas in reality, different parts of an image, based on their structure and content, could bear different perceptual quality. Further, these approaches utilize shallow CNN architectures which render them incapable of taking advantages offered by the deep CNN architectures.

To improve upon the limitations of existing CNN based approaches, we conducted a design space exploration of CNN's and proposed a suitable CNN design for NR-IQA task, that operates on bigger image portions and employs a deeper architecture. The proposed design achieves the state of the art performance on LIVE and TID datasets. We further provide informative visualization of features learned by the proposed CNN design, which shed light on its internal working while promoting further understanding regarding the nature of image quality.

Thesis Committee:

Chair:	Prof. Dr. Alan Hanjalic, Faculty EEMCS, TUDelft
University supervisor:	Dr. Judith Redi, Faculty EEMCS, TUDelft
Committee Member:	Dr. Jan van Gemert, Faculty EEMCS, TUDelft

Preface

The end of this thesis project marks the end of my wonderful journey as a student at TU Delft. This project was the most challenging, yet most exciting part of the journey. I remember just one year ago I was struggling to define my interests and was not certain about what I want to do in my thesis project. Then like a beacon of light, my supervisor Judith Redi proposed this wonderful topic to me, towards which I was able to work with my full mind and soul. In the part of my journey during this thesis project, I made numerous mistakes from point to point, which resulted in many setbacks and disappointments. But working with neural networks, I understood one thing that learning is not possible without mistakes. Being an effective teacher, my mistakes also taught me many things, and I was able to learn a lot from them. Thus standing on the verge of ending my thesis and looking back at the journey, no matter how many setbacks and disappointments I faced, they look tiny in the light of joy I am feeling right now.

I would like to express my sincere gratitude towards Judith Redi for being a wonderful supervisor and guiding me through this project. I would also like to thank Robbert Eggermont for his help and guidance in getting me familiar with the INSY cluster. I would further like to thank Alan Hanjalic and Jan van Gemert for being part of my thesis committee. Next, I would like to thank all my friends, colleagues, and relatives for the help, support, and motivation they provided from time to time. A special thank to Ahluwalia family for always being there at the time of need and for the moral support they provided. I would furthermore like to thank my parents, grandparents and my brother for always believing in my capabilities and helping me realize them. And finally, I am thankful to God for all the blessings and fortune I enjoy.

Amritpal Singh Gill
Delft, the Netherlands
June 30, 2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With an increase in the popularity of smartphones, compact cameras, and Internet services like Facebook and Instagram, past few years have seen tremendous growth in the production and sharing of digital images. The journey of a digital image starts with it being acquired by a digital camera, which converts it into a digital format and compresses it using lossy compression algorithms to meet the onboard storage availability. This image is then transmitted over wired or wireless transmission channels and is altered in its resolution to meet the available bandwidth. Finally, the end user receives this image and watches it over devices ranging from smartphones to 4K displays, which require further alterations to its resolution. As shown in Figure 1.1, all these stages generate visible artifacts in images that decrease the quality of experience among the end users who expect images to be of highest quality. The end users tend to

(a) Reference Image (b) Acquisition (c) Compression

(d) Transmission (e) Display rendering

Figure 1.1: Resulting distortions in undistorted image during (b) acquisition, (c) compression, (d) transmission and (e) display rendering.

become more inclined towards the selection of a content provider, a service provider, and a display device that could better satisfy their expectations of image quality at delivery. Thus it becomes crucial for all content providers, service providers, and display providers to optimize these respective technologies towards the provision of perceptually good results, and to do so, perceptual image quality needs to be estimated. Furthermore, this estimation process should be automated, as much as possible, to make it independent from the availability of human observers in order to determine the perceptual quality.

Image quality is defined as the characteristic of an image that measures its level of degradation as perceived by the human visual system (HVS) [64]. Since modeling HVS and its sensitivity to degradation is extremely difficult, it is very tough to perform image quality assessment (IQA) accurately. Past few decades have witnessed tremendous research in the field of IQA and many methods, ranging from subjective IQA (use human observers for IQA) to Objective IQA (use mathematical models for IQA), have been proposed.

No-reference image quality assessment (NR-IQA), a branch of objective IQA, estimates image quality without utilizing any information on reference images (the undistorted versions of the image). Since NR-IQA is not dependent upon the availability of reference images, it is more flexible than other objective IQA methods that are dependent upon either full reference images (Full-reference IQA) or partial information about them (Reduced-reference IQA). Furthermore, it has a wider scope of applications including those where reference images are usually not available. For instance, in adaptive video streaming where resources (like bandwidth) need to be optimized for delivering maximum possible quality, or in restoration algorithms (like denoising or sharpening) that are applied to improve quality at delivery. Thus NR-IQA methods are more desirable, but the unavailability of reference images render them incapable of reasoning in terms of image fidelity (inferred by the ability to discriminate between two images [52]), that make them even more challenging to achieve.

Many NR-IQA based algorithms have been proposed over time. One class of these algorithms including DIIVINE[37], BLIINDS-II[49], and BRISQUE[35] use handcrafted features (attributes (edge, color, etc.) in data (images) that are relevant to the modeling problem [3]) that supposedly captures relevant factors affecting image quality. Although their performance is acceptable, there is still large room for improvement regarding the accuracy with which they reproduce human judgment of quality. Another set of algorithms, including CORNIA[66][67] and convolutional neural network (CNN) based approaches [25][26], employ automatic learning of features from the raw image pixels, which are superior and more efficient as they make feature selection automatic and embedded within the system itself.

CNN is a multilayered feedforward neural network inspired from HVS. Given an image, the strength of CNN lies in its ability to extract multiple layers of features, which has been shown to mimic different stages of HVS [29]. The features of higher layers are developed upon the features of the lower layers, and thus they become more complex with the increase in CNN depth [11]. It has been shown in the networks trained to recognize object categories in images, that the lower layers of CNN learn

to extract simple features related to color and contrast whereas the higher layers are specialized into learning features of higher complexity, such as semantics. This ability of CNN's to learn high-level features could be utilized in developing better attributes in the context of image quality, which could simplify and thus improve the task of image quality assessment.

Though existing CNN based approaches [25][26] have reported superior performance over other NR-IQA methods, there is still a lot of room for improvement. To start with, since the fixed CNN architecture restricts input images to be of fixed size, the mentioned CNN based approaches use very small portions of input images (usually 7 x 7 or 32 x 32), called image patches, for feature learning and thus are trained to predict patch quality. During training, all patches belonging to the same image are assigned the quality label of the whole image. Since SSIM [60] has shown that image quality also depends on the structural information present in an image, considering all patches to have the same quality as their corresponding image is perceptually incorrect. It is because, depending upon the structure of an image and the content of its patches, distortion may vary among different patches within the same image, resulting in distinct perceptual quality among them. Further, since image quality also depends on the structural information of an image, global information is vital for the accurate prediction of image quality. Thus we believe that bigger input patches should be used instead to take more global information into consideration, which also bring the quality of employed patches closer to the actual quality of the whole image.

Furthermore, the existing CNN based approaches employ a small number of convolutional layers (usually one or three layers), which prevents the network from learning high-level features from input images. Thus the full potential of CNN has not yet been exploited for the application of NR-IQA.

Based on all these observations, in this research project, we further explore CNN design with an aim to improve upon the current limitations of automatic feature learning based approaches. Our main goal is to develop a CNN for the application of NR-IQA that (1.) employs bigger input patches and (2.) has a deeper architecture with more convolutional layers.

Since the inner working of CNN's is typically difficult to understand because of their big architectures, we also aim at exploring their inner working, when trained for the application of NR-IQA, by visualizing and understanding the nature of their learned features. This could further help in better understanding the requirements of efficient IQA metrics and thus could be useful in designing improved algorithms through the fulfillment of these requirements.

1.1 Research Objectives

As discussed before, existing CNN based NR-IQA approaches ([25] and [26]) suffer from many limitations. With an aim to overcome these limitations and to visualize features learned by the CNN trained for the application of NR-IQA, this thesis project addresses two main research questions.

Research Question 1

What is a suitable design for a convolutional neural network to predict perceived image quality in a no-reference setting?

We define four objectives to answer this research question:

Objective 1 *Explore the field of image quality assessment and deep learning.*

The goal here is to first understand the nature of the problem and review the amount of work that has already been done, so as to choose an appropriate methodology. A literature study on IQA methods gives the performance benchmark that guides our decision making. A literature study on deep learning elaborates upon the set of available tools that can be utilized for achieving the goals of this project. This objective is linked to the literature study presented in chapter 2.

Objective 2 *Define research methodology.*

The goal here is to define an approach, based on the outcomes of objective 1, which is suitable for answering the first research question. This objective is linked to chapter 3 in which we describe our general research methodology.

Objective 3 *Propose a convolutional neural network design.*

The goal here is to propose a CNN design for the application of NR-IQA, through design space exploration (that includes parameters of the training process and CNN architecture) of CNN. This research objective is linked to chapter 4.

Objective 4 *Evaluate the proposed convolutional neural network design*

The goal here is to conduct an extensive performance evaluation of the CNN design that obtained as an outcome of research objective 3. This research objective is linked to chapter 5 in which we report the results of the undertaken evaluation.

Research Question 2

What can we learn from the visualization of features developed by a convolutional neural network trained for image quality assessment?

We define two objectives to answer this research question:

Objective 5 *Define a methodology.*

The goal here is to define a methodology for generating and inspecting feature visualizations of proposed CNN design (the outcome of objective 3). This objective is linked to chapter 3 in which we describe the research methodology.

Objective 6 *Report the observations derived from the analysis of feature visualizations.*

The goal here is first to generate the feature visualizations of CNN, and then to analyze them in light of the perceptual quality assessment task. This objective is linked to chapter 6 in which we present the generated features and report our observations.

1.2 Contributions

This thesis project provides the following contributions:

1. **Convolutional Neural Network Design**

   We propose a state of the art CNN design suitable for the application of NR-IQA. To best of our knowledge, the proposed CNN architecture is the deepest and perceptually the most accurate (in terms that it employ bigger input patches that bring patch quality closer to the overall image quality) among existing NR-IQA based CNN architectures. Further, it delivers much superior performance in comparison to existing state of the art NR-IQA metrics.

2. **Feature visualizations**

   We provide informative visualizations of features learned by the state of the art CNN architecture. These visualizations could further help in better understanding the characteristics of image quality, which could help in improving IQA metrics.

1.3 Thesis Outline

This thesis is organized into seven chapters. After this introduction, we present the literature review in Chapter 2 in which we introduction the fields of image quality assessment and deep learning. After introducing these two main fields that are the basis of this research project, we present our research methodology in Chapter 3 that was used for answering both of our research questions. Then we move on to the main chapter of this report (Chapter 4) that presents the design space exploration of CNN and propose a final CNN design for the application of NR-IQA. After fixing the design of CNN, we present the results of its performance evaluation in Chapter 5. Chapter 3 and 5 answers our first research question. Then in Chapter 6, we answer our second research question by generating and analyzing the visualizations of features learned by the proposed CNN design. Finally, in Chapter 7, we conclude our work and give recommendations for the future work.

Chapter 2

Background and Related Work

This chapter presents the literature study over which this research project is based on. Concepts discussed in this chapter are used in proceeding chapters of this report.

In section 2.1, we present background information to the field of image quality assessment, in which we describe different methods and algorithms used for IQA. After this in section 2.2, we introduce the field of convolutional neural networks and present some of its important components. Finally in section 2.3 we conclude this chapter by summarizing its main findings.

2.1 Image Quality Assessment

As described in the introduction, image quality assessment (IQA) is vital for many applications, but is also tough to achieve as such because of its dependency on the quantification of user perception. Figure 2.1 shows how the presence of different distortion types could affect the perceived quality of an image. Till date, the most reliable way to undertake IQA is through subjective assessments (utilizing human observers), but this is not practical in real-life applications because users can't always be dependent upon to comment on the perceived quality. On the contrary, objective image quality assessment instead focuses on implementing models of human perception that can estimate the quality of an image as perceived by a user based solely on pixel analysis information.

In the following, we briefly revise existing subjective quality assessment methods to then go deeper in the state of the art of methods for objective quality assessment.

2.1.1 Subjective image quality assessment

Subjective image quality assessment methods employ human observers to give their opinion on the quality of images to be assessed. Since humans are the end users in most of the multimedia applications, subjective IQA methods are the most accurate and reliable for image quality assessment [36].

Several international standards like ITU BT 500[42], ITU P910[22] and ITU P913[23] have been proposed for performing subjective image quality assessment. For given set of images, the main goal of subjective IQA methods is to assign, to each of them, a score that quantifies the user's perceived quality. In most cases, it is achieved with a scaling process, which can either be explicit or implicit [44].

(a) Reference image (b) Additive Gaussian noise (c) Fast fadding

(d) Gaussian blur (e) JPEG compression (f) JPEG2000 compression

Figure 2.1: Examples of distortion types.

Subjective testing typically focuses on quantifying quality as perceived by an average observer. A group of subjects is asked to evaluate an image and provide their perceived quality score. These scores are then accumulated, and the final score is computed that reflects quality as perceived by an average person observing it. Different scales could be utilized for the computation of this final score, for example, direct scaling in which the perceived quality of an image is calculated as the mean of the scores that each subject assigned to that image (Mean Opinion Score (MOS)). The objective IQA methods (to be followed) aim at predicting these mean values using different models.

Despite being most accurate and reliable, subjective IQA methods are very impractical for real-world applications because it is very expensive and time-consuming to gather an adequate number of observers to assess the quality of images. Hence more practical objective IQA methods are used instead for many applications.

2.1.2 Objective image quality assessment

Objective IQA methods, instead of using human observers, aim at employing relevant models that are capable of predicting the visual quality of images as perceived by humans. Since these algorithms do not require any human observer, they are fast and very practical for many real world applications, like image enhancement, image restoration, etc.

To estimate the perceptual quality of the given image (called test image), either in the presence or absence of its reference image, most of the objective IQA methods share a common framework of three main phases as illustrated in Figure 2.2. These three phases are described in the following.

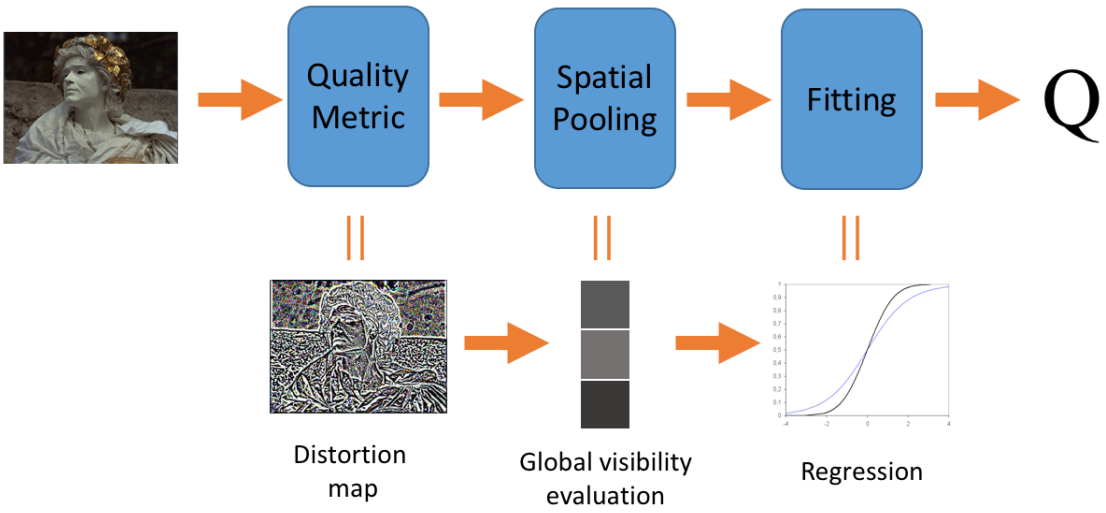


Figure 2.2: General framework of objective image quality assessment metrics.

1. Corresponding to the employed objective IQA method, the test image is processed pixel by pixel or region by region to measure the amount of distortion present in it. This phase then outputs the measured distortion in the form of a distortion map that contains the local description of image quality. This step is equivalent to the process of feature extraction.

2. The first phase produces a multidimensional, but humans perceive image quality as a single global entity rather than in terms of the local properties of an image. Thus to produce a global visibility evaluation, a spatial pooling strategy is generally applied to down-sample the multidimensional distortion map to a single quality score [61].

3. Since non-linearities, that characterize perception, are not employed in first two phases, the output might not be sufficiently accurate. Thus a fitting strategy could be applied to increase the overall accuracy of the framework. This requires a set of images along with their subjective quality scores (obtained through subjective testing as discussed in section 2.1.1), and a parametric model whose parameters are learned through regression analysis of model predictions on images and their actual subjective scores. This learned model is then used to transform predicted scores into better estimations that are supposedly in line with human perception.

Objective IQA methods are further classified into three broad categories.

2.1.2.1 Full-reference image quality assessment (FR-IQA)

FR-IQA methods aim at achieving the goals of objective IQA while taking both reference and test images as input. Since these algorithms also require reference images for estimating the visual quality, their scope is limited to few applications where reference images are easily available, like image compression and watermarking.

Many FR-IQA algorithms have been proposed over time. As per one of these algorithms, image quality can be computed in terms of peak-signal-to-noise ratio (PSNR), which is simply a ratio of the maximum possible power of a signal and power of distortion. The power of distortion is generally computed in terms of mean-square-error (MSE) to calculate the pixel-wise difference between the reference and distorted image. PSNR has its advantages of being simple and computationally very inexpensive, but it does not deliver very good performance because the essential physiological and psychophysical characteristics of the human visual system (HVS) are not incorporated in this algorithm [36].

Structural similarity index (SSIM)[60] is another FR-IQA algorithm that advances FR-IQA from raw pixels to structures. It is based on the assumption that HVS is highly adapted to extracting structural information present in an image, and image degradation is perceived as a change in this structural information. Hence SSIM aims at assessing the quality of an image by measuring variations in the structural information of distorted images (with respect to their reference image). SSIM has been shown to outperform PSNR in assessing the perceptual quality of images.

Feature similarity index (FSIM)[74] is yet another more recently proposed FR-IQA algorithm. It relies upon the fact that HVS uses low-level features (like edges and zero crossing) for the understanding of images. FSIM make use of two features for estimating the quality of an image: A primary feature called Phase Congruency, which is a contrast-invariant dimensionless measure of the significance of the local structure, and an image gradient magnitude feature. FSIM is shown to deliver superior performance than both PSNR and SSIM algorithms on various datasets.

2.1.2.2 Reduced-reference image quality assessment (RR-IQA)

RR-IQA methods aim at achieving the goals of objective IQA by estimating test image quality while utilizing partial information on reference images. This partial information is usually in the form of features that are extracted from the reference images.

RR-IQA find its application in communication networks that are used for transmitting images and videos. Using RR-IQA algorithms, the partial information on reference images transmitted via these communication networks can be utilized for tracking visual quality degradation of transmitted images and videos [36]. Thus RR-IQA algorithms are preferred over FR-IQA algorithms in similar applications as presented in [45], [9], [43] and [31].

2.1.2.3 No-reference image quality assessment (NR-IQA)

NR-IQA methods aim at achieving the goals of objective IQA by only using test images for estimating the perceptual image quality. These methods are considerably more challenging than FR-IQA and RR-IQA because of the unavailability of any information on reference images. But they are also more desirable because of their application in the wide variety of fields, ranging from image processing to image enhancement, where reference images are usually not available. NR-IQA methods are also used in wide variety of online applications, such as communication systems, image acquisition systems, etc. [4], which makes it very important for them to be computationally inexpensive.

Some early attempts at NR-IQA employed distortion specific methods that approach IQA tasks by making use of models that are very specific to a particular distortion type. These methods are more application specific where prior knowledge of distortion type is available. For example, in an application to measure the quality losses in compressed images, the knowledge on the appearance of compression artifacts, such as blockiness and ringing, could be used to design NR-IQA methods that can detect their visibility.

It is more useful to have algorithms that can be applied for general purpose NR-IQA, irrespective of the distortion types. Existing general purpose NR-IQA approaches could be further divided into two broad categories.

Natural scene statistic based approaches (NSS): The main idea behind these approaches is that the natural scene images bear some statistical regularities that are affected by the presence of distortions, and these statistical changes in the distorted images can be measured to assess the image quality [4]. In these approaches, a test image is first normalized and transformed to another domain (like wavelet domain or DCT), from which the relevant features (such as contrast, shape, variance, etc.) are then extracted and are considered as the descriptors of the distortion. This is the first phase as per the general framework of objective IQA metrics (illustrated in Figure 2.2). The extracted features are then used to perform distortion identification through classifier training. At last for the identified distortion type, as per the third phase of general objective IQA framework, a regression model is applied to map the extracted features to an appropriate quality score [68]. These approaches do not perform pooling and thus skip the second phase of general objective IQA framework. Methods following this approach differ primarily based on the way the features are extracted. In DIIVINE [37] features are extracted in wavelet domain whereas in BLIINDS-II [49], cosine transform coefficient based features are extracted. BRISQUE [35], instead of performing image transformations, extract features directly from the spatial domain, which leads to significant decrease in computation demand.

The NSS based NR-IQA approaches are highly dependent upon handcrafting a large number of features that capture relevant factors affecting image quality. Although their performance is acceptable, there is still large room for improvement regarding accuracy with which they reproduce human judgment of quality. Moreover, the transformation of images to another domain is computationally very expensive, which make these methods (except BRISQUE that does not require domain transformation) very slow in operation.

Feature learning based approaches employ methodologies that instead of using handcrafted features, directly learn features from the spatial domain. Contrary to handcrafted features that are extracted from images according to predefined algorithms and based on expert knowledge, these features are derived during the training process itself. For example, in supervised learning with labeled data, a learning algorithm could learn features (from scratch) by training on raw image pixels, to produce outputs that are very close to the actual labels. Feature learning based approaches are superior and more efficient than NSS based approaches because of their ability to learn better features automatically from raw image pixels. As per the general framework of objective IQA methods (illustrated in Figure 2.2), these approaches first extract features from

input data through training, which is then pooled to produce a global visibility of distortion, and is finally converted to a perceptual quality score by fitting it to a regression model.

CORNIA [66][67] is one such method that first learns codewords from the unlabeled raw image pixels using unsupervised learning, and then using these codewords, it learns features from IQA datasets using supervised learning. By outperforming NSS based approaches, CORNIA showed that it is possible and better to learn features directly from the raw image pixels.

Inspired by the success of CORNIA, a convolutional neural network based approach (IQA-CNN [25]) was introduced to accomplish NR-IQA. Due to the powerful feature learning capability provided by CNN's (see section 2.2), IQA-CNN was able to outperform all existing NR-IQA methods. The proposed IQA-CNN architecture consists of four layers (one Conv layer and three FC layers) and takes a 32 x 32 input patch. Later the updated version of this architecture was also introduced - IQA-CNN++ [26]- which consists of five layers (Two Conv layers and three FC layers) and takes a 7 x 7 input instead. IQA-CNN++ also houses around 90% less learnable parameters in comparison to IQA-CNN. More details on these CNN architectures are presented in section 2.2.3 where we also describe some other popular CNN architectures for object recognition tasks.

CNN based approaches have been further shown to perform better than CORNIA, which is dependent upon a large set of codewords (approximately 300,000) that are expected to learn different characteristics of distortion in an unsupervised manner. The performance of CORNIA has been shown to deplete in case the smaller number of codewords are employed instead. All these approaches employ very small input patches in the order of 7 x 7 or 32 x 32 and assume that image patches bear same quality score as the corresponding image itself. This is perceptually incorrect because distortion could vary among patches based on their content and structural information. Further, the CNN based approaches employ either one or three convolutional layers which limit their architectures from learning high-level features. Since recent research has shown that depth is core to handle complex issues [57], further increase in depth of CNN architectures could boost their performance in NR-IQA tasks.

Thus there is still a lot of room for improvement. A potentially deeper and perceptually more accurate (operating on bigger input patches) CNN architecture could further improve the performance of existing CNN based approaches in NR-IQA tasks.

In next section (section 2.2), we present a literature study on CNN and describe various components that relate to its architecture and training process. These components will be further utilized (in Chapter 4) for designing an envisioned CNN architectures that could overcome the limitations of existing CNN's for the tasks of NR-IQA.

2.2 Convolutional Neural Networks

With the introduction of transistors and computers, past few decades have witnessed a rapid change in the way we operate in our day to day lives. Contrary to pre-computer ages, we have mobile phones to establish long distance communications, fast Internet to access knowledge from another part of the world, and even a full-fledged en-

tertainment system accessible through smartphones and personal computers. Though computers could be programmed to do a wide variety of things with total ease and high efficiency, there still exists a whole family of tasks that could not be decoded into mathematical equations and thus could not be programmed explicitly into a computer chip. Such tasks like object recognition and image quality assessment are dependent upon the very subjective nature of human perception developed over their past experience and knowledge. Since it is not possible to model human perception in terms of general mathematical equations and logic, such tasks could not be directly programmed into computer systems. Thus to facilitate computers to be able to accomplish such non-programmable tasks, a branch of computer science called machine learning deals with the development of algorithms that are capable of giving the computers an ability to learn without being explicitly programmed for [53].

In the complex problems mentioned above, we typically have a situation in which, given an input (an image, a speech segment, any measurement) we want to be able to predict a label y, which represents some sort of characteristic of the input. For example, in image quality assessment, given an input image we want to predict its perceived quality; in object recognition tasks, given an image we want to predict whether a given object is in the image or not; in speech recognition, given a voice recording excerpt, we want to predict the word it represents. Because these are all difficult tasks, designing a model that, in closed form, is able to calculate the label y from the input x is very hard. Learning methods can help in this, which aim at specializing generic models (parametrized by θ) to the task at hand. This "specialization" is handled through a learning phase, where a number of examples of the x-y pairing are proposed to the model, and its parameters θ are (iteratively) updated so that the output $\hat{Y}$ of the model is as close as possible as the desired output y proposed in the examples. Thus a cost function (or loss function) $L(y-\hat{y})$ is generally employed to evaluate the performance of the model by computing how far the model output $\hat{y}$ is from the desired output y. Further, the desired output y could either belong to a set of discrete classes or range of continuous values, based on which the model can be respectively trained towards a classification task or a regression task.

Number of techniques have been proposed in the field of machine learning. One family of techniques that lately has gained momentum is that based on neural networks and connectionist paradigms. Especially the latest advances in deep learning have made neural networks one of the most promising tools for automatic image analysis. In the following we first give a brief introduction to the artificial neural networks, and then later we introduce the branch of deep learning.

Artificial Neural network (ANN) is inspired from biological neurons and working of human brain [62]. An artificial neuron is a building block of ANN that takes in various inputs and performs their weighted sum to produce its output, which further passes through a non-linear function known as activation function or transfer function. As shown in Figure 2.3, ANN's are generally organized into different layers that are made up of the number of interconnected artificial neurons. A general neural network architecture consists of three types of layers, an input layer, one or more hidden layers that process the input examples through learnable parameters (weights and biases), and finally, an output layer that, based on the processing done by its constituent neu-

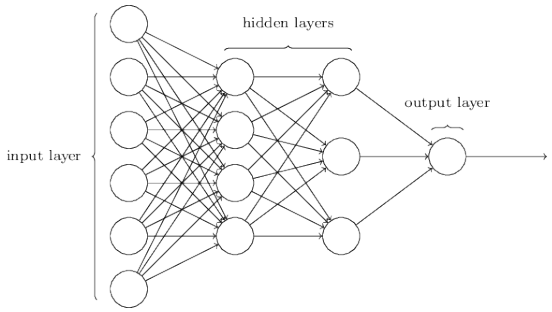


Figure 2.3: Basic structure of artificial neural network

rons, provides an output corresponding to given input examples . More details about training ANN's is provided in section 2.2.2.

Deep Learning is a part of broader machine learning family that is specialized in learning useful features from input data [63]. Input data, for instance an image, can be represented as a group of pixels, mixture of colors, set of edges, a collection of shapes, or even as more complex representations. Difficulties of a learning task could be greatly relieved if useful features are utilized. Traditionally features were handcrafted from spatial inputs,[11] but it is very difficult and inefficient to handcraft feature, especially for some complex applications such as image quality assessment. Thus one of the main ideas behind deep learning is to replace handcrafted features with efficient architectures that are fully capable of learning features through supervised and unsupervised training [55].

As pointed by the word deep, deep learning architectures are characterized by multiple layers stacked one over the other, which consists of artificial neural networks[63]. Thus in deep learning, the input data passes through number of layers and produces higher level of abstractions (features) in comparison to the shallow learning architectures.

Among several deep learning architectures, stacked denoising autoencoders[59], deep belief networks[2], and convolutional neural networks[17] are three of the most popular architectures utilized for different type of applications. Stacked denoising autoencoders and deep belief networks are generally used for the applications of dimensionality reduction (by finding different representations of input data) and feature extraction [1]. Their training process involves layer-wise unsupervised pre-training, which is followed by supervise fine-tuning of all stacked layers through backpropagation.

Although deep neural networks are generally tough to train without unsupervised pre-training, it is still possible to train very deep convolutional neural networks (CNN) using only supervised training [1]. More details about CNN is presented in the following.

Convolutional Neural Network (CNN) is a feedforward neural network inspired from the visual system structure that exists in animals. The visual cortex is known to contain a complex arrangement of cells that are sensitive to sub-regions of the visual field. This visual field is entirely covered by the repetitions of these sub-regions – that act

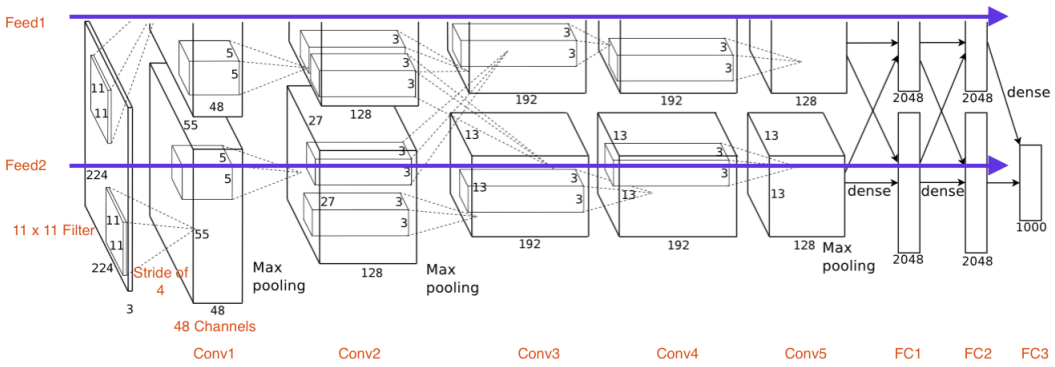


Figure 2.4: Alexnet architecture

as local filters and exploit the present local correlations [20][30]. CNN's imitate this behavior of animal visual cortex to achieve powerful visual processing. It is a field of deep learning that makes use of multiple layers of complex structure or non-linear transformations, and aims at modeling high-level abstractions of input data [1].

Two key features of CNN are:

1. **Local Connectivity**: When dealing with high dimensional inputs such as images, instead of connecting all neurons to every pixel of an image, CNN connects each neuron to a small local region and performs its repetition across the input volume [6].

2. **Parameter Sharing**: CNN makes a simple assumption that if a feature is useful to compute at one spatial location, it will also be useful to compute at another location. This assumption allows filters (see section 2.2.1.1) in CNN to share parameters while calculating output activations at different input locations [6].

These two characteristics help in training very deep CNN's without employing unsupervised pre-training [1].

In the proceeding subsections, we first describe various components of CNN architecture in subsection 2.2.1. Then in subsection 2.2.2 we present different methods components of CNN's training process. Finally in subsection 2.2.3, we present some popular CNN architectures that have been successfully employed in the field of object recognition and image quality assessment.

2.2.1 Convolutional neural network architecture

A CNN architecture consists of many components of which convolutional layers, fully connected layers, pooling units, input feeds and activation functions are described in subsections 2.2.1.1 to 2.2.1.5. Figure 2.4 presents AlexNet[29] CNN architecture, which is used as an example to illustrate various components of a CNN architecture

2.2.1.1 Convolutional layer (Conv layer)

A convolutional layer consists of a set of learnable filters, which refers to the matrix of numerical values that are convolved with the input of a Conv layer, and are characterized by their fixed width, height, and length. Width and height of a filter are specific to design choice, but its length is preset by the number of input channels (which refers to the number of two-dimensional inputs of a Conv layer). For instance, the first convolutional layer with RGB input image can have filters of shape 5 x 5 x 3, where length 3 is specified by 3 RGB input channels. A convolutional layer could contain multiple filters, but they are constrained to be of the same width, height, and length. During the forward pass of backpropagation during training, each filter of a Conv layer slides across the width and height of its input. The gap between two consecutive positions of a filter is called the **stride** and is specific to design choice. At each sliding position, the dot product is computed between the filter parameters and the input values to produce a two-dimensional output. If a Conv layer consists of C filters and each of them produces a two-dimensional output of W width and H height (called feature map), then this Conv layer produces an output of shape H x W x C. As C is the third dimension of output shape, the number of output channels of a Conv layer is decided by the number of filters it employs. Spatial size of the output of a Conv layer can also be controlled by making use of **zero-padding** (Pad), which is a simple technique of adding zeros around the border of the inputs [6].

For a Conv layer l with total C number of filters, the output of its i^{th} filter, denoted by y_i^l, is computed as per equation 2.1

$$y_i^l = s(\sum_{j=1}^{C^{l-1}} f_{i,j}^l y_i^{l-1} + b^l) \tag{2.1}$$

Where b^l is the bias vector of layer l, $f_{i,j}^l$ is the i^{th} filter of Conv layer l that is connected to the j^th feature map of layer $l-1$, and s is the employed activation function.

2.2.1.2 Fully connected layer (FC layer)

All CNN architectures generally employ at least one FC layer that has full connections to all activations from its previous layer. They are usually located between a stack of Conv layers and network output. Contrary to Conv layers, these layers do not support parameter sharing and thus result in substantial increase in learnable parameters within a CNN. The function of a FC layer is to learn weight (W) and bias (b) vectors that maps the layer input to an appropriate output.

For a FC layer l, its output y^l is computed as per equation 2.2.

$$y^l = s(y^{l-1} \cdot W^l + b^l) \tag{2.2}$$

Where W^l and b^l are the respective weight and bias vectors of layer l, and s is the employed activation function.

2.2.1.3 Pooling units (Pool unit)

A Conv layer could be followed by a Pool unit – whose main function is to reduce the dimensionality of Conv layer's output. This unit helps to control overfitting by reducing the overall number of parameters in CNN. This way it also helps in reducing the overall computation power demanded by the network. Similar to Conv layer, Pool unit also consists of a filter that slides, with specified number of strides, across the layer input to produce the layer output. These filters are not learned but are fixed to compute maximum (Max pooling), average (Average pooling) or minimum (Min pooling) of their input regions [6].

2.2.1.4 Input feeds

A CNN architecture could consists of a multiple parallel stacks of layers and units in-between its input and output, which are defined as feeds. As shown in Figure 2.4, AlexNet[29] is a two feed architecture.

As multiple feeds have been shown to specialize in different types of features [29], they are usually employed with an aim to achieve superior performance over single feed architecture.

2.2.1.5 Activation functions

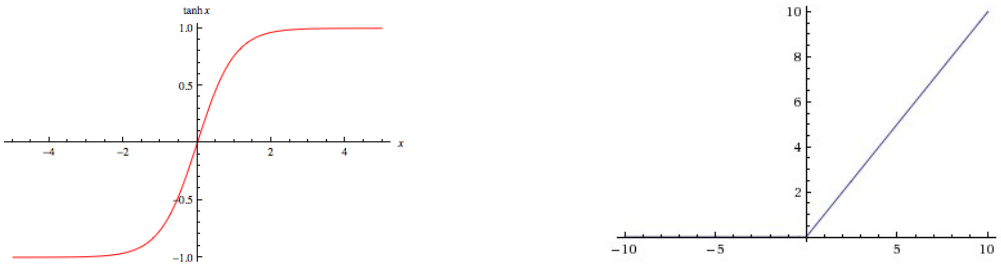
Activation function, also known as transfer function, is used for the purpose of introducing non-linearity in the model implemented by the network. There are various activation functions available around, but few of the most popular ones are describes below:

1. **Tanh** is a traditional activation function that squashes its input value to a range of [-1, 1]. For a given input x, equation 2.3 gives a mathematical operation of Tanh function. Tanh activation is also shown in Figure 2.5a. The saturating activation of Tanh function is one of its main disadvantages – because of which it is very slow (In terms of training time) in comparison to non-saturating activation functions like ReLU [29].

$$Tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.3}$$

2. **ReLU**: The Rectifier Linear Unit (ReLU), given in equation 2.4, is one of the most popular activation functions. It is a simple function that thresholds the activation at zero. For a given input x, equation 2.4 gives a mathematical operation of ReLU function. ReLU activation function is also shown in Figure 2.5b. Because of its non-saturating form, ReLU is shown to greatly accelerate the convergence of a network in comparison to other saturating activation function like Tanh [29]. It is also highly preferred because of the simple mathematical operation it performs, in comparison to more complicated expressions involving exponent term used in Tanh.

$$ReLU(x) = max(0, x) \tag{2.4}$$



(a) Tanh Activation Function (b) ReLU Activatin Function

Figure 2.5: TanH and ReLU activation functions.

3. **Maxout** [16] is an activation function that is implemented using a small sub-network with learnable parameters. Maxout could provide an improvement in performance when used along with Dropout regularization – as it helps in better model averaging performed by Dropout. For a given input vector x, Maxout can be implemented using equations 2.5 and 2.6.

$$R_j = x^T W_j + b_j \tag{2.5}$$

$$h(x) = \max_{j \in [1,R]} (R_j) \tag{2.6}$$

Here W is learnable weight parameter, b is learnable bias parameter, h is the Maxout function and R is the number of linear feature extractors. The learnable parameters W and b are specific to Maxout that are also learned during the training process.

In simple words, Maxout activation function outputs the maximum of R linear combinations of input vector x.

The disadvantages of using Maxout is that it adds extra learnable parameters to the network. These learnable parameters are directly proportional to the value of R and a total number of Maxout units employed.

2.2.2 Convolutional neural network training

This section explains various components relevant to the training of CNN's. Gradient descent optimization, reducing overfitting, weight initialization methods, and data preprocessing are presented in the following subsections.

2.2.2.1 Gradient Descent Optimization

Gradient descent is a widely used algorithm used for the optimization of machine learning algorithms. It aims at minimizing the cost function of a network by updating the network parameters in the direction opposite to the gradient (which is a derivative of the function in one dimension to a function in several functions) of the cost function

with respect to the parameters. Usually, backpropagation algorithm is used along with gradient descent to determine the extent of the update of parameters at each iteration.

Based on the amount of data used for the computation of gradients, gradient descent algorithm can be classified into three main types:

1. **Batch gradient descent**, also known as vanilla gradient descent, computes the gradient of the cost function with respect to network parameters, for the whole training dataset [47]. It can be very slow since entire training dataset is used to compute just one update of network parameters. It is also unfit for applications where training dataset is large as it would require corresponding large RAM to fit in the dataset. Though batch gradient descent results in the most significant parameter update among all three types, it is highly prone to local minima's.

2. **Stochastic gradient descent** computes the gradient of the cost function with respect to network parameters, for a single data point from the training dataset. Since network parameters are updated after the processing of every data point, it is fastest among all three types. It also results in a very noisy gradient that could be helpful in escaping the local minima's, but the resulting gradient could become too noisy for a network to converge.

3. **Minibatch gradient descent** falls in-between full batch and stochastic gradient descent as it employs a subset of the dataset, called a minibatch, for computing the gradient. It provides an option to vary the size of a minibatch, which could help in introducing to the neural network a right amount of noise that is large enough for it to escape local minima's, but also small enough to help converge in a reasonable number of epochs.

There exist many variants of gradient descent algorithm that differ in ways the network parameters are updated. Four of the most popular gradient descent algorithms are described below. In all described algorithms, ε is the learning rate, p_t and p_{t-1} are the learnable parameters at time t and $t-1$ respectively, and dp_{t-1} is the gradient. In neural networks, time is generally reported in terms of number of epochs. In case of CNN's, p consists of both network weights W and biases b.

1. **Vanilla** is the simplest method for updating network parameters, which are updated along the negative gradient direction [6]. Equation 2.7 gives the mathematical expression for vanilla update.

$$p_t = p_{t-1} - \varepsilon * dp_{t-1} \tag{2.7}$$

2. **Momentum** is an improvement over the Vanilla update method. There may exist a scenario where cost function has the form of a long shallow ravine leading to an optimum solution with steep walls around. In such a scenario, Vanilla method results in very slow convergence as it tends to oscillate across the slopes while making small steps towards the optimum solution. Momentum, in this case, helps in accelerating the convergence and dampening the oscillations. Thus Momentum helps the parameter vector in building up velocity in a direction with

consistent gradient [6] Equations 2.8 and 2.9 gives the mathematical expressions for Momentum update method.

$$v_t = m * v_{t-1} - \varepsilon * dp_{t-1} \tag{2.8}$$

$$p_t = p_{t-1} + v_t \tag{2.9}$$

Here v is the momentum parameter that is initialized to zero in the starting, and m is the momentum variable. The function of variable m is to dampen the velocity and oscillations in the system.

3. **Nesterov Momentum**[40] is an improvement over Momentum update method. It aims at achieving faster convergence by giving the momentum term some kind of prescience about the next step. In equations 2.8 and 2.9, if the gradient term is ignored then it would imply that momentum term $(m * v_{t-1})$ updates parameter (p) to its new position [6]. In other words, computing $(p_{t-1} + m * v_{t-1})$ already gives us an estimation of the parameter's next position. Thus in Nesterov momentum method, gradient is computed at position $(p_{t-1} + m * v_{t-1})$ that helps in increasing the responsiveness of the updates. Equations 2.10 and 2.11 gives the mathematical expressions for Nesterov momentum update method.

$$v_t = m * v_{t-1} - \varepsilon * d(p_{t-1} + m * v_{t-1}) \tag{2.10}$$

$$p_t = p_{t-1} + v_t \tag{2.11}$$

Similar to momentum method, here v is the momentum parameter that is initialized to zero in the starting, and m is the momentum variable.

4. **Adam**[27] also known as "Adaptive Momentum Estimation" is a parameter update method that computes adaptive learning rates for each parameter. In previous three methods, learning rate is manipulated globally and equally for all network parameters, which makes it difficult to tune the learning rate and requires very good initialization of the network parameters. Equations 2.12, 2.13, 2.14, 2.15 and 2.16 gives the mathematical expressions for Adam parameter update method.

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * dp_{t-1} \tag{2.12}$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * dp_{t-1}^2 \tag{2.13}$$

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \tag{2.14}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \tag{2.15}$$

$$p_t = p_{t-1} - \frac{\varepsilon * \hat{m}_t}{(\sqrt{\hat{v}_t} + \varepsilon)} \tag{2.16}$$

It could be noticed that Adam keeps the track of exponentially decaying average of old gradients m_t and old squared gradients v_t. Here m_t and v_t are the estimates

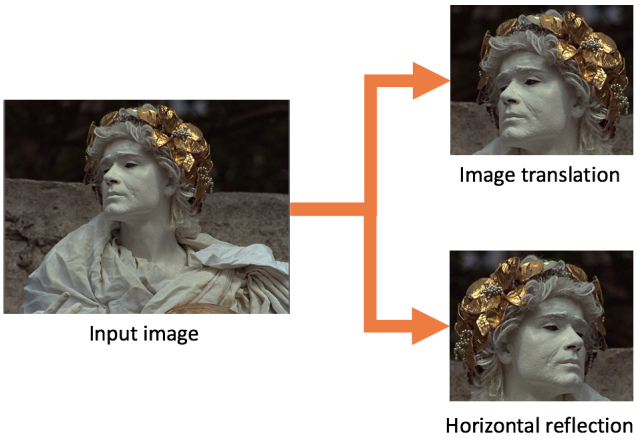


Figure 2.6: Data augmentation technique used for AlexNet training.

of first and second moment of gradient. $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected first and second moments respectively. The smoothing term ε is to avoid division by zero. The authors propose default values of 0.9 for β_1, 0.999 for β_2 and 10^{-8} for ε.

2.2.2.2 Reducing Overfitting

Overfitting is defined as a situation in which a statistical model is too closely fit to a limited set of data points and begins to describe random error or noise instead of its underlying relationship [65]. A network with a large number of learnable parameters is highly prone to overfit, as in such a case the learnable parameters fully adapt to the training set resulting in network to lose its ability to generalize. The following solutions have been proposed over time to cope up with the problem of overfitting.

1. **Data Augmentation** is a technique of artificially enlarging the dataset using various methods. Krizhevsky et al. [29] introduced some of the data augmentation techniques that could be applied in vision related tasks:

   As illustrated in Figure 2.6, Krizhevsky et al. introduced a method of generating image translations (smaller patches) and their horizontal reflections by extracting random 224 x 224 patches and horizontal reflections from the 256 x 256 images. This increased the size of their data set by a factor of $2 * ((256-224) * (256-224)) = 2048$. They also introduced a method of altering the intensities of RGB channels in training images for the same purpose.

   The data augmentation helps by providing extra set of examples for the network to train on, which further helps in learning more robust learnable parameters that promote better generalization of the network on unknown data.

2. **Regularization** aims at imposing stability to ill-posed problems towards avoiding overfitting. Three popular regularization techniques are presented in the following:

   a) **Dropout** [56] is a regularization technique that forces a neural network into learning multiple representations of the same data by randomly switching

off neurons based upon the fixed probability during the learning process. Thus during every training iteration, a little bit different architecture is trained, and the network parameters are updated based on the average computation of all these different architectures. In this way, dropout forces the network into learning more robust parameters that are less dependent upon each other. During the testing phase, the output of neurons is multiplied by the probability value with which they were switched off to ensure that geometric mean of predictive distributions produced during dropouts (while training) is taken into consideration.

b) **L1 Regularization** is a commonly used regularization technique that aims at making the network more invariant to the noisy inputs by making the weight vector more sparse in nature [6] It is implemented as per equation 2.17

$$Cost\_function = Cost\_function + \lambda|W| \tag{2.17}$$

Here $|W|$ is the sum of absolute values of all weights present in a network and λ is the regularisation strength that decides the contribution of $|W|$ term in the cost function

c) **L2 Regularization** is the most commonly used type of regularization in neural networks. It aims at regularizing the network by penalizing the peaky weight vectors while preferring more diffused weight vectors [6]. This prevents the network from highly favoring some of the inputs over the others. L2 regularization promotes linear decay of network weights towards zero. It is implemented as per equation 2.18

$$Cost\_function = Cost\_function + \lambda W^2 \tag{2.18}$$

Here W^2 is the L2 norm of the parameter present in a network and λ is the regularisation strength that decides the contribution of W^2 term in the cost function

2.2.2.3 Weight initialization methods

The overall performance of CNN's is highly dependent upon the initialization of its weights. This requirement becomes more crucial as the depth of an architecture increases. A poorly initialized network could suffer through problems of vanishing or exploding gradients, and might take significantly long to converge than the better initialized counterpart. Thus weight initialization is vital, and many methods have been proposed over the years for good initialization of network weights.

1. **Gaussian Distribution**: As per this method, weights in each layer are initialized from a zero-mean Gaussian distribution with small standard deviation. It is the most basic and widely used method for weight initialization. Krizhevsky et al. [29] also uses this method from the initialization of its weights.

2. **Xavier Initialization**[14] is a weight initialization method that aims at initializing weights that are not too small or large, but are just right to keep the passing signal in the reasonable range of values. Xavier initialization also initialize weights from a zero-mean Gaussian distribution but instead of randomly selecting the standard deviation value, this method proposes to calculate standard deviation using equation 2.19

$$\sigma = \sqrt{\frac{1}{n_{in}}} \tag{2.19}$$

Here n_{in} is the number of inputs to the layer.
Equation 2.19 was derived while using traditional activation functions such as *tanh* and *sigmoid*. Later [19] repeated the derivation for *ReLU* activation functions and proposed to calculate standard deviation using equation 2.20.

$$\sigma = \sqrt{\frac{2}{n_{in}}} \tag{2.20}$$

3. **Batch Normalization**[21] is not a weight initialization method in itself but it aims at avoiding the need for having proper weight initializations. Apart from this, it also helps in accelerating the whole training process by allowing higher learning rates and eliminating the need for regularization like the dropout. It is also shown to give greater accuracy in deep architectures. Batch normalization requires a minibatch of size greater than one to operate. Hence it could not be applied for stochastic gradient descent optimization.

Typically the layer activations are made too big or too small by the network parameters – referred to as "internal covariate shift" by Sergey et al. [21]. Batch normalization method helps in avoiding such situations by forcing the activations passing through the network to take a unit Gaussian distribution (zero mean and unit variance) across the minibatch. It is usually applied over the existing layers of the network, but could also be implemented over few selected layers. Equations 2.21, 2.22, 2.23 and 2.24 gives the mathematical expression for batch normalization method.

$$\mu_B = \frac{1}{N} * \sum_{i=1}^{N} x \tag{2.21}$$

$$\sigma_B^2 = \frac{1}{N} * \sum_{i=1}^{N} (x - \mu_B)^2 \tag{2.22}$$

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + c}} \tag{2.23}$$

$$y = \gamma * \hat{x} + \alpha \tag{2.24}$$

Here x and y are the respective inputs and outputs of batch normalization layer, N is the size of a minibatch, and μ_B and σ_B^2 are the computed mean and variance

across the minibatch respectively. In equation 2.23, the input is normalized to have zero mean and unit variance across the minibatch, and c is the constant to prevent division by zero. Finally the output of the batch normalization layer is computed in equation 2.24 by scaling the normalized input by γ and shifting it by α. Here γ and α are both learnable parameters.

Since the test dataset could not be expected to have minibatches, usually the running mean and variance is calculated during the training process, and their obtained values are used in equation 2.23 instead. Alternative to this is to make a post-training run for computing the average mean and variance values across the minibatches.

Batch normalization method has shown to produce promising results especially in the field of convolutional neural networks, but because of the additional mathematical operations it performs, it could significantly increase the overall computation demand of the training process.

4. **Data Dependent initialization**[34] is a weight initialization method inspired from batch normalization. It aims at keeping all the advantages offered by batch normalization while resulting in an insignificant increase in the computation demand of the training process.

   In data dependent initialization, all layers are first initialized with a non-zero Gaussian distribution of small standard deviation. This is followed by the layer-wise fine-tuning of weight initialization with the help of activations produced by the selected subset of the training dataset. For a given layer, the weights are fine-tunes using equations 2.25 and 2.26.

$$\sigma_B^2 = \frac{1}{N} * \sum_{i=1}^{N} (x - \mu_B)^2 \tag{2.25}$$

$$W = \frac{W}{\sqrt{\sigma_B^2 + c}} \tag{2.26}$$

   Here x is the input to the data dependent initialization method, σ_B^2 is the computed mean across the minibatch, N is the size of a minibatch, and c is the constant to prevent division by zero. For each layer, equations 2.25 and 2.26 are repeated until equation 2.27 is satisfied.

$$|\sigma_B^2 - 1| \geq Tol_{var} \tag{2.27}$$

   Here Tol_{var} is called variance tolerance is usually set in the range of 0.01 to 0.1.

   [28] and [50] are some other papers that propose similar methods of weight initialization based on the input data.

2.2.2.4 Data pre-processing

The performance of CNN is directly proportional to its ability to extract the good feature from the input data. Thus sometimes it is desirable to pre-process the input data so as to make this feature extraction process more easy and effective. Two of the main data pre-processing techniques used in literature are:

1. **Global contrast normalization (GCN)** is a pre-processing technique that normalizes each channel (RGB) of an input image to zero mean and unit standard deviation. It helps by making the network more robust to illuminations and contrast variations [25]. Moreover, it is widely considered as a good practice to be used in CNN, especially for object recognition tasks [7]. For an image of length X, width W and Z number of channels, GCN value can be computed as per equations 2.28, 2.29 and 2.30.

$$\hat{I}(x,y,z) = \frac{I(x,y,z) - \mu(z)}{\sigma(z) + c} \tag{2.28}$$

$$\mu(z) = \frac{1}{(X * Y)} \sum_{i=1}^{X} \sum_{j=1}^{Y} I(i,j,z) \tag{2.29}$$

$$\sigma(z) = \sqrt{\frac{1}{(X * Y)} \sum_{i=1}^{X} \sum_{j=1}^{Y} (I(i,j,z) - \mu(z))^2} \tag{2.30}$$

   Here $\hat{I}(x,y,z)$ is the GCN value of a pixel at location (x,y,z) of intensity $I(x,y,z)$; $\mu(z)$ and $\sigma(z)$ are the respective mean and standard deviation computed along the z channel of the input image; and $c = 1$ is a positive constant that prevents division by zero.

2. **Local contrast normalization (LCN)** is another pre-processing technique that locally normalizes each channel (RGB) of an input image to zero mean and unit standard deviation. It is believed to help by decorrelating image pixels [35]. Previous NR-IQA methods, such as BRISQUE[35], IQA-CNN[25], and IQA-CNN++[26], also perform local contrast normalization of input images. For an image of length X, width W and Z number of channels, GCN value can be computed as per equations 2.28, 2.29 and 2.30.

$$\hat{I}(x,y,z) = \frac{I(x,y,z) - \mu(x,y,z)}{\sigma(x,y,z) + c} \tag{2.31}$$

$$\mu(x,y,z) = \frac{1}{k^2} \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{j=-\frac{k-1}{2}}^{\frac{k-1}{2}} I(x+i, y+j, z) \tag{2.32}$$

$$\sigma(x,y,z) = \sqrt{\frac{1}{k^2} \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{j=-\frac{k-1}{2}}^{\frac{k-1}{2}} (I(x+i, y+j, z) - \mu(z))^2} \tag{2.33}$$

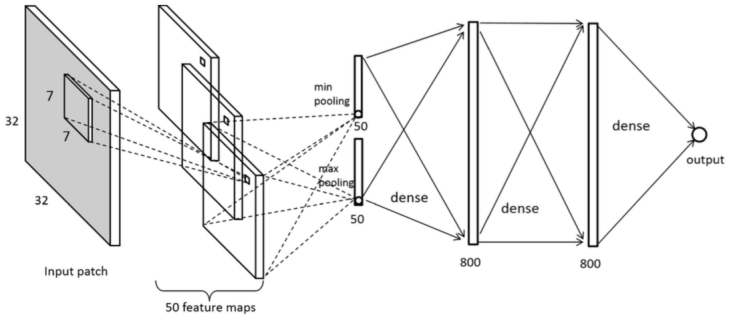


Figure 2.7: IQA-CNN architecture [25] for the application of NR-IQA.

Here $\hat{I}(x,y,z)$ is the LCN value of a pixel at location (x,y,z) of intensity $I(x,y,z)$, $\mu(x,y,z)$ and $\sigma(x,y,z)$ are the respective mean and standard deviation computed along the z channel of the input image, $c = 1$ is a positive constant that prevents division by zero; and k is the local normalization window over which the mean and standard deviation is computed. k should always carry an odd value to symmetrically cover local pixels around the pixel whose LCN value is to be computed. It has been shown in [35] that smaller values of k results in better performance in IQA related tasks.

2.2.3 Applications

From past few year, CNN architectures have gained immense popularity and CNN has become a default choice in many computer vision related tasks. AlexNet[29] was the first work of its kind that popularized CNN's for Computer Vision. As shown in Figure 2.4, AlexNet consists of five Conv layers followed by three fully connected layers and has around 60 million learnable parameters. It was the winner of ImageNet ILSVRC 2012[10] challenge (Object recognition challenge involving 1000 object classes) and gave superior results with top 5 error (Error in recognizing object within 5 closest classes) of 15.3%. Many success stories with CNN architecture followed AlexNet.

ZFNet[72] was the winner of ILSVRC 2013 competition. It is an improvement over the AlexNet obtained through tweaking of its hyperparameters. GoogleNet[57] was the winner of ILSVRC 2014. The proposed architecture consists of 22 layers and 4 million parameters and recorded top 5 error of 6.67%. ResNet[18] was the winner of ILSVRC 2015, and reported top 5 error of 3.57%. The proposed architecture consists of 152 layers and is by far state of the art CNN architecture for object recognition tasks.

Between 2012 and 2015, from AlexNet to ResNet, the depth of CNN architecture increased from 8 layers to 152 layers and the performance on ILSVRC dataset increased from 15.3% to 3.57% top 5 error. These numbers very well reflect the popularity of CNN's and the amount of research happening in this field.

CNN's have also been successfully applied for the application of IQA. Kang et al. in [25] proposed a fully supervised CNN architecture, called IQA-CNN, for the application of NR-IQA. IQA-CNN, as shown in Figure 2.7, takes a 32 x 32 input patch

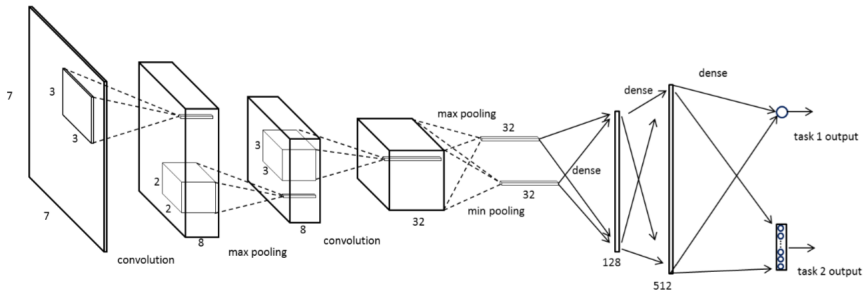


Figure 2.8: IQA-CNN++ architecture [26] for the application of NR-IQA.

that is processed by one Conv layer with 50 filters, followed by max and min pooling to produce single max and single min value for each of its feature maps. This is further followed by three FC layers, and an output of the last layer is used for regression to output image quality score. Also, no activation function is employed in its Conv layer and last FC layer whose output is used for regression. Apart from this, ReLU activation functions are used in rest of its FC layers. IQA-CNN contain total 724,901 learnable parameters. This architecture reported state of the art performance on LIVE dataset[51] among other NR-IQA metrics

After the success of IQA-CNN, another CNN architecture called IQA-CNN++[26] was introduced for the application of NR-IQA. This architecture, as shown in Figure 2.8, takes a 7 x 7 input patch that is processed by three Conv layers, which is followed by max and min pooling to produce single max and single min value for each of its feature maps. This is further followed by three FC layers, and the output of last FC layer is used for regression to output image quality score. It also performs classification in its last layer to predict the distortion type of input patches. Again no activation function is employed in any of its Conv layers and last FC layer. Apart from this, ReLU activation functions are used in rest of its FC layers. IQA-CNN++ contain total 77,501 learnable parameters, which is around 90% less than IQA-CNN. This architecture reported a state of the art performance on TID dataset[41] among other NR-IQA metrics.

2.3 Chapter conclusion

In this chapter, we presented background information related to the fields of image quality assessment (IQA) and convolutional neural network (CNN). Based on the literature study on IQA, it was observed that NR-IQA methods are very desirable as their operation is independent of the availability of reference images. But these methods are also more challenging because the unavailability of reference images makes it difficult to reason regarding image fidelity (inferred by the ability to discriminate between two images [52]). In spite of all these challenges, CNN based approaches have been shown to deliver superior performance in comparison to other NR-IQA methods. But there still exist a lot of room for improvement. For instance, the CNN based approaches operate on very small patches of input images and assume image quality to be uniformly distributed across images. This is perceptually inaccurate, as based on their structure and content, different patches could bear different quality scores. Further

existing CNN based approaches only utilize few Conv layers (max three) that prevent them from exploiting the full power of CNN architectures.

Furthermore, based on the literature study on CNN, it was observed that CNN's have been very successful especially in the field of object recognition. There is a lot of research happening, and many methods have been proposed over time to optimize and accelerate the working of CNN's. Many of these methods like batch normalization, Adam, etc. were also discussed in detail.

Thus based on these studies, we believe that better CNN architectures could be developed for the application of NR-IQA.

Chapter 3

Methodology

From the literature study presented in Chapter 2, we established that NR-IQA is the most desirable setting of objective IQA, but it is very tough to achieve due to the unavailability of reference images. We further observed that CNN based approaches are the state of the art among other NR-IQA methods, but they undergo many limitations because of the small size of employed input patches and less number of utilized Conv layers.

Based on the research objectives (section 1.1) and background literature (Chapter 2), we identified three phases of this research project, with an aim to address the limitations of existing CNN based approaches for NR-IQA tasks and to understand the internal working of trained CNN through feature visualizations.

Design phase : This phase deals with the design space exploration of CNN and aims at selecting various design parameters that constitute a suitable CNN design for the application of NR-IQA. This phase is linked to the research objective 3.

Evaluation phase : This phase deals with the testing of CNN design obtained from the design phase, with an aim to evaluate and compare its performance with existing state of the art NR-IQA algorithms. This phase is linked to the research objective 4.

Inspection phase : This phase deals with the visualization of features learned by CNN architecture, obtained from the design phase, to understand the internal working of CNN architecture.This phase is linked to the research objective 6.

The completion of first and second phase answers the first research question (*What is a suitable design for a convolutional neural network to predict perceived image quality in a no-reference setting?*), and the completion of the third phase answers the second research question (*What can we learn from the visualization of features developed by a convolutional neural network trained for image quality assessment?*).

For each of the design, evaluation and inspection phase, we employed a three phase methodology respectively presented in sections 3.1, 3.2 and 3.3.

3.1 Methodology for design phase

To undertake this design space exploration, we considered various design parameters of CNN based on the literature study presented in section 2.2, from which we identified two main categories of CNN design parameters.

1. **Training process design parameters**: These are defined as the design parameters that are utilized for the training of a CNN. These design parameters are fixed before the initialization of CNN architecture. Based on the literature study, we identified six main types of design parameters related to the training process of a CNN:

    - *Training method*: It refers to the method of CNN training, which could be either supervised or unsupervised. The selection of training method depends on the desired output of a CNN, and thus depends on the target problem.
    - *Learning task*: It refers to the type of task a CNN approaches towards, which could either by classification or regression. The selection of a learning task also depends on the target problem.
    - *Cost function*: It refers to the mathematical function that is used to assess the difference between the produced output by CNN and its desired output. A cost function is important to guide a CNN towards the reducing cost, i.e., to decrease the difference between produced and desired outputs. Since the employment of different cost functions could influence the overall performance of a CNN, they are considered as the design parameters that should be carefully selected.
    - *Optimization methods and algorithms*: It refers to the methods and algorithms using which the learnable parameters of CNN are updates. Since the employment of different optimization methods and algorithms could influence the overall performance of a CNN, they are considered as the design parameters that should be carefully selected. Various optimization methods are presented in section 2.2.2.1.
    - *Regularization*: It refers to the techniques that could be utilized towards the prevention of overfitting. Since the employment of different regularization techniques could influence the overall performance of a CNN, they are considered as the design parameters that should be carefully selected. Various regularization techniques are presented in section 2.2.2.2
    - *Weight initialization method*: It refers to the methods that could be utilized for the initialization of CNN weights during the starting of the training process. Since the employment of different weight initialization methods could influence the rate of convergence of a CNN, they are considered as the design parameters that should be carefully selected. Various weight initialization methods are presented in section 2.2.2.3

2. **Architecture design parameters**: These are defined as the design parameters that are related to or based on the architecture design of a CNN. From the liter-

ature study, we identified three main types of design parameters of CNN architecture.

- *Network layers*: The components of CNN that contain learnable parameters (weights and biases). For example, Conv layer and FC layer. Different types of layers, number of layers, order of layers, configuration (for example, number of channels in Conv layers, or number of neurons in FC layers, etc.) of layers in a CNN architecture could influence its overall performance, and thus all these settings of layers are the design parameters of CNN architecture and should be carefully selected. These layers are presented in section 2.2.1.
- *Network units*: The components of CNN that perform a predefined operation on the output of a layer. For example, Pool unit, LCN unit, GCN unit, and LRN unit. Different types of units, number of units, order of units, configuration (for example, a Pool unit could perform max-pooling or average pooling, etc.) of units in a CNN architecture could influence its overall performance, and thus all these settings of units are the design parameters of CNN architecture and should be carefully selected. These units are presented in section 2.2.1 and 2.2.2.4.
- *Activation functions*: Different types of activation functions could perform different operations on the output of CNN layers and could affect the overall performance of a CNN. Thus different types of activation functions are the design parameters of CNN architecture and should be carefully selected. They are presented in section 2.2.1.5.

For these two categories of design parameters, we used the following four-step methodology for their exploration:

Step 1 As the first step, we started by exploring the training process design parameters because they constitute the basis of CNN training and it is not possible without them to construct or train any CNN architecture. Our method of selection of these parameters was either based on the references from literature study, or through a set of initial experiments that reflected their usefulness.

Step 2 After the selection of training process design parameters, it was possible to construct and train a CNN architecture. Thus based on the literature study, we initialized a CNN architecture to be used for the further exploration of architecture related design parameters. To conduct proper investigation of these design parameters, we designed an experimental setup to evaluate the performance of various architecture design parameters using publicly available benchmarking dataset (TID[41]) and image content-sensitive 5 fold cross-validation.

Step 3 After this, we further explored different architecture design parameters that contribute to the overall increase in the performance of a CNN design. By evaluating their performance on the selected experimental setup, the design parameters were either selected or rejected from their inclusion to the final CNN design.

Step 4 Finally, based on the results of the design space exploration, two CNN architectures were proposed, one for the prediction accuracy and another for minimizing the computational complexity.

3.2 Methodology for evaluation phase

This section presents the three-step methodology used for the testing of the CNN designs obtained from the design phase. This methodology is linked to Chapter 5

Step 1 : Based on the literature study, we designed an experimental setup to evaluate the proposed CNN architectures obtained from the design phase, using publicly available benchmarking datasets (LIVE[51] and TID[41]) and image content-sensitive cross-validation.

Step 2 : The two CNN architectures resulting from the design phase were first compared among each other in order to better understand the trade-off that could be achieved between the complexity and accuracy. The comparison was made in terms of their performance on selected evaluation metrics.

Step 3 : The proposed CNN design from the design phase were then compared to the state of the art IQA methods to verify whether the proposed innovations (more depth and use of larger image patches as input) were of added value, again in terms of their performance on evaluation metrics.

3.3 Methodology for inspection phase

Though CNN's have been known to give superior performance in many application fields, their internal working is very difficult to understand especially because of their big size and a large number of learnable parameters. Thus to obtain a better understanding, we selected two feature visualization techniques namely Synthetic-Max and Image-Max.

Synthetic-Max : This visualization technique generates a synthetic image that maximally activates a target neuron in a CNN.

Image-Max : This visualization technique selects an image patch from the training dataset, that maximally activates a target neuron in a CNN.

The selected techniques aim at generating or extracting image patches that maximally activate the target neuron. Thus they helped in the visualization of characteristics of input images that a target neuron is responsible for detecting. They were also attractive because of their capability to generate unique and informative visualizations of features learned by even deeper layers of a CNN. This methodology is linked to Chapter 6

Chapter 4

Convolutional Neural Network Design

In section 2.1, various IQA methods and algorithms were presented, and it was shown that NR-IQA (a branch of objective IQA) is the most desirable method because of its non-requirement of a reference image and wider scope of applications. Further, it was shown that feature learning based approaches of NR-IQA are more efficient and superior to traditional handcrafted feature-based approaches. Thus in this thesis project our goal was to approach NR-IQA by making use of a feature learning based approach of CNN.

Existing feature learning based approaches ([67],[66],[25] and [26]) for NR-IQA suffer from two main limitations:

- *Loss of global information*: Existing feature learning-based approaches, as reported in section 2.1, often assume distortions to be homogeneously distributed across the image, and their perceptual impact to be just as strong throughout, independent on the structure and content of the image region where they are located. As a consequence, common approaches are based on feeding learning algorithms with very small portions of the images (so-called patches), of the size of e.g. 32 x 32 or even 7 x 7 (which is even smaller than the typical block size used by compression algorithms such as JPEG). The learning algorithms are then trained to predict, based on these small image patches, the quality of the overall image. In other words, the quality score of the entire image is assigned as a label to all small patches extracted from the image. Though this approach is advantageous in artificially increasing the size of training datasets, which further promotes generalization, it is perceptually incorrect. Depending upon the content and structural information contained in these patches, image patches could bear quality score much different from each other. Thus it is possible to improve these approaches by making use of bigger input patches that are capable of incorporating more global information on input images.

- *Lack of depth*: Existing CNN based approaches for NR-IQA only employ maximum three Conv layers for the purpose of feature extraction. This lack of depth prevents them from learning higher levels of features from input images. Since it has been shown in [18] that higher level features could simplify a much complex problem of object recognition into delivering superior performance, with

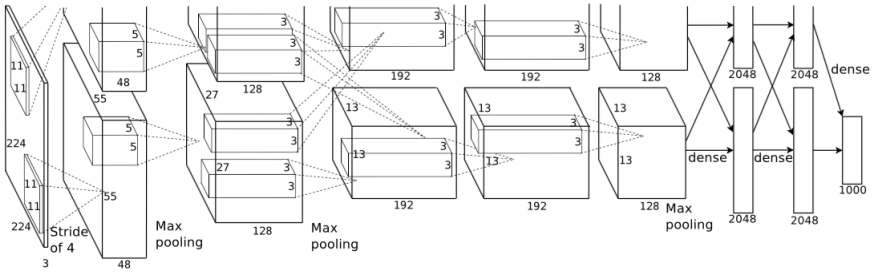


Figure 4.1: AlexNet architecture [29]

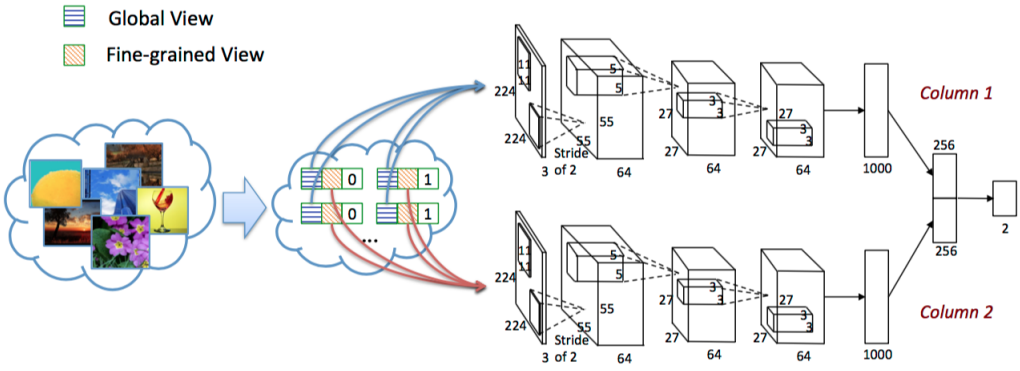


Figure 4.2: RAPID architecture [33]

the employment of more number of layers, NR-IQA based CNN architectures could also improve upon their current level of performances.

With an aim to overcome the above-mentioned limitations, we considered an AlexNet based two feed CNN architecture called RAPID [33], which was proposed for the task of image aesthetic quality assessment. RAPID architecture, shown in Figure 4.2, employs two identical feeds called local feed for processing local view of an image, and global feed for processing global view of an image. The local feed takes image crops as input, and the global feed takes image warps (reshape image to input size of CNN) as input. This type of architecture meets our first requirements by taking into consideration the global information of input images.

Since the RAPID architecture was refined from AlexNet and was proposed for a different application of image aesthetic quality assessment, instead of directly adopting RAPID for our application of NR-IQA, we decided to start fresh from AlexNet architecture while adopting the idea of using local and global feeds from RAPID. Since AlexNet consists of five Conv layers, considering AlexNet helped us in fulfilling our second requirement of deeper CNN architecture.

We further adopted the idea of using identical local and global feeds, because first of all the good results reported by RAPID in [33] showed that its is an efficient design choice, and secondly, it also reduces the design space to be explored for NR-IQA task based CNN architecture design.

Based on the above discussions, we decided to start the design space exploration

with the only local feed of AlexNet based CNN architecture. Global feed was analysed later after the local feed was configure with selected design parameters. We named our target CNN design for the application of NR-IQA as IQA-DCNN: Image Quality Assessment using Deep Convolutional Neural Networks. The starting configurations of IQA-DCNN, with only local feed, are described below.

IQA-DCNN architecture

AlexNet contains approximately 60 million learnable parameters and was originally trained on ILSVRC2012 [10] dataset of 1.2 million training images. Since available IQA datasets are not very big (TID[41], the biggest IQA dataset contain 1700 images), directly adopting AlexNet architecture for the application of NR-IQA would result in high levels of over-fitting. Thus instead of directly using AlexNet, its simplified version was used so as to have less number of learnable parameters in IQA-DCNN. The version zero (V-0) of the IQA-DCNN architecture is shown in Figure 4.3.

Similar to RAPID, we utilized only one feed of AlexNet for constructing the local feed of IQA-DCNN. To decrease the number of learnable parameters, we dropped one FC layer from the 3 FC layered AlexNet architecture and utilized FC1 containing 1000 neurons (similar to RAPID) and FC2 with one neuron to provide regression output. As in AlexNet, we also employed a dropout unit with dropout probability of 0.5 over the output of FC1 layer. The utilization of a dropout unit could prevent overfitting by regularizing the IQA-DCNN architecture.

We then adopted the structure of convolutional layers of AlexNet and configured IQA-DCNN to contain five Conv layers (Conv1 to Conv5), with three overlapping max pool units (Pool1, Pool2 and Pool3 respectively added to the output of Conv1, Conv2, and Conv5 layers), and two local response normalization units (LRN1, and LRN2 added further to the outputs of Pool1 and Pool2 units respectively). We configured Conv1, Conv2, Conv3 and Conv4 layers with 100 channels for simplification. After this, the Conv5 layer was configured with 25 channels because it is an interface between Conv and FC layers, and since the parameters of FC layers are not shared (unlike Conv layers), connecting them to high number of channels would result in very high total amount of learnable parameters to be trained.

Further following AlexNet, ReLU activation unit was used in all layers of IQA-DCNN, except in FC2 layer whose output is used for regression. Rest of the configurations including filter shapes, strides and zero-padding were also directly adopted from the AlexNet architecture and are explicitly illustrated in Figure 4.3.

The IQA-DCNN operates on 227 x 227 RGB input patches and outputs a corresponding image quality score to the processed inputs. It should be noted that AlexNet employed 224 x 224 size of input patches, but it was pointed out in [6] that 224 x 224 input size with 11 x 11 filters configured with the stride of 4 would result in some of the filters to unsymmetrical fit across the network inputs. Thus to avoid this, inputs of size 227 x 227 should be used instead. This mistake was later rectified in reimplementation of AlexNet.

The selected configurations of IQA-DCNN are merely the starting configurations of IQA-DCNN, many of which were further optimized during the design space exploration of architecture design parameters (presented in section 4.3).

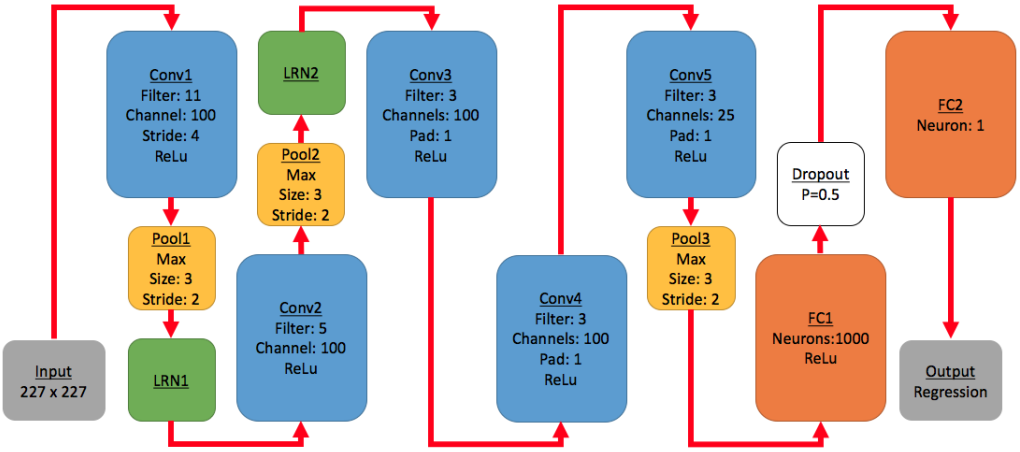


Figure 4.3: IQA-DCNN architecture: V-0

Though the configurations of version zero (V-0) of IQA-DCNN were motivated from successful AlexNet and RAPID architectures, it was still very challenging to efficiently train it. The biggest challenge was the small size of available IQA datasets for CNN training. The original AlexNet architecture was trained on the ILSVRC-2012[10] dataset of 1.2 million images, and RAPID was trained on AVA[39] dataset of 250,000 images. Contrary to this, the biggest IQA dataset (TID[41]) merely contains 1700 images in total, which is a very small number in comparison. The situation became even more challenging as the bigger size of employed input patches also drastically reduced the total number of distinct training patches (in comparison to other CNN based NR-IQA approaches) that could be used for IQA-DCNN training.

Thus all these challenges posed the need for very careful selection of CNN design parameters. In section 4.1, we describe our choice of different training process design parameters that were employed for the training of IQA-DCNN. With selected training process design parameters, in section 4.2, we describe the experimental setup that was used for comparing and selecting various architecture related design parameters. Using this experimental setup, in section 4.3, various design parameters were chosen for the configuration of the local feed of IQA-DCNN. Then in section 4.4, we explore the possibility of adding a global feed to the configured local feed architecture. Finally in section 4.5, we conclude the findings of this Chapter.

4.1 Training process design parameters

As described in section 3.1, training process design parameters are defined as the design parameters that are utilized for the training of a CNN. Selection of these design parameters is described is subsections 4.1.1 till 4.1.3.

4.1.1 Learning task, training method and cost function

Since image quality is presented as a range of continuous values (rather than discrete classes), it was obvious for us to train IQA-DCNN towards a regression target.

To fix the training method, we considered two ways in which IQA-DCNN can be trained:

1. Supervised pre-training using a bigger dataset, followed by supervised fine-tuning using IQA dataset.

2. Supervised training using only IQA dataset.

We did not consider unsupervised training because CNN's are best known as supervised learning algorithms and have been shown to portray superior performance with full supervised learning in [29], [57] and [72].

In the case of deep CNN's, it is desirable to have large amount of training examples for better generalization. Since most of the IQA datasets are very small in size, we considered a pre-training step using ILSVRC2012[10], which is a bigger dataset (with 1.2 million images) for object recognition tasks. We figured that, since CNN's are claimed to extract features at different levels of abstraction, using an object recognition dataset for pre-training would have helped in learning basic features in lower layers, which we assume to be the same for any task (because they model the early stages of human vision - color and orientation processing). Features in higher layers, corresponding to higher levels of abstraction (perhaps corresponding to the task specialization), would have then been fine-tuned for image quality assessment with an IQA dataset.

Using the method of supervised pre-training, we were never able to successfully fine-tune IQA-DCNN using pre-trained weights of the ILSVRC-2012 dataset. We found that it was very difficult to train a CNN with a pre-training step on the dataset from a very different domain. This could be because relevant features (especially high-level features) for IQA task are much different than features developed through training on object recognition dataset, and selectively dropping some of the co-adopted layers could prevent CNN from converging [70].

Thus based on these observations, we decided to only use IQA datasets for the supervised training of IQA-DCNN.

With learning task and training method set, we considered two popular cost functions, L1 norm loss and L2 norm loss, for the regression target.

It was shown in [32] that the L2 norm is more stable and produces a unique solution in comparison to L1 norm, which is less stable and produces multiple solutions. Since properties like stability and ability to provide a unique solution are important for smooth convergence of CNN to a global optimum, we decided to use L2 norm loss as the cost function for IQA-DCNN. Furthermore, L2 norm was also recommends in [6] as a cost function for CNN's with regression task.

Hence we decided to train IQA-DCNN towards a regression task, by using only IQA datasets through supervised training method, while employing L2 norm cost function.

4.1.2 Optimization method

Gradient descent optimization using backpropagation is a standard method utilized for the optimization of CNN's. Many popular CNN architectures ([29],[57],[72],[18], etc.) have been successfully trained using this method. Motivated by this, we also employed gradient descent based optimization method along with backpropagation for training IQA-DCNN.

In this section, we describe our selection of the gradient descent optimization method and the gradient descent optimization algorithm. Detailed background information on these methods and algorithms is presented in section 2.2.2.1.

We considered three possible gradient descent optimization methods based on the number of training examples employed for the weight update: Batch gradient descent, stochastic gradient descent, and minibatch gradient descent.

Through some initial trials, it was observed that batch gradient descent optimization could not be employed because operating IQA-DCNN on a full batch of images poses unrealistically large requirement on RAM size. Furthermore, it is also known to be a very slow in comparison to other two methods. In addition, some trials with stochastic gradient descent optimization method showed it to very unstable (because of very noisy gradients), preventing the IQA-DCNN from converging. Finally, other trials showed minibatch-based gradient descent to be a good compromise. Being it widely used by a number of popular CNN architectures (AlexNet[29], ResNet[18], etc.) we could rely on previous literature to foresee its application to be effective.

Based on these observations, we selected minibatch gradient descent as the optimization method for IQA-DCNN.

4.1.2.1 Gradient descent optimization algorithm

Having selected minibatch gradient descent optimization method, we considered four gradient descent optimization algorithms: Vanilla, Momentum, Nesterov Momentum, and Adam.

Since Adam[27] is an intelligent gradient descent optimization method that employs adaptive learning rates for each of the network parameters, it was considered for IQA-DCNN. But in our initial trials, it was observed that using Adam makes IQA-DCNN very unstable and prevents it from converging. This could be due to different learning rates adopted by learnable parameters that could make the whole CNN very unstable especially when it is being trained to achieve a regression target. Thus Adam was rejected.

We further trained IQA-DCNN using Vanilla, Momentum and Nesterov momentum on TID dataset (presented in section 4.2.2) with minibatch size of 32 (which is not very large or very small), and L2 norm as the cost function. As shown in Figure 4.4, it was observed that Momentum and Nesterov momentum results in faster convergence and lower training cost in comparison to Vanilla method. Since Nesterov momentum was further observed to be faster than Momentum method in early epochs, Nesterov momentum was selected as a gradient descent optimization algorithm for IQA-DCNN.

Equations 2.10 and 2.11 were utilized for the update of learnable parameters with Nesterov momentum. In equation 2.10, we used $m = 0.9$ as recommended in [6].

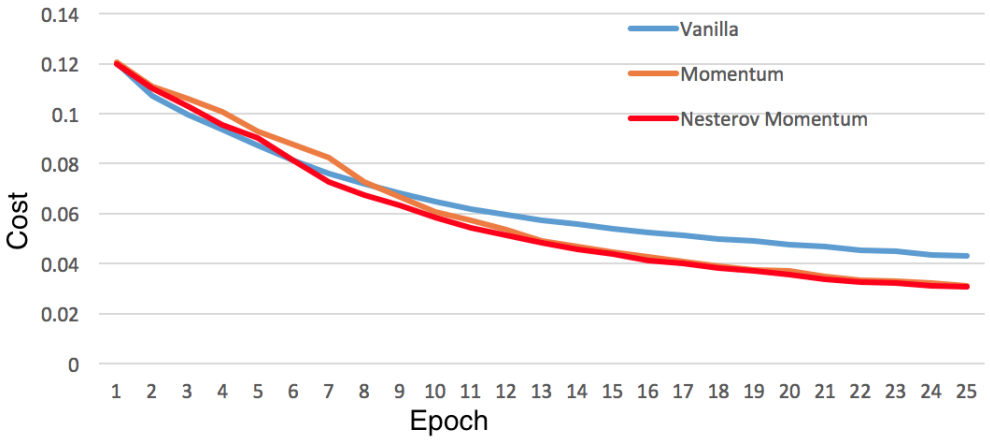


Figure 4.4: Training cost (L2 norm) vs Epoch graph: Comparison of convergence speed between Vanilla, Momentum, and Nesterov Momentum gradient descent optimization algorithms.

Further the learning rate ε was updated in epoch t using equation 4.1, which was adopted from [25] because of its reported superior performance.

$$\varepsilon = 0.02 * (0.9)^t \tag{4.1}$$

As an outcome of this section, we decided to use **Nesterov momentum**, in minibatch gradient descent setting, as a gradient descent optimization algorithm for IQA-DCNN.

4.1.3 Weight Initialization

Weight initialization plays an important role in the efficient functioning of CNN's. Good weight initialization can accelerate the overall training process whereas bad weight initialization can prevent a CNN from converging.

We considered four weight initialization methods for the application of IQA-DCNN: Gaussian distribution method, Xavier initialization, batch normalization, and data dependent initialization. These methods were selected because of their superior performance reported in the literature. Detailed background information on these methods is presented in section 2.2.2.3.

In some initial trials, it was observed that utilization of Xavier initialization and data dependent initialization method prevent IQA-DCNN from converging. The reason behind this could be the regression target for which the IQA-DCNN is trained. Training to solve a regression task is not a very stable setting in itself as it requires CNN to learn very specific configurations of weights to output precise values for a given set of inputs, which is not the same with classification task that employs Softmax functions, which render precise output values less important. Since these weight initialization methods have been shown to display good performance for classification tasks in [19], [28], [34], and [50], they don't seem to perform well with a regression target and thus were rejected for IQA-DCNN.

Some further trials with the initialization of weights using Gaussian distribution method gave good results. But these results were observed to be dependent upon a specific selection of standard deviation value. Hence it was concluded that Gaussian distribution method can be used for the initialization of IQA-DCNN weights, but it requires some initial trials to find suitable values of standard deviation to ensure stable convergence of IQA-DCNN.

IQA-DCNN was also tested with batch normalization method. It was observed that though batch normalization results in very fast convergence, the proposed methods for the estimation of unique values of mean and standard deviation (computed across minibatches) failed at producing corresponding results during validation and testing phase. The validation and test performance were observed to improve only when fresh values of mean and standard deviation were computed across respective validation and test minibatches (that contain distorted images produced from same reference image). Since test images could not be expected to have minibatches, batch normalization was also rejected.

From considered weight initialization methods, **Gaussian distribution method** was selected for the initialization of IQA-DCNN weights.

4.2 Experimental setup

With training process design parameter already selected in section 4.1, in this section we describe the experimental setup that was used for selecting various design parameters of IQA-DCNN architecture.

Based on the selected training methods, we first describe the initialization of V-0 IQA-DCNN architecture (in subsection 4.2.1). Then we introduce the dataset (in subsection 4.2.2) that was utilized for the training, validation and testing of IQA-DCNN for the purpose of this Chapter. Next we move to data augmentation technique (in subsection 4.2.3) that was employed to artificially increase the size of IQA datasets. Finally, we describe the evaluation method (in subsection 4.2.4) used for the selection of architecture design parameters in the local feed of IQA-DCNN, and then later for exploring the possibility of adding an extra global feed in section 4.4.

4.2.1 Architecture initialization

In the starting of this Chapter, we defined a version zero (V-0) of IQA-DCNN architecture (shown in Figure 4.3) and explained its configuration. Then in section 4.1, we decided upon the training process design parameters and chose to train IQA-DCNN using only IQA dataset based supervised training towards a regression target. We further chose L2 norm as the cost function, and employed Nesterov momentum based minibatch gradient descent. And finally, we decided to initialize network weights using Gaussian weight initialization method.

Based on the selected parameters of the training process, and V-0 IQA-DCNN architecture, we further initialized minibatch size to 32 images per minibatch, which is not very large or small considering the size of IQA datasets. The biases in all layers were initialized to value 0.1 (adopted from [29]) which accelerates learning in early stages of training by providing positive inputs to ReLU activations functions. All

weights were initialized from Gaussian distribution of zero mean and standard deviation of 0.02 for Conv layers, 0.005 for FC1 and 0.05 for FC2 layer. These standard deviation values were obtained through initial trials and were found to fit IQA-DCNN into promoting smooth convergence. We further normalized IQA-DCNN targets between 0 and 1, that fits well with initialized weights.

We also used a data augmentation technique to artificially increase the size of our dataset, which is presented in subsection 4.2.3.

4.2.2 Dataset

In IQA task, the perceived quality of an image is predicted in terms of Mean Opinion Score (MOS). Thus in order to train IQA-DCNN to predict an image quality, set of example pairs of images and their MOS values were required. This requirement could be fulfilled through public datasets that performed subjective experiments on the set of images and made both images and their respective MOS available in the form of IQA datasets. We considered TID[41] dataset for the training, validation, and testing of IQA-DCNN. The choice of TID was motivated from its utilization in related work ([66] and [26]), and also because it is the biggest among available IQA dataset.

TID[41] is an image quality dataset that consists of 25 reference images distorted with 17 different distortion types at 4 degradation levels to provide 1700 distorted images in total. The 17 distortion types included in this dataset are: Additive Gaussian noise (WN), Additive noise in color components (WNC), Spatially correlated noise (SCN), Masked noise (MN), High frequency noise (HFN), Impulse noise (IN), Quantization noise (QN), Gaussian blur (BLUR), Image denoising (IDN), JPEG compression (JPEG), JPEG2000 compression (JPEG2K), JPEG transmission errors (JPEGTE), JPEG2000 transmission errors (JP2KTE), Non eccentricity pattern noise (NEPN), Local block-wise distortions of different intensity (LBD), Inten- sity shift (IS) and Contrast change (CC). TID dataset provides a Mean Opinion Score (MOS) for each distorted image. MOS varies in range [0,9] for images with lowest and highest visual quality respectively.

For IQA-DCNN we have used all 25 reference images (Including non-natural scene image) distorted with first 13 distortion types. To make our results comparable with other NR-IQA algorithms, last four distortion types (NEPN, LBD, IS and CC) were excluded following the setup of [66] and [26]. Reference images were also not included in either of the training, validation or test sets because of the same reason. Thus total 1300 distorted images were used from TID dataset.

4.2.3 Data Augmentation

As discussed in section 2.2.2.2, data augmentation is a good technique that helps tackle overfitting. Thus we considered an efficient method of augmenting data inspired from the data augmentation methods used in CNN's for object recognition tasks [29].

The reference images used in TID dataset are bigger than 227 x 227 input size of V-0 IQA-DCNN. This difference can be used for producing a large number of data augments of input images. For instance, all reference images in TID dataset are of

size 512 x 384, which makes it possible to extract $(512-227)*(384-227)=44,745$ different but overlapping data augments of size 227 x 227 from each image. Thus for total 1300 images, it is possible to have 581,68,500 (58.1 million) data augments, which is a large number in itself.

Data augmentation is beneficial for IQA-DCNN but storing all these data augments would require very large amount of disk space. One way to tackle this problem is to randomly take 227 x 227 crops of input image during training, which would result in different input patches in every epoch. Given large enough number of epochs, using this scheme, a CNN could train on all possible augments of a dataset. But there is a complication as minibatch gradient descent optimization require inputs in form of minibatches, and it is not possible to make minibatch of different sized images (as per general practice, a minibatch is constructed and processed as a multidimensional matrix, and it is not possible to form a multidimensional matrix with inputs of different dimensions).

It was also observed that if 64 evenly spaced overlapping patches of size 300 x 300 are extracted from an image of size 720 x 720 (720 being the largest dimension among images in TID and LIVE (LIVE dataset is used in Chapter 5), then all possible data augments of 720 x 720 image can be reproduced from the extracted 300 x 300 patches. This is also valid for any other image of the size smaller than 720 x 720. Thus to save disk space, 64 patches of size 300 x 300 were extracted from every image in the dataset, which was then given as an input to the IQA-DCNN that randomly takes a 227 x 227 crop as its input.

During training, all patches were assigned the quality label of the whole image. Further during the testing phase, 64 227 x 227 patches were directly extracted from the image dataset and were given as an input to IQA-DCNN without any further need of cropping. For this testing scheme, the final quality score of an image was obtained by taking an average of individual scores on 64 patches.

This data augmentation technique resulted in total $1300*64=83,200$ image patches from TID dataset.

It should be noted that the described data augmentation technique could only be used with the local feed of IQA-DCNN, and no data augmentation is possible with the input of its global feed.

4.2.4 Evaluation method

IQA-DCNN with different architecture design parameters was trained, validated and tested using TID dataset, using which, we further employed 5 fold cross validation (that is illustrated in Figure 4.5), as per which the 25 reference images were divided in 5 non-overlapping groups (folds) of 5 images each. Out of total 5 folds, 3 folds (60%) were used for training, 1 fold (20%) for validation and 1 fold (20%) for testing. All distorted version of a reference image were assigned to the fold to which the reference image was assigned, and to no other fold. Hence, in the cross-validation setting, the network was forced, at test time, to generalize over image contents that it had not seen in the training (or validation). In addition, the 5-fold cross-validation made sure that all contents were used for test at least once, thereby covering the whole dataset.

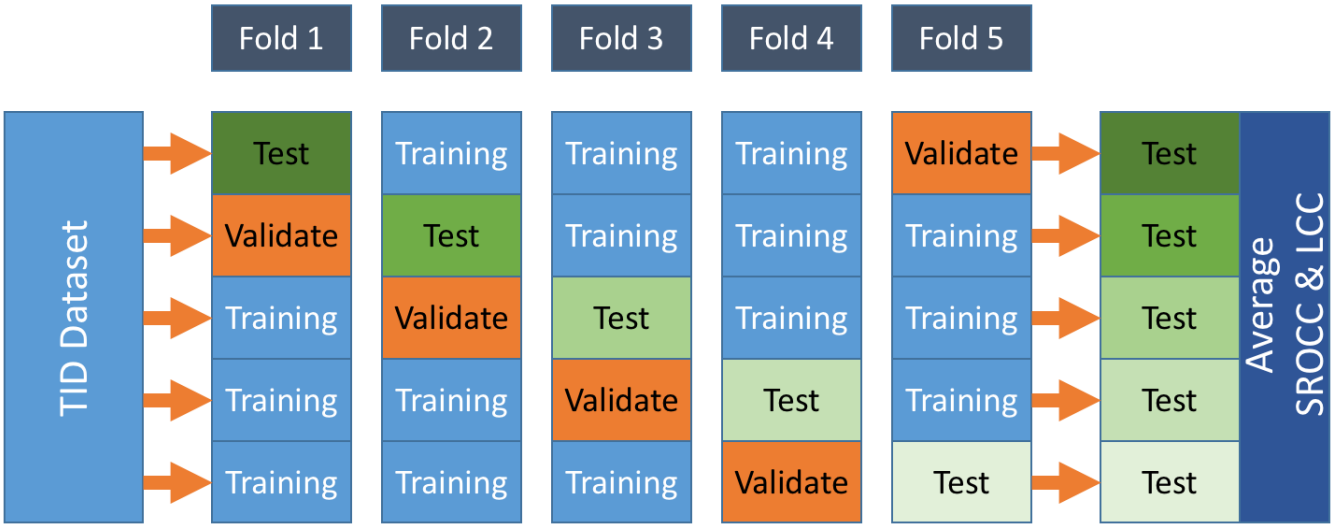


Figure 4.5: Illustration of five fold cross validation on TID dataset.

As per the recommendations of VQEG[46] and ITU[23], we employed two measures to evaluate the performance of each fold:

1. **Spearman Rank Order Correlation Coefficient (SROCC)**: It is correlation metric that measures how well a variable can be described as a monotonic function of another variable.

   .

2. **Linear Correlation Coefficient (LCC)**: It measures the linear dependencies between two variables.

These evaluation metrics were used to compute correlations between the desired output quality scores (provided by IQA datasets) and quality scores generated by CNN. Higher values of these correlations correspond to higher performance of CNN. The performance of various architecture design parameters were evaluated based on the average SROCC and LCC evaluation measures across the five test sets as per five fold cross-validations.

4.3 Architecture design parameters

In this section, we describe our selection of various architecture design parameters for IQA-DCNN.

As explained in section 2.2, the configuration and inclusion of different architecture elements could greatly influence the performance of a CNN. Thus in this section we explore a number of elements, namely data pre-processing units (subsection 4.3.1), local response normalization units (subsection 4.3.2), activation functions (subsection 4.3.3), convolutional layer and pooling units (subsection 4.3.4), fully connected layers (subsection 4.3.5), regularization (subsection 4.3.6), and minibatch size (subsection

	F1	F2	F3	F4	F5	Average
No-Pre	0.447	0.547	0.59	0.568	0.573	**0.545**
GCN	0.465	0.623	0.542	0.576	0.597	**0.561**
LCN	0.71	0.871	0.874	0.861	0.825	**0.828**

Table 4.1: SROCC: Performance evaluation of data pre-processing units on 5 test folds of TID

	F1	F2	F3	F4	F5	Average
No-Pre	0.477	0.542	0.608	0.642	0.589	**0.572**
GCN	0.54	0.655	0.602	0.632	0.63	**0.612**
LCN	0.758	0.883	0.887	0.882	0.841	**0.85**

Table 4.2: LCC: Performance evaluation of data pre-processing units on 5 test folds of TID

4.3.7), of IQA-DCNN design. The experimental setup used for the comparison of various design parameters is explained in section 4.2.

4.3.1 Data pre-processing

Data pre-processing is generally used with an aim to reform the input data such that the data could be utilized into accelerating the overall learning process [6]. Thus to inspect their advantages, we considered and compared three different architecture design of IQA-DCNN that respectively employ:

1. *No data pre-processing (No-Pre)* (as in V-0 architecture)
2. *Global contrast normalizing unit (GCN)*
3. *Local contrast normalization unit (LCN)*

GCN and LCN were selected because they are most commonly used in object recognition and IQA tasks [35][25][26]. Background information on GCN and LCN is presented in section 2.2.2.4.

It should be noted that for LCN (similar to BRISQUE, IQA-CNN, and IQA-CNN++) we used the local normalization window of size seven ($k = 7$).

The SROCC and LCC measures on five folds of TID dataset, obtained after the performance evaluation of these three architecture design, are respectively summarized in Table 4.1 and 4.2. From the average performance on all five folds under SROCC evaluation metric, it was observed that LCN performed around 26.7% better than GCN, and around 28.3% better than No-Pre. On LCC evaluation metric, LCN outperformed GCN and No-Pre by 23.8% and 27.8% margin respectively.

Clearly, the LCN unit is a valuable addition to IQA-DCNN, and thus it was included in its architecture. The updated version of IQA-DCNN, V-1, is shown in Figure 4.6.

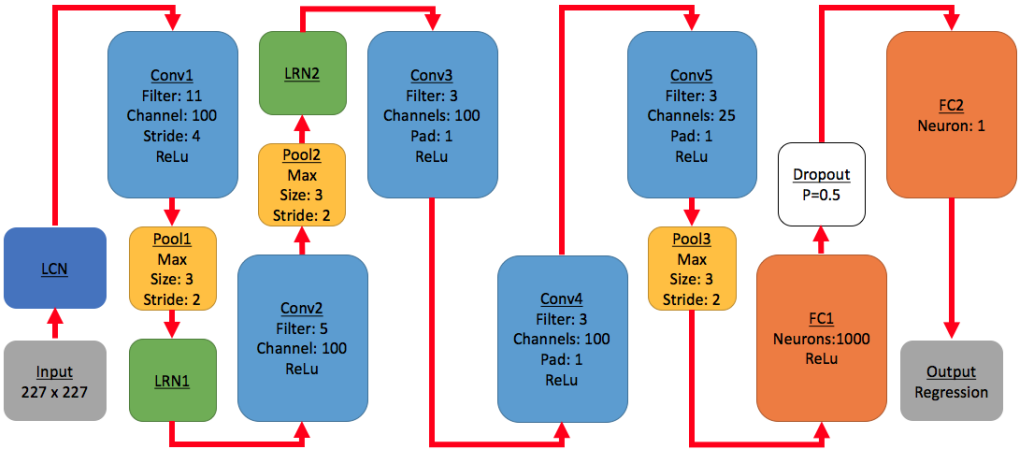


Figure 4.6: IQA-DCNN architecture: V-1

	F1	F2	F3	F4	F5	Average
LRN	0.71	0.871	0.874	0.861	0.825	**0.828**
No-LRN	0.788	0.866	0.881	0.839	0.844	**0.844**

Table 4.3: SROCC: Performance evaluation of LRN unit on 5 test folds of TID

	F1	F2	F3	F4	F5	Average
LRN	0.758	0.883	0.887	0.882	0.841	**0.85**
No-LRN	0.819	0.881	0.895	0.865	0.857	**0.863**

Table 4.4: LCC: Performance evaluation of LRN unit on 5 test folds of TID

4.3.2 Local response normalization

LRN unit was introduced in AlexNet and was claimed to aid generalization [29], but recent CNN architectures like ResNet [18] does not employ it as it is no longer considered very useful. As IQA-DCNN architecture was adopted from AlexNet, LRN units are used in V-1 architecture. Thus it is important to compute its usefulness, without which it only contributes to the increase in computational demand of the architecture design.

To test the usefulness of LRN unit is IQA-DCNN, we considered and compared two different architecture designs that respectively apply:

1. *Local response normalization (LRN) unit* (as in V-1 architecture)

2. *No local response normalization (No-LRN) unit*

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these two architecture design, are respectively summarized in Table 4.3 and 4.4. From the average performance on all five folds under SROCC evaluation metric, it was observed that No-LRN setting performed 1.6% better than

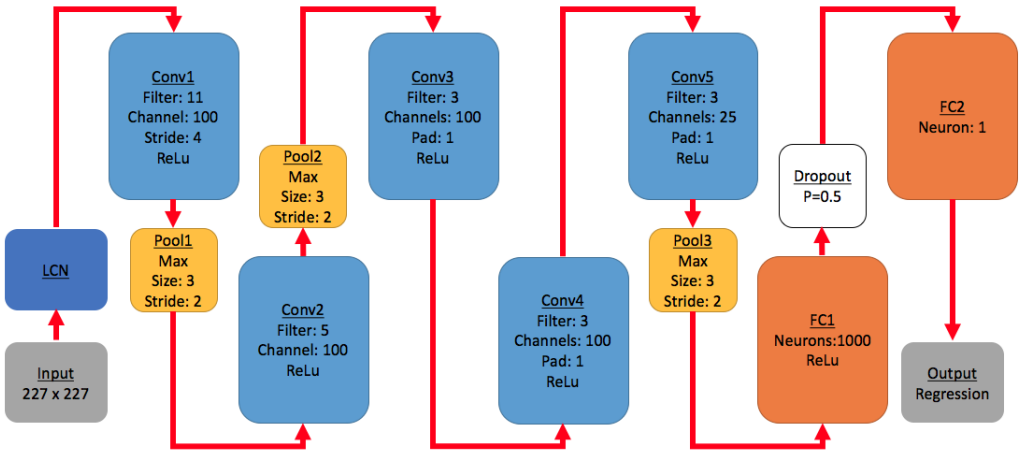


Figure 4.7: IQA-DCNN architecture: V-2

LRN setting. On LCC evaluation metric, No-LRN setting was observed to outperform LRN setting by 1.3% margin.

Since not employing any LRN unit also decreases the computational demand of the architecture, it was decided that LRN unit should be dropped from the current version of IQA-DCNN. The updated version of IQA-DCNN, V-2, is shown in Figure 4.7.

4.3.3 Activation Functions

As the activation of every layer of CNN is generally processed using an activation function, the selection of an appropriate activation function is vital for the proper functioning of IQA-DCNN. Thus we considered and compared two different architecture designs that use the different type of activation functions. We compared the architectures employing:

1. *ReLU* activation function in all layers except in FC2. (as in V-2 architecture)
2. *Maxout*[16] with FC1, and ReLU activation function with all other layers except FC2. Here we used $R = 3$ number of linear feature extractors with each Maxout function (see section 2.2.1.5).

   Since biggest advantage of Maxout unit comes from its ability to improve the model averaging performed by the dropout unit (as claimed in [16]), its utilization in FC1 layer (that is followed by dropout unit) is the best place in the IQA-DCNN architecture to record its advantages. Further three linear feature extractors were used as they were shown to deliver good performance in [8].

Tanh activation function was also considered first, but because of its poor performance reported in [6] due to its saturation problem, it was directly rejected. Background information on these employed activation functions is presented in section 2.2.1.5.

	F1	**F2**	**F3**	**F4**	**F5**	**Average**
ReLU	0.788	0.866	0.881	0.839	0.844	**0.844**
Maxout	0.743	0.859	0.887	0.861	0.884	**0.847**

Table 4.5: SROCC: Performance evaluation of activation functions on 5 test folds of TID

	F1	**F2**	**F3**	**F4**	**F5**	**Average**
ReLU	0.819	0.881	0.895	0.865	0.857	**0.863**
Maxout	0.783	0.822	0.906	0.89	0.892	**0.859**

Table 4.6: LCC: Performance evaluation of activation functions on 5 test folds of TID

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these two architecture design, are respectively summarized in Table 4.3 and 4.4. From the average performance on all five folds under both SROCC and LCC evaluation metrics, it was observed that Maxout did not provide any significant advantage over ReLU activation function. The reported results are very closely similar to each other as Maxout outperformed ReLU by 0.02% under SROCC metric, and ReLU outperformed Maxout by 0.4% under LCC metric.

Thus it was decided to not use the Maxout activation function as it does not provide any noticeable advantage over ReLU. Hence version V-2 of IQA-DCNN remains unchanged.

4.3.4 Convolutional layer and pooling unit

In this section, we present the results of the exploration of different design parameters related to the convolutional layers and pooling units, which are the building blocks of the CNN architecture. We start the exploration from the interface of Conv and FC layers, then we move to the exploration of the number of convolutional layers (depth), then to the type of pooling layers employed, and finally to the number of channels used in Conv layers. Background information about Conv layer and Pool unit is presented in section 2.2.

4.3.4.1 Interface of convolutional layers and fully connected layers

We considered and compared three different architecture designs that employ a different number of channels in the Conv5 layer, i.e., the Conv layer present at the interface of Conv and FC layers. The number of channels in the Conv5 layer is of significant importance. As Conv5 layer is followed by FC1 layers (as per V-2 architecture), each output of the Conv5 layer results in the increase of 1000 learnable parameters (corresponding to 1000 neurons in FC1 layer). Thus the number of channels of Conv5 were more carefully selected in comparison to other Conv layers. We compared the architectures employing:

1. *Conv5 channels = 25* (as in V-2 architecture)
2. *Conv5 channels = 50*

	F1	F2	F3	F4	F5	Average
Conv5 channels = 25	0.788	0.866	0.881	0.839	0.844	**0.844**
Conv5 channels = 50	0.756	0.877	0.896	0.859	0.875	**0.853**
Conv5 channels = 64	0.783	0.871	0.853	0.844	0.862	**0.843**

Table 4.7: SROCC: Performance evaluation of number of channels of Conv5 layer (which is present at the interface of Conv layers and FC layers) on 5 test folds of TID

	F1	F2	F3	F4	F5	Average
Conv5 Channels = 25	0.819	0.881	0.895	0.865	0.857	**0.863**
Conv5 Channels = 50	0.803	0.893	0.911	0.879	0.888	**0.875**
Conv5 Channels = 64	0.813	0.889	0.881	0.866	0.875	**0.865**

Table 4.8: LCC: Performance evaluation of number of channels of Conv5 layer (which is present at the interface of Conv layers and FC layers) on 5 test folds of TID

3. *Conv5 channels = 64*

Initially, only these three architecture designs were considered, and the plan was to explore further configurations if performance peak was not identified within the considered configurations.

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these three architecture design, are respectively summarized in Table 4.7 and 4.8. From the average performance on all five folds under SROCC evaluation metric, it was observed that Conv5 layer with 50 channels performed around 1% better than 25 and 64 channel setting. On LCC evaluation metric, 50 channel setting performed around 1.2% better than 25 channel setting, and around 1% better than 64 channel setting.

Hence the IQA-DCNN architecture was updated to the version V-3 and is shown in Table 4.8.

4.3.4.2 Number of convolutional layers

Since the number of Conv layers decide the depth of IQA-DCNN, we considered and compared three different architecture designs that use the different number of convolutional layers to evaluate the required depth of architecture to perform the NR-IQA task. We compared architectures employing:

1. *Conv layers = 4*

2. *Conv layers = 5* (as in V-3 architecture)

3. *Conv layers = 6*

Initially, only these three architecture designs were considered, and the plan was to explore further configurations if performance peak was not identified within the considered configurations.

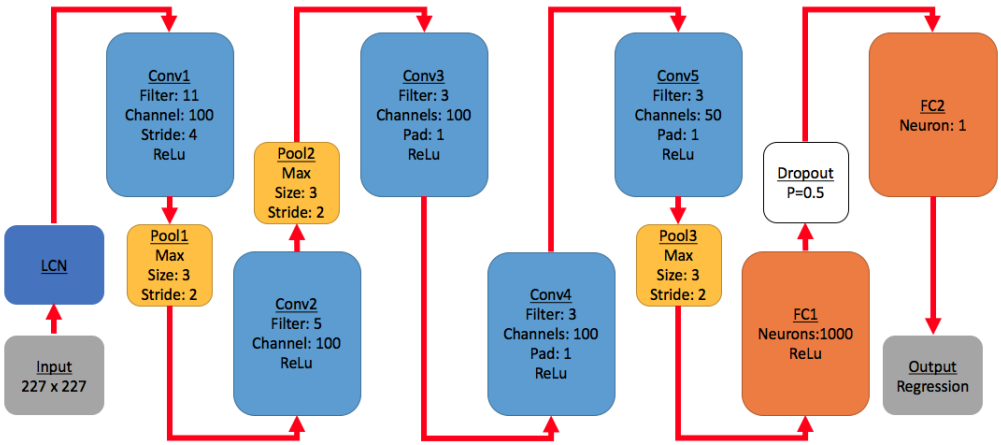


Figure 4.8: IQA-DCNN architecture: V-3

	F1	F2	F3	F4	F5	Average
Conv layers = 4	0.679	0.86	0.86	0.834	0.863	**0.819**
Conv layers = 5	0.756	0.877	0.896	0.859	0.875	**0.853**
Conv layers = 6	0.767	0.854	0.884	0.821	0.854	**0.836**

Table 4.9: SROCC: Performance evaluation of depth of convolutional layers on 5 test folds of TID

	F1	F2	F3	F4	F5	Average
Conv layers = 4	0.736	0.877	0.88	0.859	0.877	**0.846**
Conv layers = 5	0.803	0.893	0.911	0.879	0.888	**0.875**
Conv layers = 6	0.81	0.873	0.903	0.841	0.863	**0.858**

Table 4.10: LCC: Performance evaluation of depth of convolutional layers on 5 test folds of TID

It should be noted that for increasing the number of convolutional layers over V-3 architecture, Conv4 layer was duplicated because Conv4 did not change the shape of its input and duplicating it would require no further changes to be made elsewhere in the architecture. Also for decreasing the number of convolutional layers, the Conv4 layer was deleted following the same idea.

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these three architecture design, are respectively summarized in Table 4.9 and 4.10. From the average performance on all five folds under SROCC evaluation metric, it was observed that the depth of 5 Conv layers performed around 3.4% better than the depth of 4, and around 1.7% better than the depth of 6. On LCC evaluation metric, depth of 5 performed around 2.9% better than the depth of 4, and around 1.7% better than the depth of 6.

Hence the V-3 architecture was not updated.

	F1	F2	F3	F4	F5	Average
Overlapping Max pooling	0.756	0.877	0.896	0.859	0.875	**0.853**
Overlapping average pooling	0.698	0.83	0.838	0.802	0.841	**0.802**
Non-overlapping Max pooling	0.705	0.849	0.882	0.779	0.833	**0.81**
Non-overlapping average pooling	0.658	0.841	0.793	0.858	0.844	**0.799**
No pooling	0.653	0.801	0.8	0.78	0.796	**0.766**
Pooling scheme of IQA-CNN	0.71	0.784	0.809	0.806	0.832	**0.788**

Table 4.11: SROCC: Performance evaluation of pooling units on 5 test folds of TID

4.3.4.3 Pooling

Different types of pooling units have been reported in the literature. To select a proper pooling scheme for IQA-DCNN, we considered and compared six different architecture designs that use different types of pooling units. We compared architectures employing:

1. *Overlapping Max pooling* (as in V-3 architecture): A pooling unit that uses the stride of 2 and max pooling filters of size 3.

2. *Overlapping Average pooling*: A pooling unit that uses strides of 2, and average pooling filters of size 3.

3. *Non-overlapping Max pooling*: A pooling unit that uses strides of 2, and average pooling filters of size 2.

4. *Non-overlapping Average pooling*: A pooling unit that uses strides of 2, and average pooling filters of size 2.

5. *No pooling*: Inspired from the results reported in [7], instead of using separate pooling units, here we increased the stride values in the convolutional layers to decrease their output dimensionality. In this architecture design, Conv1 layer used the stride of 8, and Conv2 and Conv5 layers used the stride of 2. No pooling unit was used in this architecture design.

6. *Pooling scheme of IQA-CNN*: In this architecture design, we adopted the pooling scheme used in IQA-CNN[25] and IQA-CNN++[26]. Here we used average max pooling units after Conv1 and Conv2 layers, and every channel of the Conv5 layer was down-sampled to one max and one min value, which was then fed to the FC1 layer. Following the same setup as IQA-CNN and IQA-CNN++, we also did not use any activation function in any of the convolutional layers.

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these six architecture design, are respectively summarized in Table 4.11 and 4.12. From the average performance on all five folds under SROCC and LCC evaluation metrics, it was observed that overlapping max pooling units performed best among other types of pooling unit. Overlapping Max pooling performed 4.3% and 3.8% better than non-overlapping Max pooling (Second best performing pooling layer) on SROCC and LCC evaluation metrics respectively.

	F1	F2	F3	F4	F5	Average
Overlapping Max pooling	0.803	0.893	0.911	0.879	0.888	**0.875**
Overlapping average pooling	0.753	0.848	0.866	0.827	0.857	**0.83**
Non-overlapping Max pooling	0.76	0.864	0.894	0.818	0.848	**0.837**
Non-overlapping average pooling	0.706	0.853	0.837	0.877	0.859	**0.826**
No pooling	0.712	0.829	0.803	0.831	0.824	**0.799**
Pooling scheme of IQA-CNN	0.746	0.805	0.822	0.840	0.846	**0.812**

Table 4.12: LCC: Performance evaluation of pooling units on 5 test folds of TID

Since overlapping max pooling was already present V-3 architecture, IQA-DCNN architecture was not updated.

4.3.4.4 Number of Conv channels

Higher number of Conv channels could increase the learnable parameters in IQA-DCNN without providing any significant improvement in performance. Thus to select proper number of Conv channels, we considered and compared six different architecture designs that use the different number of channels in Conv layers. We compared architectures with Conv layers employing:

1. *Conv Channels = 8*
2. *Conv Channels = 16*
3. *Conv Channels = 64*
4. *Conv Channels = 100* (as in V-3 architecture)
5. *Conv Channels = 128*

Initially only these six architecture designs were considered and the plan was to explore further configurations if performance peak was not identified within the considered configurations. It should be noted that the number of channels of the Conv5 layer were fixed to 50 based on the result of a previous experiment (see section 4.3.4.1). So in case the employed Conv channels, as per above mentioned design parameters, were greater than 50, the channels of the Conv5 layer were fixed to 50, otherwise channels of Conv5 layer were also changed.

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these six architecture design, are respectively summarized in Table 4.13 and 4.14. From the average performance on all five folds under SROCC and LCC evaluation metrics, it was observed that architectures with 64 and 100 Conv channels deliver almost similar performance that is over 1% better in comparison to other architecture designs. Thus we decided to use the architecture design with 64 Conv channels as it contains less number of learnable parameters than the current V-3 architecture.

Hence V-3 architecture was updated to V-4, which is shown in Figure 4.9

	F1	F2	F3	F4	F5	Average
Conv Channels = 8	0.713	0.791	0.851	0.792	0.826	**0.795**
Conv Channels = 16	0.768	0.83	0.883	0.844	0.864	**0.838**
Conv Channels = 64	0.777	0.843	0.894	0.882	0.887	**0.857**
Conv Channels = 100	0.756	0.877	0.896	0.859	0.875	**0.853**
Conv Channels = 128	0.741	0.871	0.88	0.849	0.873	**0.843**

Table 4.13: SROCC: Performance evaluation of number of channels in convolutional layers on 5 test folds of TID

	F1	F2	F3	F4	F5	Average
Conv Channels = 8	0.788	0.832	0.843	0.821	0.854	**0.828**
Conv Channels = 16	0.806	0.863	0.893	0.869	0.873	**0.861**
Conv Channels = 64	0.806	0.865	0.909	0.894	0.895	**0.874**
Conv Channels = 100	0.803	0.893	0.911	0.879	0.888	**0.875**
Conv Channels = 128	0.788	0.889	0.898	0.873	0.883	**0.866**

Table 4.14: LCC: Performance evaluation of number of channels in convolutional layers on 5 test folds of TID

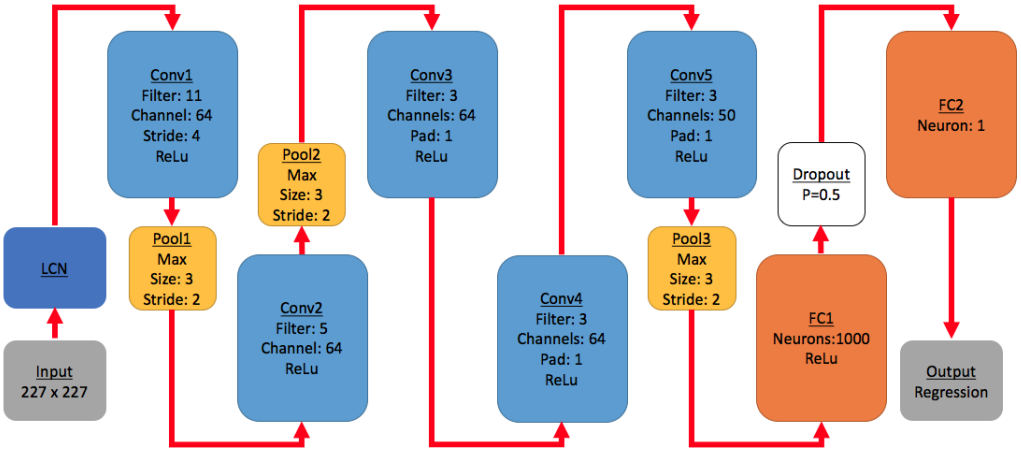


Figure 4.9: IQA-DCNN architecture: V-4

	F1	**F2**	**F3**	**F4**	**F5**	**Average**
FC layers = 1	0.732	0.879	0.886	0.89	0.862	**0.85**
FC layers = 2	0.756	0.877	0.896	0.859	0.875	**0.853**
FC layers = 3	0.66	0.739	0.876	0.818	0.802	**0.779**

Table 4.15: SROCC: Performance evaluation of depth of fully connected layers on 5 test folds of TID

	F1	**F2**	**F3**	**F4**	**F5**	**Average**
FC layers = 1	0.771	0.896	0.9	0.904	0.872	**0.869**
FC layers = 2	0.806	0.865	0.909	0.894	0.895	**0.874**
FC layers = 3	0.695	0.773	0.889	0.858	0.829	**0.809**

Table 4.16: LCC: Performance evaluation of depth of fully connected layers on 5 test folds of TID

4.3.5 Fully connected layers

FC layers result in the biggest increase in learnable parameters among other components of CNN, and their configuration should be carefully selected to control the number of learnable parameters in a CNN architecture. Thus we considered and compared three different architecture designs that use the different number of fully connected layers. We compared architectures employing:

1. *FC layers = 1*: In this architecture design, FC1 layer was removed. Since dropout unit was applied on FC1 layer, removing FC1 layer also removed the dropout regularization.

2. *FC layers = 2* (V-4 architecture)

3. *FC layers = 3*: In this architecture design, FC1 layer was duplicated (without dropout) and added between Conv5 and FC1.

Initially only these three architecture designs were considered and the plan was to explore further configurations if performance peak was not identified within these considered configurations.

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these three architecture design, are respectively summarized in Table 4.15 and 4.16. From the average performance on all five folds under SROCC and LCC evaluation metrics, it was observed that architecture with 3 FC layers performed worse among all three architecture designs. Further 2 FC layers performed 0.3% better on SROCC metric and 0.5% better on LCC metric.

It was noticed that using 2 FC layers add 1.26 million learnable parameters to the architecture while providing the small increase in the overall performance. Thus it was decided that the architecture design with 1 FC layer should be adopted. Hence IQA-DCNN architecture was updated to the version V-5 that is shown in Figure 4.10.

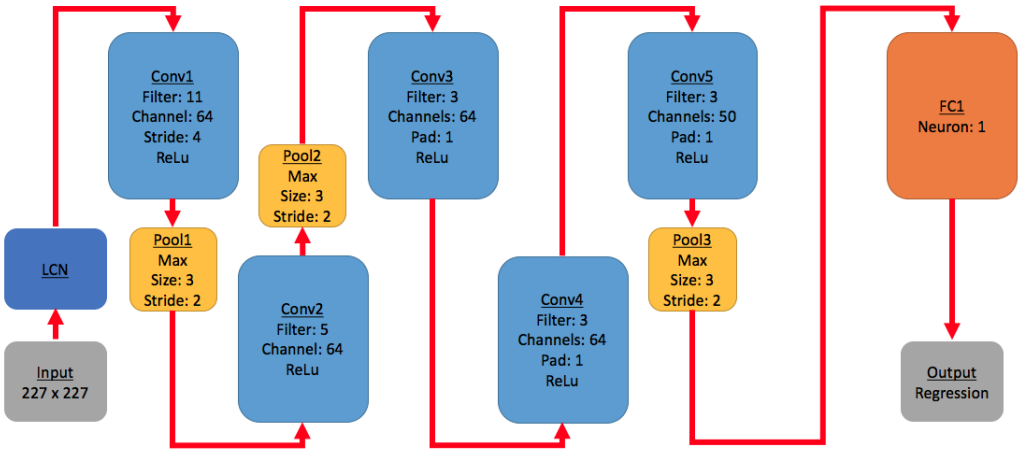


Figure 4.10: IQA-DCNN architecture: V-5

4.3.6 Regularization

After exploring and fixing various design parameters related to the physical structure of the IQA-DCNN architecture, we advance to the exploration of regularization techniques that could benefit the overall performance of IQA-DCNN by preventing overfitting. We considered and compared four different architecture designs that use different regularization method as follows:

1. *No regularization* (as in V-5 architecture)
2. *Dropout unit*: with dropout probability = 0.5 was applied to the output of Conv5 layer
3. *L1 regularization*: $\lambda = 0.00001$ and 0.000001 was chosen for L1 regulation. This value was selected through some initial trials.
4. *L2 regularization*: $\lambda = 0.005$ and 0005 was chosen for L1 regulation. This value was selected through some initial trials.

These three regularization techniques were employed because they are very common and widely used techniques in the field of neural networks [6]. Detailed background information on regularization is given in section 2.

We considered only one dropout unit in the starting, and the plan was to explore it further in case better results were obtained through this setting. Furthermore, we considered dropout probability of 0.5 because it is a commonly used value in many CNN architectures. [29][54][18][57][25][26] Since the high values of λ prevent the network from converging in the case of both L1 and L2 regularization, the values of λ were selected by first presetting them to a very high value and then by gradually decreasing it by a factor of 10. The first two encountered values of λ that resulted in the convergence of IQA-DCNN were then employed for the purpose of this section. This way we were able to select $\lambda = 0.00001$ and 0.000001 for L1 regularization and

	F1	**F2**	**F3**	**F4**	**F5**	**Average**
No regularization	0.732	0.879	0.886	0.89	0.862	**0.85**
Dropout	0.743	0.854	0.891	0.891	0.843	**0.844**
L1 with $\lambda = 0.00001$	0.716	0.844	0.895	0.87	0.853	**0.836**
L1 with $\lambda = 0.000001$	0.681	0.885	0.904	0.899	0.904	**0.855**
L2 with $\lambda = 0.005$	0.56	0.828	0.90	0.879	0.841	**0.802**
L2 with $\lambda = 0.0005$	0.713	0.89	0.902	0.918	0.879	**0.86**

Table 4.17: SROCC: Performance evaluation of regularization methods on 5 test folds of TID

	F1	**F2**	**F3**	**F4**	**F5**	**Average**
No regularization	0.771	0.896	0.9	0.904	0.872	**0.869**
Dropout	0.784	0.872	0.904	0.907	0.851	**0.864**
L1 with $\lambda = 0.00001$	0.768	0.86	0.901	0.884	0.86	**0.855**
L1 with $\lambda = 0.000001$	0.728	0.884	0.907	0.914	0.907	**0.868**
L2 with $\lambda = 0.005$	0.533	0.846	0.901	0.891	0.85	**0.804**
L2 with $\lambda = 0.0005$	0.768	0.899	0.912	0.928	0.888	**0.879**

Table 4.18: LCC: Performance evaluation of regularization methods on 5 test folds of TID

$\lambda = 0.005$ and 0.0005 for L2 regularization.

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these four architecture design, are respectively summarized in Table 4.17 and 4.18. From the average performance on all five folds under SROCC and LCC evaluation metrics, it was observed that the architecture with L2 regularization using $\lambda = 0.0005$ performed better than the other architecture designs.

Thus L2 regularization was adopted to the current V-5 IQA-DCNN architecture.

4.3.7 Minibatch size

The minibatch size could be altered to induce different proportions of noise into the CNN architecture. This induced noise could be beneficial in preventing network from converging to local minima, but if this noise becomes too large, it could negatively affect the CNN by preventing it from converging. Thus appropriate minibatch size should be selected for proper functioning of a CNN, for which we considered and compared two minibatch sizes for IQA-DCNN.

1. *Minibatch size = 16*
2. *Minibatch size = 32* (as in V-5 architecture)

We also considered minibatch size smaller than 16, but decreasing the size below 16 prevents IQA-DCNN from converging. The reason behind it could be that the level of noise in the architecture could become very high which ultimately prevents the network from converging.

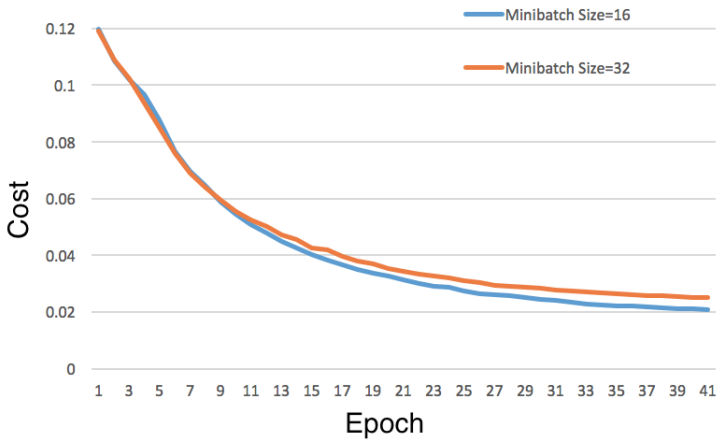


Figure 4.11: Training cost (L2 norm loss) vs Epoch graph: Comparison of convergence speed between minibatch size of 16 and 32.

	F1	**F2**	**F3**	**F4**	**F5**	**Average**
Minibatch size = 16	0.736	0.894	0.909	0.909	0.883	**0.866**
Minibatch size = 32	0.713	0.89	0.902	0.918	0.879	**0.86**

Table 4.19: SROCC: Performance evaluation of minibatch sizes on 5 test folds of TID

	F1	**F2**	**F3**	**F4**	**F5**	**Average**
Minibatch size = 16	0.774	0.908	0.917	0.916	0.9	**0.883**
Minibatch size = 32	0.768	0.899	0.912	0.928	0.888	**0.879**

Table 4.20: LCC: Performance evaluation with global feed on 5 test folds of TID

We did not consider minibatch size larger than 32 because larger minibatch size is known to make the convergence slow because of less parameter updates [6]. Furthermore, minibatch size of 32 has already been observed to work well in previous experiments.

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of these two architecture design, are respectively summarized in Table 4.19 and 4.20. From the average performance on all five folds under SROCC evaluation metric, it was observed that minibatch size of 16 performed around 0.6% better than minibatch size of 32. Under LCC evaluation metric, minibatch of size 16 performed around 0.4% better than minibatch of size 32.

As shown in Figure 4.11, it was also observed that minibatch of size 16 resulted in lower training cost in comparison to minibatch of size 32.

From the observations of these experiments, we selected minibatch of size 16 to be used for IQA-DCNN because it resulted in lower training cost while providing the overall performance improvement of IQA-DCNN. Thus minibatch of size 16 was adopted to V-5 architecture.

4.3.8 Discussion

Various architecture design parameters were compared in section 4.3.1 to 4.3.7, and the V-0 architecture of IQA-DCNN was updated to V-5 based on the outcome of these sections.

We started the design space exploration with section 4.3.1 in which we compared LCN and GCN data pre-processing methods. LCN was observed to be the best performing method and thus was employed in the IQA-DCNN design. The possible explanation behind the superior performance of LCN could be the decorrelating effect that it is claimed to have on input data [48]. Since decorrelated input data is easier to learn from, the network could have exhibited superior performance.

Then in section 4.3.2, we computed the usefulness of LRN unit and found that IQA-DCNN performed similarly well in case LRN unit is not employed. Thus LRN unit was dropped from IQA-DCNN architecture. This could also be due to the presence of LCN unit that could complement the functioning of LRN unit as they are very similar in their operation.

We then compared ReLU activation function with Maxout and found that Maxout did not provide any advantage over ReLU activation function. Since Maxout further increases the number of learnable parameters in the network, it was not adopted in IQA-DCNN architecture. Maxout unit is known to be advantageous because it complements dropout unit [16], but later in section 4.3.6 dropout unit was observed to be a poorly performing design parameter for IQA-DCNN, which could also have taken away the advantage from Maxout unit.

Later in section 4.3.4, we compared various configurations of Conv layers and pool units. We started with setting the number of channels for Conv5, i.e., the layer at the interface of Conv layers and FC layers, to 50. For the other Conv, a number of channels equal to 64 was found to be optimal. Small drops in performance were observed with Conv layers employing widely different number of channels, which could point to the fact that performance improvement of IQA-DCNN might have started to saturate with number of channels being as small as 16. In fact, no improvement was recorded when using a number of channel greater than 64. Rather, a slight inflexion in performance was notice, which could either be due to overfitting or simply due to slow convergence resulted by the increase in learnable parameters.

In the same section, we also explored the depth of Conv layers and it was observed that increasing or decreasing the number of conv layers with respect to the original number set to 5 would lead to lower performance. The decrease in depth could have resulted in reduced performance because of the absence of high-level features that are present in deeper architectures, and loss in performance due to further increase in depth of the Conv layers could be due to the increased difficulty of the training process to promote convergence.

We also compared various pooling schemes and found overlapping max pooling to perform best among others. It was claimed in [29] that overlapping max-pooling also helps in regularizing the architecture, which could be a possible reason for the observed superior performance.

We then explored the depth of fully connected layers in section 4.3.5 and found that IQA-DCNN almost gave similar performance for depth of one and two FC layer. Since removing a FC layer also removes a large number of learnable parameters, the

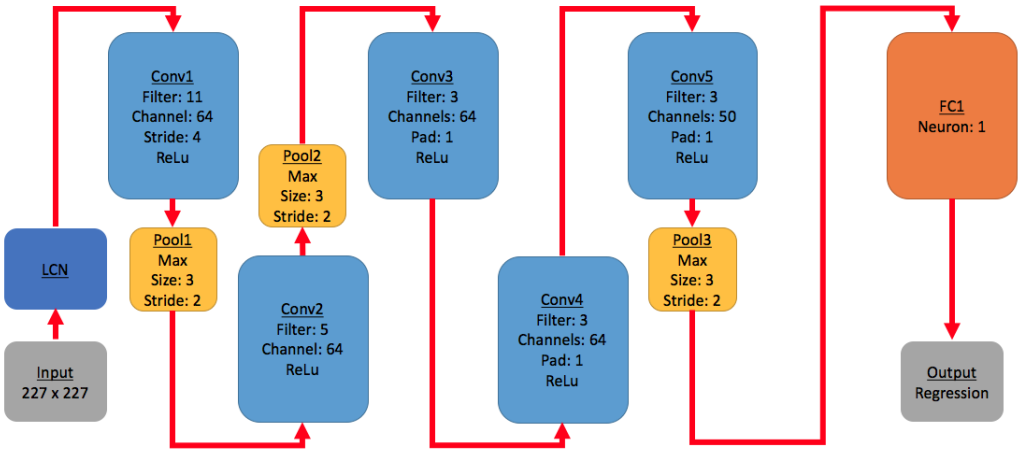


Figure 4.12: IQA-DCNN architecture

configuration with one FC layer was selected for IQA-DCNN. This could also mean that features learned by Conv layers of IQA-DCNN are very good and relevant to the task of IQA, and thus does not require an extra FC layers to deliver similar performance.

Further in section 4.3.6, we explored different types of regularization and observed that L2 regularization with $\lambda = 0.0005$ performed best among all other settings. It was further observed that the performance of this setting was very similar to the setting in which no regularization was employed, and to the setting in which L1 regularization was employed with $\lambda = 0.000001$. The reason behind this could be that selected values of λ for L1 and L2 regularization are very small which make these settings very similar to the setting in which no regularization was employed.

At last in section 4.3.7, we compared different minibatch sizes and observed that reducing the size of a minibatch to 16 resulted in faster convergence and helped in achieving lower training cost, both of which could be due to the induced noise by small sized minibatches and more number of parameter updates in comparison to bigger minibatches.

The IQA-DCNN architecture obtained from this design space exploration is presented in Figure 4.12. IQA-DCNN architecture contains total 229,717 learnable parameters.

As already mentioned earlier, it was observed from Table 4.13 and 4.14 that employing 16 Conv channels did not result in big performance loss in comparison of 64 Conv channel configuration. This could be the result of saturating improvement in the performance of IQA-DCNN with the different number of employed Conv channels. Along with this, the architecture with 16 Conv channels have much less learnable parameters than the architecture with 64 Conv channels. Thus we also propose another architecture design, named as IQA-DCNN-s (shown in Figure 4.13), for minimizing computational complexity. IQA-DCNN-s architecture contains total 19,601 learnable parameters which is over 90% less than IQA-DCNN.

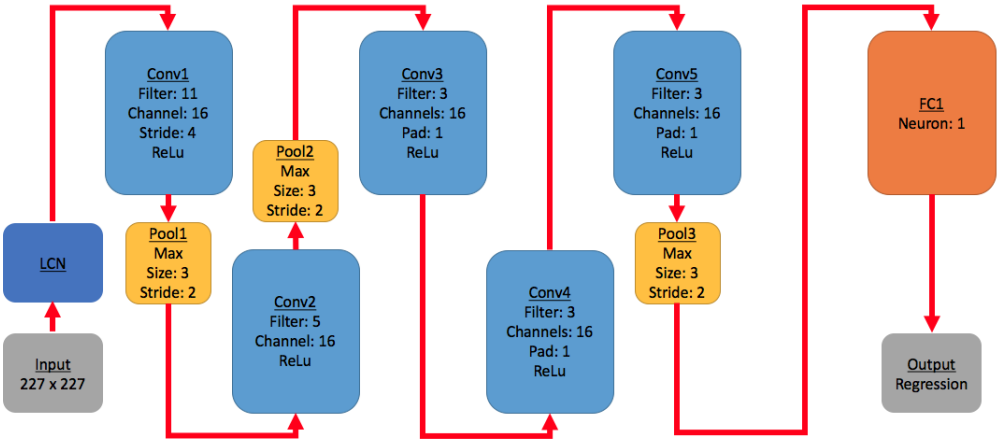


Figure 4.13: IQA-DCNN-s architecture

4.4 Global Feed

After selecting various training process design parameters and CNN architecture design parameters, we further explored the possibility of adding a global feed to the existing single feed architecture that operates on local view (crops) of input images. Since among previously selected IQA-DCNN and IQA-DCNN-s architectures, IQA-DCNN is a better performing architecture, it was considered to be extended by adding an extra global feed. The plan was to extend IQA-DCNN-s as well in case global feed was observed to perform well with IQA-DCNN.

As mentioned in the introduction of this Chapter, instead of configuring the global feed from scratch, we decided to directly duplicate the local feed of IQA-DCNN to form a global feed. This decision was motivated from the RAPID [33] architecture that contains identical local and global feeds and has been shown to provide superior performance. This decision also helped in restraining our design space.

Thus a two feed architecture was created by duplicating the local feed of IQA-DCNN. The obtained architecture is shown in Figure 4.14. The local feed of this architecture operates on image crops, while the global feed operates on image warps (similar to RAPID) that are created by simply resizing the input image. Further the image warps and image crops were created from the same image during training, validation and testing phase.

To train this two feed architecture, we adopted the same training setup as IQA-DCNN. We further tried training the two feed architecture from scratch with weights initialized using Gaussian distribution method. But since there is no data augmentation on global feed (which is not possible), this setting was observed to result in high levels of overfitting. It is because the number of image warps on global feed were much smaller than possible data augment on local feed, due to which the global feed tends to overfit on its input examples, which ultimately resulted in the whole network to overfit.

To overcome this, we further considered an alternative approach as per which we

	F1	F2	F3	F4	F5	Average
Only Local	0.736	0.894	0.909	0.909	0.883	**0.866**
Local + Global	0.709	0.883	0.885	0.874	0.864	**0.843**

Table 4.21: SROCC: Performance evaluation with global feed on 5 test folds of TID

	F1	F2	F3	F4	F5	Average
Only Local	0.774	0.908	0.917	0.916	0.9	**0.883**
Local + Global	0.762	0.885	0.889	0.872	0.881	**0.858**

Table 4.22: LCC: Performance evaluation with global feed on 5 test folds of TID

pre-train the local feed (IQA-DCNN architecture) until convergence and then adopt its learnable parameters to the global feed. The two feed architecture was then fine-tuned with re-initialized FC layers, and with lower learning rate set for Conv layers of both feeds and relatively higher learning rate set for FC layers. The idea here was to prevent overfitting by slowly updating the learnable parameters of global feed, while rapidly learning the learnable parameters of the FC layers for fast convergence.

The SROCC and LCC measures on five test folds of TID dataset, obtained after the performance evaluation of local feed architecture (IQA-DCNN) and the two feed architecture are respectively summarized in Table 4.21 and 4.22. From the average performance on all five folds under SROCC evaluation metric, it was observed that adding a global feed did not result in any improvement over single local feed.

Figure 4.15 shows the resulting SROCC measure on validation set with respect to epochs during the training of two feed architecture. From this plot, it was observed that the network jumps to high SROCC reading in its very first epoch (which is recorded in Table 4.21 and 4.22) of fine-tuning but then rapidly drops in proceeding epochs of fine-tuning. We believe that the initially observed gain in performance was because of the pre-trained weights of local feed, after which the global feed slowly kicks which results in the overfitting and rapid drop in the performance of the architecture. This clearly indicates that even this scheme was incapable at avoiding overfitting of two feed architecture.

Thus based on these observations, we decided to not use extra global feed over IQA-DCNN or IQA-DCNN-s architectures because it did not result in any performance gain.

4.5 Chapter conclusion

In this Chapter we have presented the design space exploration of CNN for the application of NR-IQA tasks. With an aim to overcome the limitations of existing NR-IQA based approaches, we envisioned a two feed CNN architecture to process the local and global information of given images in NR-IQA task. We started by selecting various training process design parameters, which were used for the training of CNN design. Then we presented the experimental setup using which various architecture design parameters were selected for local feed of CNN architecture. Later using the same experimental setup and selected configurations of local feed, we also explored the pos-

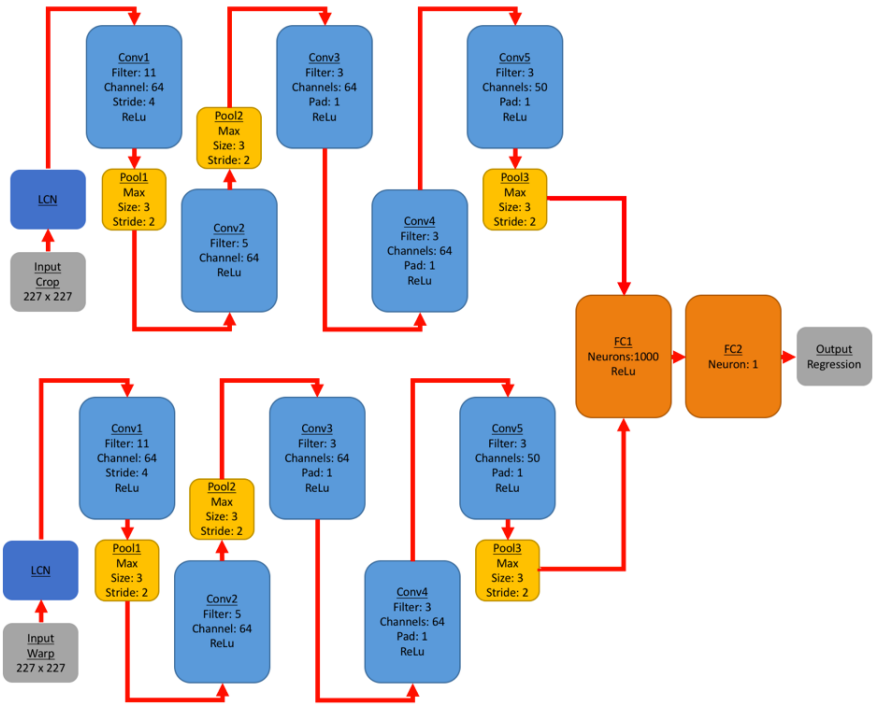


Figure 4.14: IQA-DCNN architecture with global feed

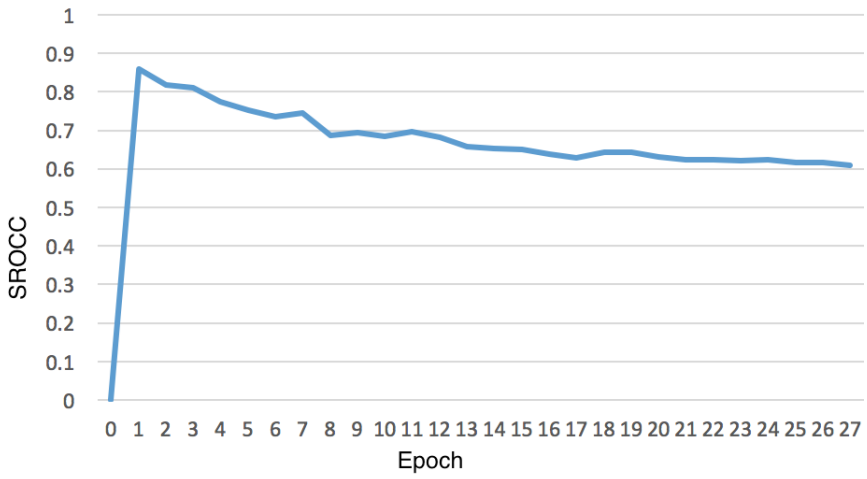


Figure 4.15: Plot: SROCC measure on validation dataset when fine-tuned with local and global feed architecture.

sibility of adding a global feed to the architecture. But based on the outcome of the two utilized methods that resulted in overfitting, we decided not to adopt it in the final version of the architecture design.

Based on the outcome of this Chapter, we propose two CNN architectures, IQA-DCNN (shown in Figure 4.12) for prediction accuracy, and IQA-DCNN-s (with 90% less learnable parameters that IQA-DCNN, as shown in Figure 4.13) for minimizing computational demand.

Chapter 5

Performance Evaluation

In chapter 4, we proposed two CNN architecture designs for the task of no-reference image quality assessment: IQA-DCNN and IQA-DCNN-s. These architectures are respectively shown in Figure 4.12 and 4.13.

In this chapter, we conduct the performance evaluation of IQA-DCNN and IQA-DCNN-s. We start with section 5.1 in which we describe the employed evaluation method. Using this evaluation method, in section 5.2, we present the obtained results and compare them among existing FR-IQA and NR-IQA algorithms.

FR-IQA algorithms (described in section 2) such as PSNR, SSIM[60] and FSIM[74], utilize reference images while predicting the quality of an image and thus are easier to achieve and usually better at performance than NR-IQA algorithms. Among NR-IQA algorithms, DIIVINE[37], BLIINDS-II[49] and BRISQUE[35] are NSS based approaches that use handcrafted features for IQA. On the other hand CORNIA[66], IQA-CNN[25] and IQA-CNN++[26] are feature learning based approaches that are generally more efficient and better performing. Among these feature learning based approaches, IQA-CNN and IQA-CNN++ are the CNN based approaches that respectively employ one and three Conv layers in their implementation.

After conducting the performance comparison with these algorithms, in section 5.3, we describe the robustness of IQA-DCNN and IQA-DCNN-s in assessing the perceptual quality of image patches. Finally in section 5.4, we give present the conclusions of this Chapter.

5.1 Evaluation method

This section presents the evaluation method used for the performance evaluation of IQA-DCNN and IQA-DCNN-s. The employed evaluation method was adopted from [25] and [26], where it was also used for the performance evaluation of CNN architecture (IQA-DCNN and IQA-DCNN++) trained for the application of NR-IQA.

For the performance evaluation of IQA-DCNN and IQA-DCNN-s, we made use of two datasets, TID[41] and LIVE[51]. The TID dataset and its usage in this thesis project was described in section 4.2.2.

LIVE is an image quality dataset that consists of 29 reference images distorted with 5 different distortion types at 7-8 degradation levels to provide 779 distorted images in

total. The 5 distortion types included in this dataset are: JP2k compression (JPEG2K), JPEG compression (JPEG), White Gaussian (WN), Gaussian blur (BLUR) and Fast Fading (FF). The LIVE dataset provides a Differential Mean Opinion Score (DMOS) for each distorted image. DMOS varies in range [0,100] for images with highest and lowest visual quality respectively.

In this thesis project, we have used all 29 reference images distorted with 4 distortion types. Reference images were also included in training, validation, and test sets. Thus total 982 images were used from LIVE dataset. Further, we used the data augmentation technique described in section 4.2.3, which resulted in total $982 * 64 = 62,848$ image patches from LIVE dataset to be used in a single epoch.

The evaluation setup using these two datasets is described below:

1. **LIVE**: To create training, validation and test sets, LIVE dataset was divided into 29 folds based on its 29 reference images. This counts to one fold containing one reference image and its distorted versions. Out of total 29 folds, 17 folds (58.6%) were used for training, 6 folds (20.7%) for validation and 6 folds (20.7%) for testing. These 29 folds were then randomly shuffled among training, validation and test sets to perform total 100 train-validation-test repetitions.

2. **TID**: To create training, validation and test sets, TID dataset was divided into 25 folds based on its 25 reference images. This counts to one fold containing distorted versions of only one reference image. Out of total 25 folds, 15 folds (60%) were used for training, 5 folds (20%) for validation and 5 folds (20%) for testing. These 25 folds were then randomly shuffled among training, validation and test sets to perform total 100 train-validation-test repetitions.

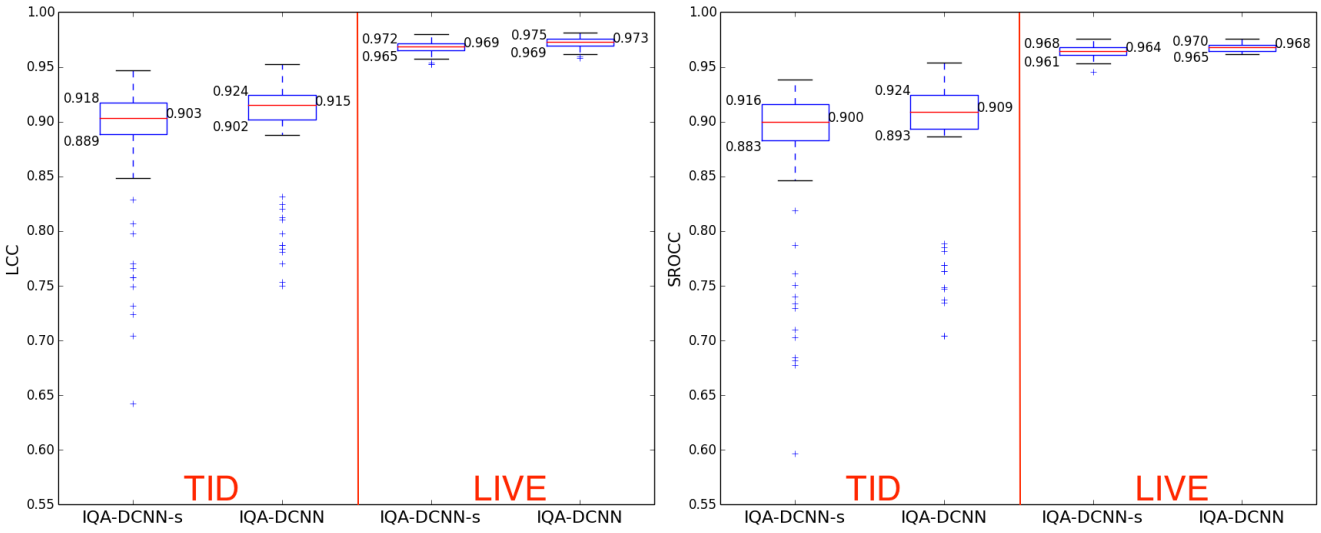
We used SROCC and LCC evaluation metrics (similar to evaluation method in design phase as described in section 4.2.4) to evaluate the performance of IQA-DCNN and IQA-DCNN-s on respective test sets and reported the average and median performance on all 100 repetitions for both LIVE and TID datasets.

Both average and median performance were reported because some of the existing NR-IQA algorithms report their performance on the average, and some other on the median of the number of employed repetitions. Thus reporting both average and median performances helped us in performing a fair comparison of results with both types of approaches.

5.2 Results

This section presents the results of performance evaluations of IQA-DCNN and IQA-DCNN-s, on LIVE and TID datasets, as obtained using the evaluation method that is described in section 5.1.

The SROCC and LCC evaluation measures in 100 train-validation-test repetitions of IQA-DCNN and IQA-DCNN-s on TID and LIVE dataset are shown using box plots in Figure 5.1b and 5.1a. IQA-DCNN was observed to be performing marginally better than IQA-DCNN-s. It was further observed that difference between the first and third



(a) LCC metric

(b) SROCC metric

Figure 5.1: Box plots of 100 repetitions of evaluation on LIVE and TID dataset, using (a) LCC and (b) SROCC measures of IQA-DCNN-s and IQA-DCNN.

	SROCC	**LCC**
LIVE	$1.9 * 10^{-08}$	10^{-06}
TID	0.125	0.063

Table 5.1: p-values from independent sample t-test to compare SROCC and LCC measures over 100 repetitions of IQA-DCNN and IQA-DCNN-s training on LIVE and TID datasets.

quartile of all box plots is quite small, which reflects the robustness of the reported results. Furthermore, this difference was observed to be smaller with evaluations on LIVE dataset in comparison to TID dataset reflecting superior performance on LIVE.

We also performed independent sample t-test to test if the performance with IQA-DCNN was significantly better than IQA-DCNN-s. The results of this test are presented in Table 5.1. It was observed that the p-values in the case of evaluations on LIVE dataset are very small, which implies that the reported mean of IQA-DCNN performance is significantly different from IQA-DCNN-s during evaluations on LIVE dataset. Whereas in case of evaluation on TID dataset, the reported p-values are greater than 0.05, which implies that the performance of IQA-DCNN is not very different from IQA-DCNN-s.

For evaluation on LIVE and TID datasets, Table 5.2 and 5.3 respectively compare the evaluation results of 100 train-validation-test repetitions of IQA-DCNN and IQA-DCNN-s with previous state of the art NR-IQA and FR-IQA algorithms (*Italicized*). The best performing result among NR-IQA methods is in bold. The second column in these tables points out if the corresponding algorithm computes median or average

Algorithm	Evaluation	SROCC	LCC
PSNR		0.866	0.856
SSIM [60]		0.913	0.906
FSIM [74]		0.964	0.960
DIIVINE [37]	Median	0.916	0.917
BLIINDS-II [49]	Median	0.931	0.930
BRISQUE [35]	Median	0.940	0.942
CORNIA [66]	Median	0.942	0.935
IQA-CNN [25]	Average	0.956	0.953
IQA-CNN++ [26]	Average	0.950	0.950
IQA-DCNN-s	Average	0.964	0.968
IQA-DCNN-s	Median	0.964	0.969
IQA-DCNN	Average	0.968	0.972
IQA-DCNN	Median	**0.968**	**0.973**

Table 5.2: Comparison of SROCC and LCC measure on LIVE dataset. *Italicized* are FR-IQA algorithms for reference.

over its multiple repetitions of performance evaluation.

It should be noted that we do not compare our performance evaluation on TID dataset with NSS based NR-IQA algorithms. The reason is that first of all these algorithms did not report their performance evaluation on TID dataset, and secondly, the evaluation results on the LIVE dataset (Table 5.2) clearly shows that these algorithms project inferior performance in comparison to feature learning based NR-IQA algorithms.

From these tables, it was observed that both IQA-DCNN and IQA-DCNN-s are the best-performing algorithms on SROCC and LCC evaluation metrics among all other NR-IQA algorithms. Further in evaluation on LIVE dataset, reported SROCC and LCC measures of IQA-DCNN and IQA-DCNN-s were also observed to be higher than all FR-IQA algorithms. Whereas on TID dataset, only FSIM algorithm was observed to be outperforming IQA-DCNN-s in both evaluation metrics, and IQA-DCNN in SROCC evaluation metrics.

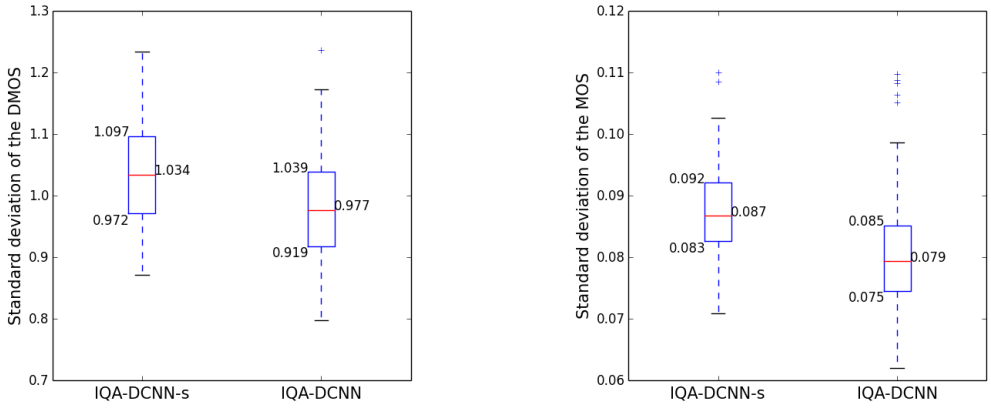
5.3 Robustness in quality assessment across patches

As mentioned in section 4.2.3, during testing we extract 64 patches of size 227 x 227 from every image, whose final quality score is computed equal to the average score on the extracted 64 patches.

To inspect the deviation in quality scores on patches per image, we calculated the 95% confidence interval of image quality predicted by IQA-DCNN and IQA-DCNN-s across 64 patches belonging to same image. Then, we took the average of this value (the 95% interval of predicted quality across patches) across the whole dataset. Figure 5.2a and 5.2b present the box-plots of this average confidence interval across the 100 repetitions on LIVE and TID datasets respectively.

Algorithm	**Evaluation**	**SROCC**	**LCC**
PSNR		0.669	0.652
SSIM [60]		0.878	0.857
FSIM [74]		0.926	0.913
CORNIA [66]	Median	0.813	0.837
IQA-CNN [25]	Average	0.862	0.873
IQA-CNN++ [26]	Average	0.870	0.880
IQA-DCNN-s	Average	0.879	0.887
IQA-DCNN-s	Median	0.900	0.903
IQA-DCNN	Average	0.893	0.901
IQA-DCNN	Median	**0.909**	**0.915**

Table 5.3: Comparison of SROCC and LCC measure on TID dataset. *Italicized* are FR-IQA algorithms for reference.



(a) LIVE (b) TID

Figure 5.2: Box plot of 100 repetitions of evaluation on (a) LIVE and (b) TID dataset: Average 95% confidence interval of image quality prediction on 64 patches per image, using IQA-DCNN-s and IQA-DCNN.

From these plots, It was observed that for both IQA-DCNN and IQA-DCNN-s, the median of standard deviation in quality score with evaluation on LIVE dataset is approximately 1 DMOS (DMOS varies in range [1, 100]), and the median of standard deviation in quality score with evaluation on TID dataset is equivalent to 0.09 MOS (MOS varies in range [1,10]). These reported values are very small which shows that IQA-DCNN and IQA-DCNN-s predict almost similar image quality for all patches of an input image. It was further observed that this confidence interval is marginally narrower for IQA-DCNN than IQA-DCNN-s, which again reflects that IQA-DCNN is little better at predicting the quality of image patches than IQA-DCNN-s.

5.4 Chapter conclusions

In this Chapter, we presented the results of the performance evaluation of IQA-DCNN and IQA-DCNN-s. First we presented the evaluation method that was used for the evaluation of the proposed CNN architectures. As per this method, the IQA-DCNN and IQA-DCNN-s were trained on approximately 60%, validated on 20% and tested on the remaining 20% of the LIVE and TID datasets. Though the similar evaluation method was employed for IQA-CNN and IQA-CNN++, other NR-IQA algorithms (DIIVINE, BLINDS-II, BRISQUE and CORNIA) employed much easier evaluation method by using 80% of the data for training and remaining 20% for testing. Thus in spite of using relatively tough evaluation method, IQA-DCNN and IQA-DCNN-s were able to outperform all existing NR-IQA algorithms. We believe that increasing the training set to 80% would further improve the reported performance of IQA-DCNN and IQA-DCNN-s on LIVE and TID datasets.

Since FR-IQA algorithms use reference images for IQA, they are much easier to achieve than NR-IQA algorithms that do not employ reference images for IQA. From Table 5.2 and 5.3 that report evaluation results on LIVE and TID datasets respectively, it was observed that both IQA-DCNN and IQA-DCNN-s approach the performance of FR-IQA algorithms, and even outperform them in some scenarios (when evaluated on LIVE dataset). Thus these results further indicate the superior performance of IQA-DCNN and IQA-DCNN-s that without using reference images match the performance of algorithms that do use them.

Apart from comparing IQA-DCNN and IQA-DCNN-s with other algorithms, we also compared them among themselves. From box plots shown in Figure 5.1 and the result of individual sample t-test reported in Table 5.1, it was observed that in spite of having 90% more learnable parameters, the performance of IQA-DCNN is just marginally better than IQA-DCNN-s. Thus IQA-DCNN-s could easily substitute IQA-DCNN in numerous on-line applications where low computational complexity is the primary requirement.

We further tested the robustness of IQA-DCNN and IQA-DCNN-s in estimating the perceptual quality score for individual patches of input images. The reported results, through box-plots shown in Figure 5.2, clearly indicate that performance of IQA-DCNN and IQA-DCNN-s is very robust as they could very well estimate the perceptual quality of image patches almost equivalent to the quality of the whole image. The reason behind could be the large size (227 x 227) of employed patches that could be large enough to carry required structural information of an image to estimate its quality through the quality estimation of its single patch. Thus instead of extracting large number of patches from the input images (as we extracted 64 patches per image), much less number of patches could be utilized for estimating the quality of an image through the average quality of extracted patches. This could further reduce the computational demand of IQA-DCNN and IQA-DCNN-s.

Chapter 6

Visualization

This chapter presents the feature visualizations of the IQA-DCNN architecture that was trained separately on LIVE and TID datasets. The goal of this Chapter is to generate informative visualizations of the learned features of IQA-DCNN and use them to understand the characteristics of image quality as identified by the learned features.

We start with section 6.1, in which we present a literature review on various feature visualization methods being used for CNN's. Then in section 6.2 we describe two methods that were utilized for the visualization of IQA-DCNN features. In section 6.3 and 6.4 we present the obtained visualizations using the two selected methods. At last in section 6.5, we discuss the observations related to the generated visualizations.

6.1 Background

Past few years have seen a tremendous increase in the popularity of CNN's and their application in different fields. Though CNN's have been shown to deliver superior performance in tasks like object recognition and image quality assessment, their internal working is typically difficult to understand because of their deep architecture and the large number of utilized learnable parameters. Thus many efforts have been lately made to actually understand their internal working by generating representative visualizations of their learned features.

One of the straightforward methods of feature visualization involves the direct plotting of the CNN weights that are learned during the training process. But this method has only shown to produce good visualizations of the very first CNN layer that directly interacts with the input images [6]. Features of deeper layers are much more abstract to be visualized by this direct weights plotting method. Thus it is more desirable to use other techniques that can also produce informative visualizations of features learned by deeper layers of CNN.

Another set of visualization methods aim to overcome the limitations of weight plotting method by instead plotting the network activations during the forward pass [6]. They could help in assessing the CNN operations by inspecting the changes brought about in an input image as it propagates forward into the network. Another similar method, called deconvolution method[73], also adds a backward pass and highlights

the portions of input images that are responsible for the firing of target neurons in a CNN layer.

These two visualization methods do produce informative visualizations that show the internal working of a CNN, but the generated visualizations are associated with specific network inputs, i.e., different input images result in different visualizations. This makes it tough to generalize the observations to comment on the general functioning of a CNN.

There also exist a third set of visualization methods that aim at generating or extracting image patches that maximally activate the target neuron. These methods help in the visualization of the characteristics of input images that a target neuron is responsible for detecting. Furthermore, these methods are also capable at generating unique and informative visualization of features learned by even deeper layers of a CNN.

6.2 Visualization techniques

Based on the literature review of various feature visualization methods presented in section 6.1, we employed two methods that respectively generate and extract image patches responsible for maximally activating the neurons present in the Conv layers of IQA-DCNN.

1. *Generating synthetic images that maximally activate the target neuron* ***(Synthetic-Max)***: As per this method, a target neuron is selected, and a randomly generated image is fed as an input to the CNN. This is followed by a sequence of fixed number of forward and backward passes in which the input image is repeatedly updated (using gradient descent optimization) to produce higher activations of the target neuron[71] [38].

2. *Retrieving images from dataset that maximally activate the target neuron* ***(Image-Max)***: As per this method, a target neuron is selected, and images from the training dataset are fed through CNN while keeping track of an image patch (or fixed number of patches) that results in maximum activation of the target neuron. Once all images are processed, the recorded patch is retrieved from the dataset.[13]

These methods were selected because they produce unique visualizations that depict the general characteristics of corresponding features, and also because they can be used for the visualization of any Conv layer irrespective of its depth.

6.3 Synthetic-Max Visualizations

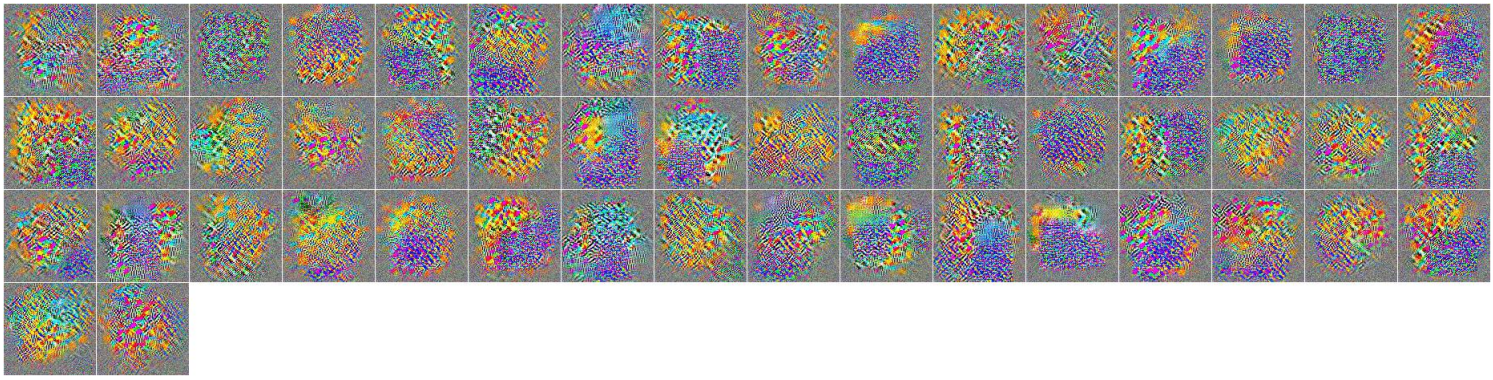
For generating Synthetic-Max visualizations, we initialized the pixel values of a set of images of size 227 x 227 to random values drawn from a Gaussian distribution with the intensity value of RGB channels varying in the range [0,255]. We targeted all neurons of Conv layers of IQA-DCNN whose synthetic images were generated using 100 update iterations of forward and backward passes. To generate synthetic images

for the neurons in the first Conv layer, we also employed the regularization technique of Gaussian blurring the synthetic images (introduced in [71]) after every four iterations. Since the receptive field of Conv1 layer neurons is very small (11 x 11), this regularization technique was observed to improve the generated feature visualizations, without which the output was a very noisy image. No other regularization was used in any other layer of IQA-DCNN.

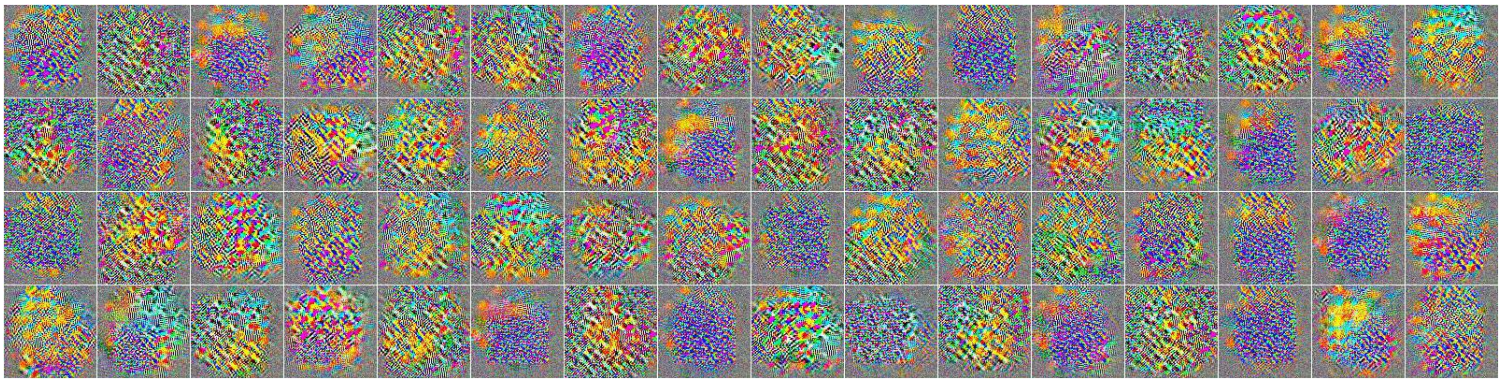
Using Synthetic-Max visualization method, two feature visualizations were then generated:

1. Synthetic-Max visualizations of the neurons in five Conv layers of IQA-DCNN training on LIVE dataset. The generated visualizations are presented in Figure 6.1.

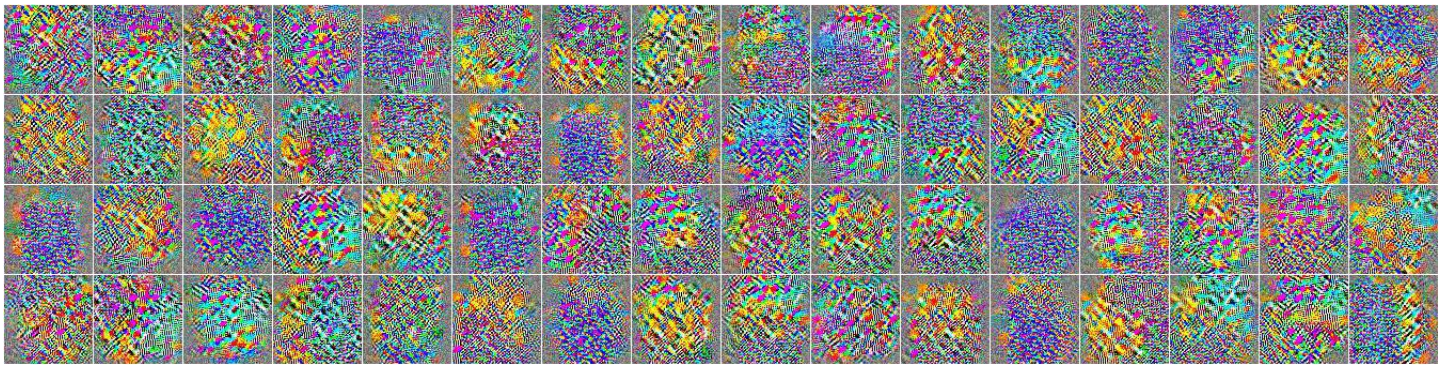
2. Synthetic-Max visualizations of the neurons in five Conv layers of IQA-DCNN training on TID dataset. The generated visualizations are presented in Figure 6.2.



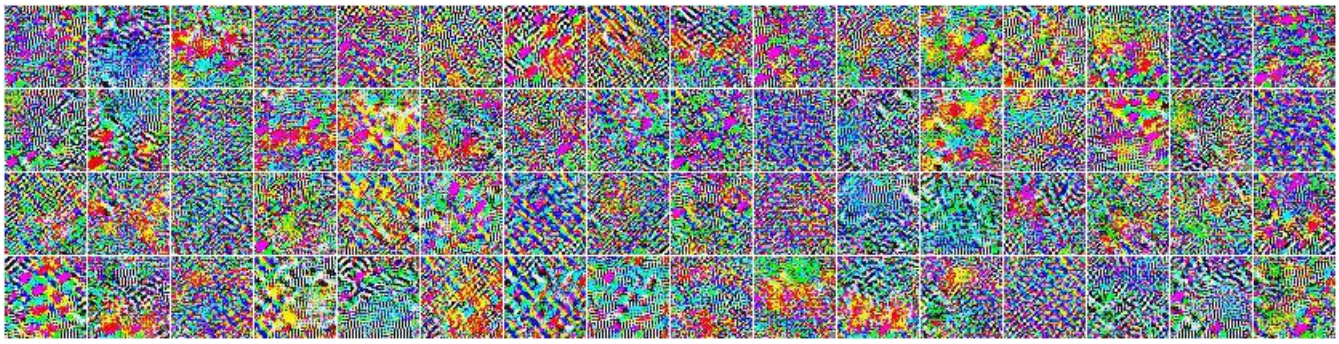
(a) Synthetic images that maximally activates 50 neurons of Conv5 layer trained on LIVE dataset.



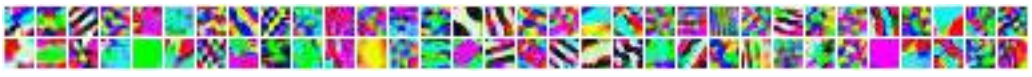
(b) Synthetic images that maximally activates 64 neurons of Conv4 layer trained on LIVE dataset.



(c) Synthetic images that maximally activates 64 neurons of Conv3 layer trained on LIVE dataset.

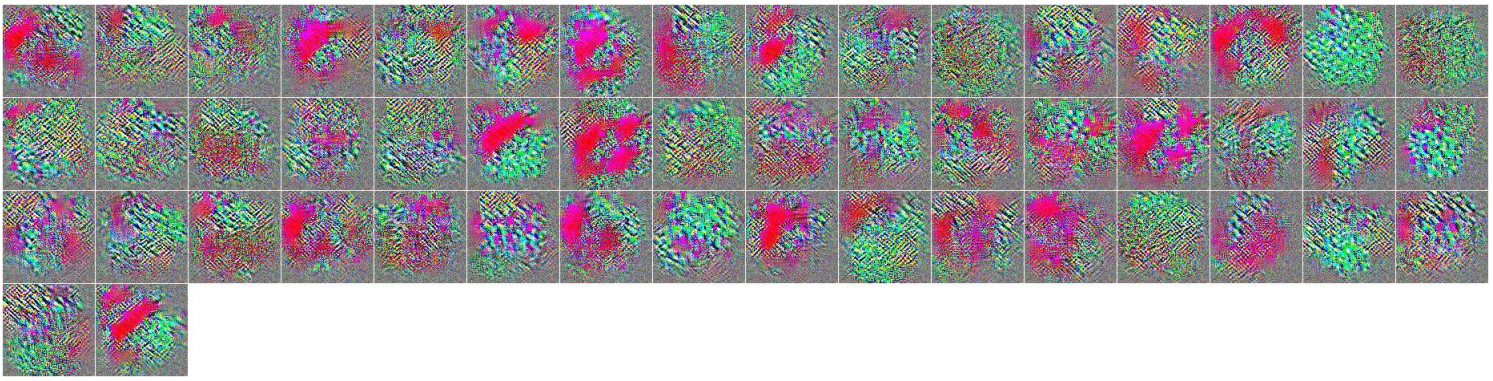


(d) Synthetic images that maximally activates 64 neurons of Conv2 layer trained on LIVE dataset.

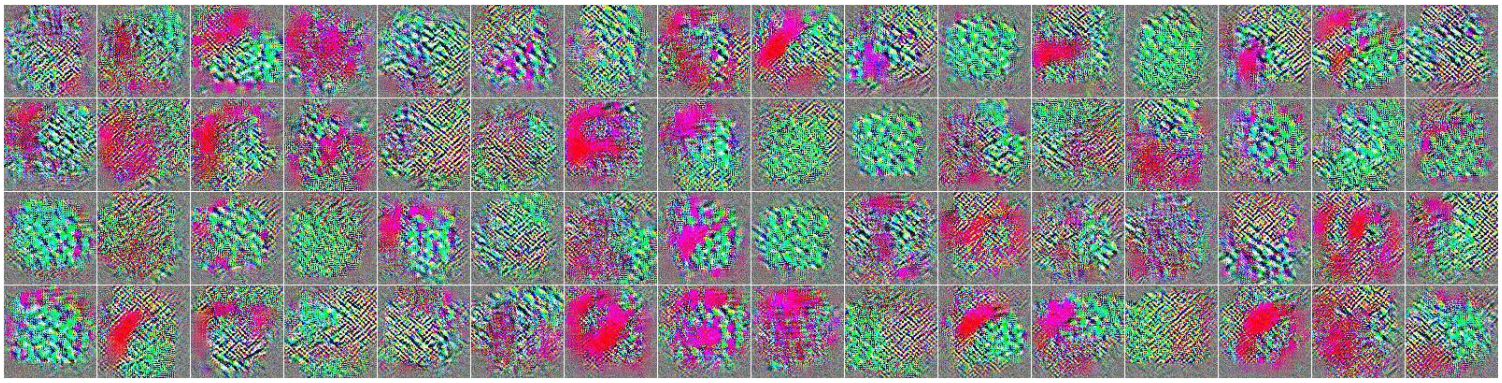


(e) Synthetic images that maximally activates 64 neurons of Conv1 layer trained on LIVE dataset.

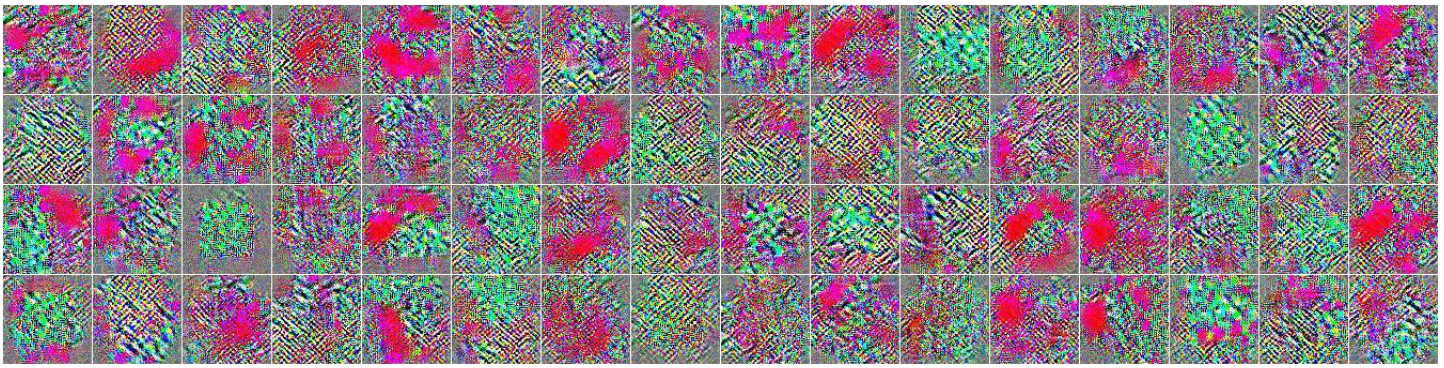
Figure 6.1: Synthetic-Max visualizations of IQA-DCNN trained on LIVE dataset.



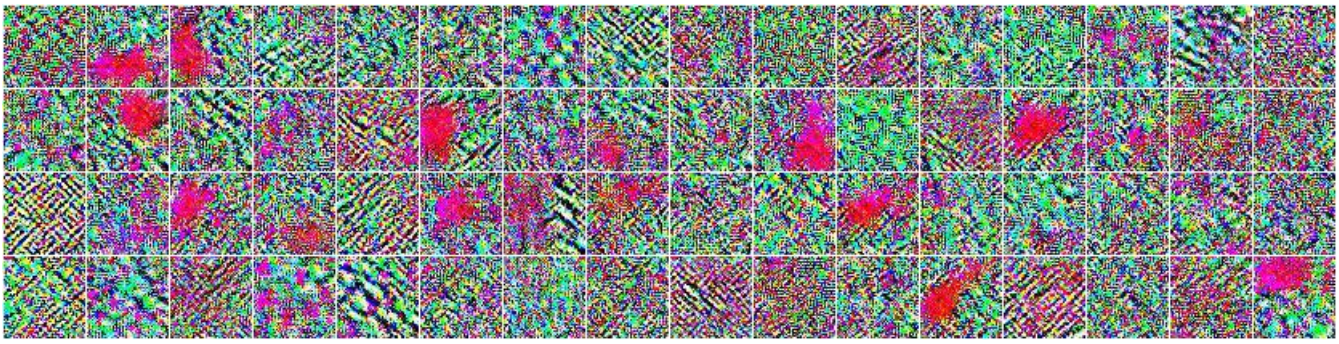
(a) Synthetic images that maximally activates 50 neurons of Conv5 layer trained on TID dataset.



(b) Synthetic images that maximally activates 64 neurons of Conv4 layer trained on TID dataset.



(c) Synthetic images that maximally activates 64 neurons of Conv3 layer trained on TID dataset.



(d) Synthetic images that maximally activates 64 neurons of Conv2 layer trained on TID dataset.

(e) Synthetic images that maximally activates 64 neurons of Conv1 layer trained on TID dataset.

Figure 6.2: Synthetic-Max visualizations of IQA-DCNN trained on TID dataset

6.4 Image-Max Visualizations

For generating Synthetic-Max visualizations, we directly employed the images of LIVE and TID dataset and fed them to two different IQA-DCNN architectures that were respectively trained LIVE and TID datasets. All images were then fed to the network, the index of the top nine maximally activating image patches was recorded for every neuron of each Conv layer. The recorded image patches were then retrieved from their respective positions at the end of the processing phase.

Using Image-Max visualization method, two feature visualizations were then generated:

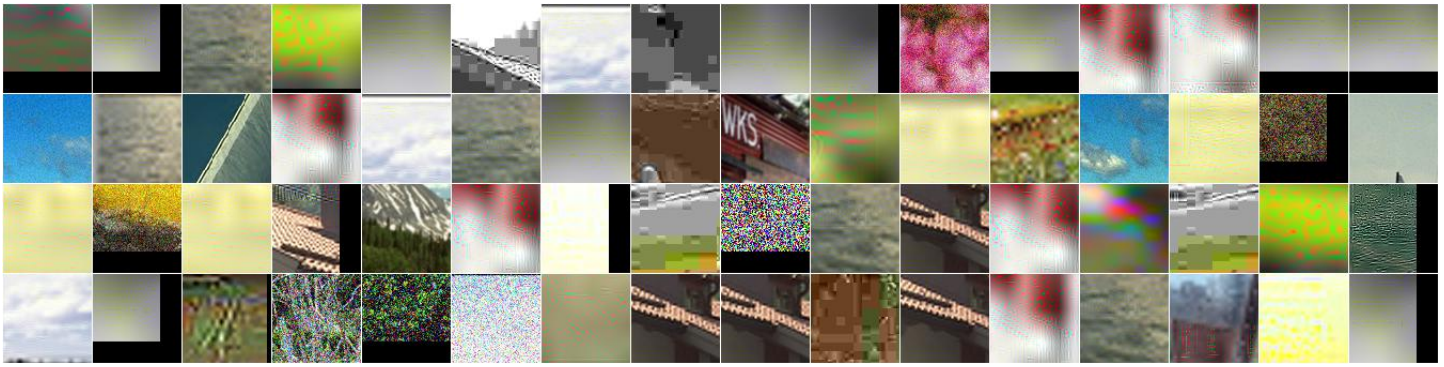
1. The Image-Max visualizations of the image patches of LIVE that maximally activate the neurons in the five Conv layers of the IQA-DCNN architecture trained on LIVE (Figure 6.3).

2. The Image-Max visualizations of the image patches of TID that maximally activate the neurons in the five Conv layers of the IQA-DCNN architecture trained on TID (Figure 6.4). We excluded the non- natural scene image for this visualization.

We also computed the distortion types of top nine image patches against the neurons of five convolutional layers that were maximally activated by them. We then computed the statistical mode of those distortion types per neuron to get an approximate idea as of whether neurons would specialize in processing a particular type of distortion. The graph containing these mode distortion types for IQA-DCNN trained on the LIVE dataset is presented in Figure 6.5, and another graph for IQA-DCNN trained on TID dataset is presented in Figure 6.6.

(a) Image patches that maximally activates 50 neurons of Conv5 layer trained on LIVE dataset.

(b) Image patches that maximally activates 64 neurons of Conv4 layer trained on LIVE dataset.

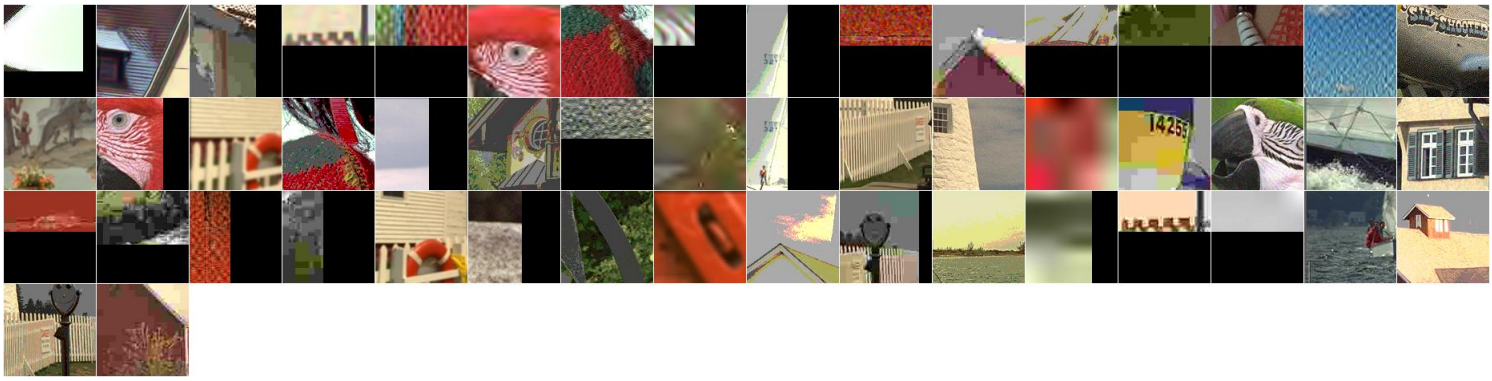


(c) Image patches that maximally activates 64 neurons of Conv3 layer trained on LIVE dataset.

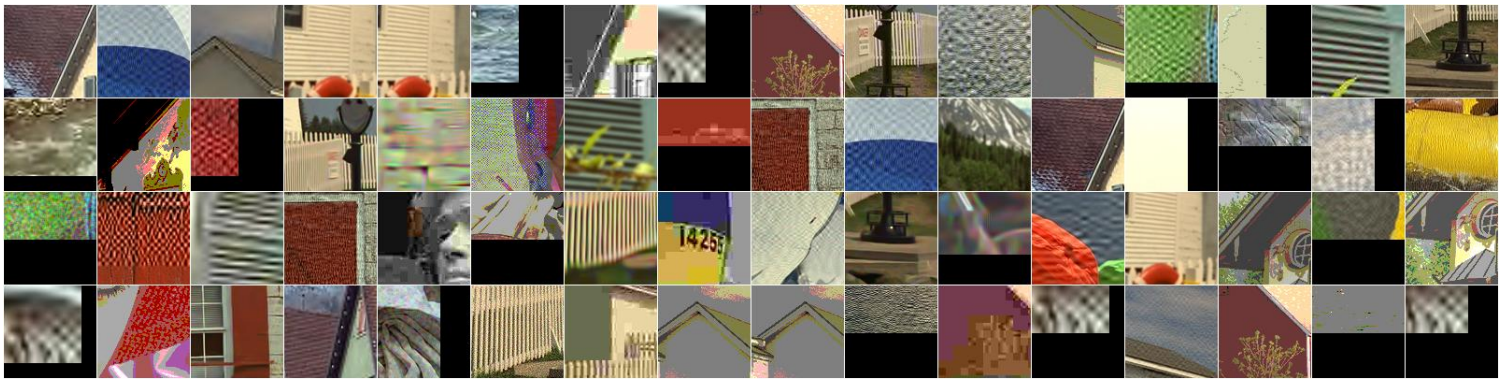
(d) Image patches that maximally activates 64 neurons of Conv2 layer trained on LIVE dataset.

(e) Image patches that maximally activates 64 neurons of Conv1 layer trained on LIVE dataset.

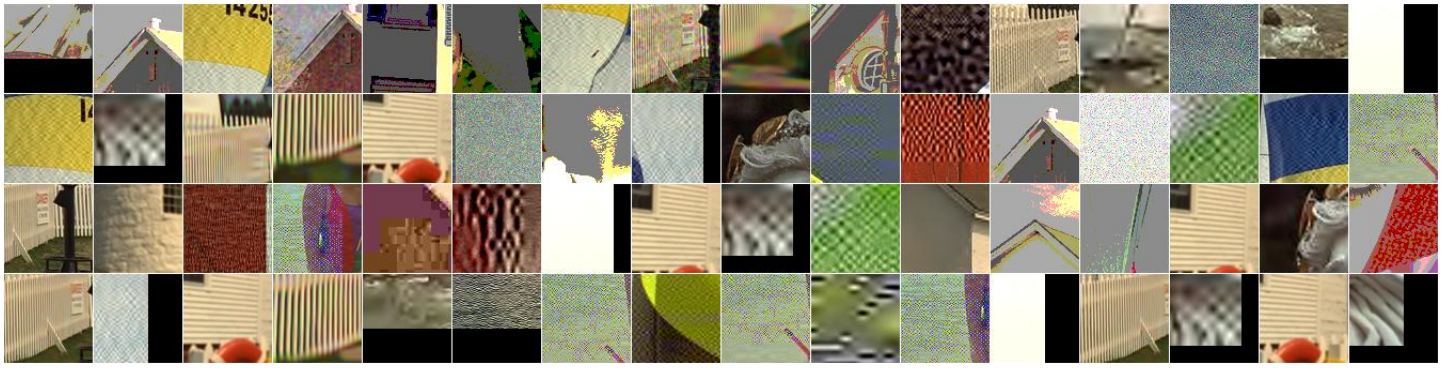
Figure 6.3: Image-Max visualizations of IQA-DCNN trained on LIVE dataset, using images from LIVE dataset.



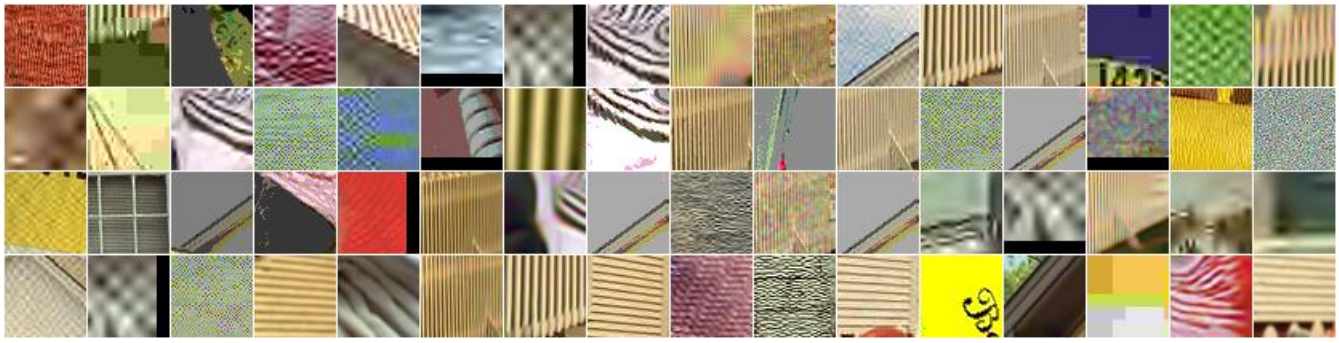
(a) Image patches that maximally activates 50 neurons of Conv5 layer trained on TID dataset.



(b) Image patches that maximally activates 64 neurons of Conv4 layer trained on TID dataset.



(c) Image patches that maximally activates 64 neurons of Conv3 layer trained on TID dataset.



(d) Image patches that maximally activates 64 neurons of Conv2 layer trained on TID dataset.

(e) Image patches that maximally activates 64 neurons of Conv1 layer trained on TID dataset.

Figure 6.4: Image-Max visualizations of IQA-DCNN trained on TID dataset, using images from TID dataset

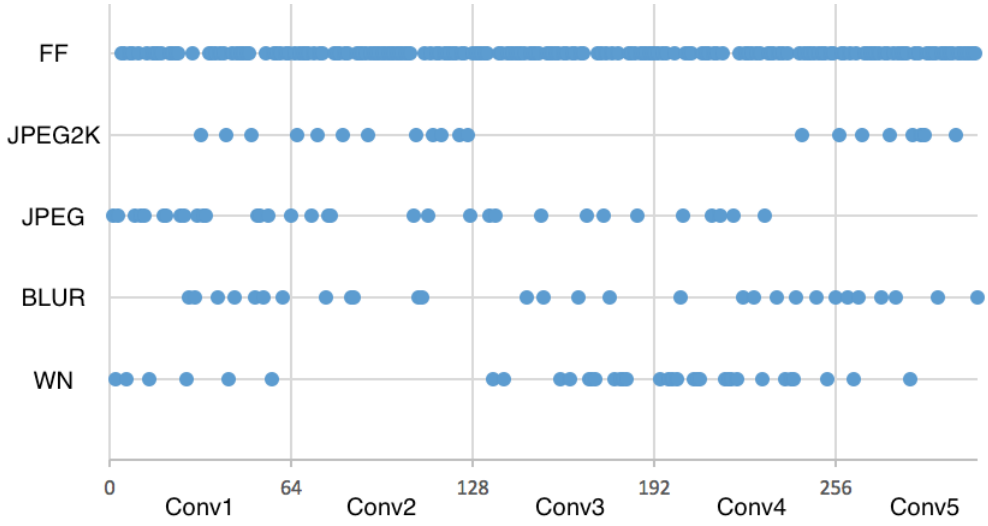


Figure 6.5: Plot of the statistical mode distortion types among distortion types of top 9 images with respect to the neuron they maximally activated in IQA-DCNN trained on LIVE dataset.

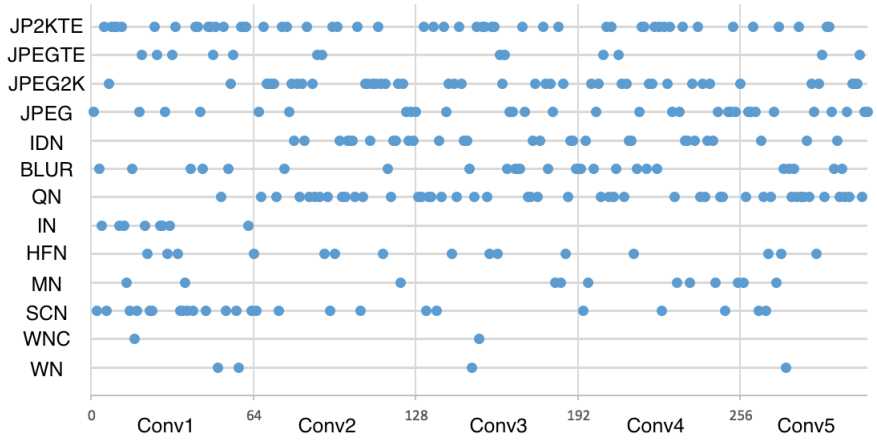


Figure 6.6: Plot of the statistical mode distortion types among distortion types of top 9 images with respect to the neuron they maximally activated in IQA-DCNN trained on TID dataset.

6.5 Discussion

In this section, we present some of the interesting observations that were made from the generated visualizations of the features of IQA-DCNN.

- The Synthetic-Max and Image-Max visualizations of the Conv1 layer were observed to contain images of simple patterns and colors, which behaves like edge detectors and color filters on the input images. It is a very similar behavior to the first layer of CNN's trained for object recognition tasks [71].

- The Synthetic-Max visualizations of the Conv2 layer were observed to contain different combinations of many simple patterns that were detected in the visualization of the Conv1 layer. From this observation, we believe that in Conv2 layer, IQA-DCNN looks for textured areas. This argument is also based on the observations from Image-Max visualization of the Conv2 layer that shows many patches with different textures. The textured areas are rich in high frequencies and therefore very sensitive for some types of distortions such as blur and jpeg.

  We also point out that this behavior of Conv2 layer is very different from the second layer of CNN's trained for the application of object recognition: that looks for complex structures (like eyes, wheels, etc.) in an input image. Thus it may suggest that CNN's trained for IQA application start to develop different types of features from the very second layer of the CNN architecture. It may suggest that we can only adopt the pre-trained filters of the first layer from the CNN trained for the application of object recognition. This observation also explains our findings presented in section 4.1.1 in which we were not able to successfully fine-tune a CNN pre-trained on object recognition dataset.

- Similar behavior to Conv2 layer was also observed in Conv3, Conv4, and Conv5 layers as they also seem to look for textured areas. But from the Synthetic-Max visualizations, it was observed that though the patterns found in Conv1 layer visualizations do combine together in Conv2, Conv3, Conv4 and Conv5 layers, the complexity of the combinations of these patterns also increases in the same sequence. Conv2 layer visualizations contain simpler combinations of few patterns found in Conv1 layer visualizations, whereas Conv5 layer visualizations contain much more of these patterns.

- The IQA-DCNN employs zero padding in Conv3, Conv4, and Conv5 layers. From the Image-Max visualization of these layers, it was observed that some of the neurons get highly activated at the edge of an image that falls into their receptive field because of the employed zero padding. From this observation, we believe that IQA-DCNN also looks for fast color transitions in input images that can also point to a distortion type. The number of such observed neurons also increases from Conv3 layer to Conv5 layer, which may suggest that these features become more important as we go deeper into the CNN architecture.

  From these observations, we recommend to not use zero padding in CNN's trained for the application of image quality assessment because the zero padding could also be considered as a distortion in an image.

- From the inspection of graphs shown in Figure 6.5 and 6.6, some distortion types, more than others, were observed to be among the majority of max-activating images of the larger number of neurons of IQA-DCNN. This could further help in understanding better the role of different layers with respect to the distortion type of input images. For example, It is interesting to note that the Conv2 layer, which looks for simple textures, does not activate with noise. Though both textures and noise have strong high frequency components, the network seems to be able to distinguish high frequency due to structure, from high frequency due to noise.

Chapter 7

Conclusions and Future Work

In this Chapter we present the main contributions of this thesis project, point out some of its limitations, and provide suggestions for the future work.

7.1 Contributions

In this thesis project, our two main goals were: to improve upon the limitations of existing NR-IQA metrics by designing an efficient and suitable NR-IQA based CNN architecture, and to understand its internal working by generating informative visualizations of its learned features. Thus this work provides two main contributions.

Convolutional neural network design: After performing an intensive design space exploration, we proposed two CNN architectures – IQA-DCNN (shown in Figure 4.10) and IQA-DCNN-s (shown in Figure 4.13) for NR-IQA, the former privileging prediction accuracy, the latter minimizing computational complexity – that were found suitable for the application of NR-IQA. Both of the proposed architectures contain five Conv layers and operate an input patch of size 227 x 227. To the best of our knowledge, the proposed CNN architectures are the deepest and perceptually most accurate (as they employ bigger input patches that bring patch quality closer to the overall image quality) among all NR-IQA based CNN architectures. They were also reported as best-performing architectures among existing state of the art NR-IQA metrics.

The IQA-DCNN-s contains around 20,000 learnable parameters, which also makes it the most compact CNN architecture for NR-IQA task. It contains around 75% less learnable parameters than IQA-CNN++[26], which used to be the most compact architecture. Furthermore, the bigger size of input patches allows both IQA-DCNN and IQA-DCNN-s to scan an image much faster than other architectures that operate on the smaller size of input patches. For instance, since IQA-CNN[25] operates on an input patch of size 32 x 32 (biggest patch size among existing NR-IQA based CNN architectures), it has to operate $(227/32) * (227/32) \approx 50$ times to scan an area of 227 x 227 that IQA-DCNN and IQA-DCNN-s could cover in a single operation.

In this thesis project, we have introduced an efficient data augmentation technique (see section 4.2.3), that was inspired from the CNN's employed in the object recognition tasks. This technique allowed us to train CNN architectures with a large amount of learnable parameters on small IQA datasets. Using this technique, we were able to

successfully train a CNN architecture with as many as 6 million learnable parameters in our trials. Further, by successfully training an architecture similar to AlexNet with 227 x 227 input patches, we have shown that it is possible to adopt CNN architectures (their design and not their learned parameters) for NR-IQA that have been developed for the application of object recognition. Thus the proposed data augmentation technique could be used in future to adopt deeper and efficient CNN architectures (for example, GoogleNet[57] architecture with 22 layers and 4 million learnable parameters) to provide performance improvement in NR-IQA applications.

Feature visualization: We generated Synthetic-Max and Image-Max visualizations of the IQA-DCNN features, which respectively helped us in visualizing synthetic images and dataset image patches that maximally activate IQA-DCNN neurons. Both of these visualizations helped us in understanding the image characteristics that neurons look for. We observed that neurons of IQA-DCNN looked for textured areas and fast color transitions. These features are very different from the higher layers of CNN's in object recognition task that look for complex structures like wheels, eyes, faces, etc. [71]. This observation suggests that it is not advantageous to adopt the parameters of CNN's that are trained on object recognition datasets (which are very big in size), to be later fine-tuned using IQA datasets, which is a recommended technique for training CNN's with less training data [70].

We further found that IQA-DCNN considers zero padding as distortion, and thus to prevent any potential harm to the overall performance, zero-padding should not be employed in CNN's that are trained for the application of NR-IQA.

7.2 Limitations

Though this thesis project was successful in improving upon the existing NR-IQA metrics, there are still many limitations to this work. There are dimensions of the design space which still have to be explored. For example, we constrained the number of channels to be the same in all convolutional layers, but this may be suboptimal. Similarly, we may have developed an entirely new architecture for the global feed, for which instead we used the architecture developed for the local one. These assumptions were made to limit the design space to be explored in given time, but they might have also prevented us from converging to a more optimal CNN design, that could have contained even less learnable parameters and supposedly have given better performance.

We based our CNN design on an outdated AlexNet[29] architecture, whereas many new and efficient CNN architectures (like VGG[54], GoogleNet[57], ResNet[18], etc.) have been proposed since AlexNet and using them could have resulted in further improvements in our reported performance.

There is yet another limitation in our employed methodology as per which the design phase was followed by evaluation phase, which was then followed by inspection phase. During inspection phase, we realized that zero padding was a bad design choice for IQA-DCNN as it potentially harms its performance by acting as distortion. Thus a better choice of methodology could have been the design phase, followed by inspection phase, then followed by a second design phase for fine-tuning, which could then have

been followed by the evaluation phase. As per this new methodology, we could have dropped zero padding from our design and this could have supposedly resulted in improved performance during the evaluation phase.

7.3 Future work

Based on findings of this thesis, we provide suggestions for the future work on further improvements of CNN design.

We suggest improving upon our CNN architecture by further exploring different combinations of the number of Conv channels in Conv layers. It could be possible to drop some of the Conv channels from the Conv layers without affecting the performance of the proposed architectures. This could help in decreasing the overall number of learnable parameters and thus could make the resulting architectures even more compact.

We also recommend to further investigate the addition of global feed to current CNN architectures. The global feed could provide a network with overall structural information of images, and since image quality is also assumed to be dependent upon the structural information present in an image [60], this could further result in overall improvement in performance. To start with, a smaller global feed (in terms of learnable parameters) with better regularization could be adopted to prevent the issue of overfitting. Since global feed is expected to learn structural information of an image, it could also be pre-trained with object recognition datasets and later could be fine-tuned along with local feed on IQA datasets.

We recommend on exploring other successful CNN architectures as well, which could help in further improving the performance of CNN's in NR-IQA tasks. For example, GoogleNet[57] is a much deeper architecture with total 22 Conv layers and was able to provide around 10% reduction in top 5 error of ILSVRC dataset, over AlexNet.

We also recommend exploring other feature visualization techniques that we did not employ in this project, for example, visualization of layer activations or deconvolution based visualization techniques [71]. Though visualizations produced by these techniques are specific to input images, they could help in understanding the internal working of CNN's with respect to images of different distortion types. This understanding could further help in designing CNN's that could perform better on a given set of distortion types.

Bibliography

[1] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[2] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[3] Jason Brownlee. An Introduction to Feature Selection. http://machinelearningmastery.com/an-introduction-to-feature-selection/, 2014.

[4] Damon M Chandler. Seven challenges in image quality assessment: past, present, and future research. *ISRN Signal Processing*, 2013, 2013.

[5] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[6] Stanford CS231n class notes. CS231n: Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io, 2016.

[7] Anderson de Andrade. Best practices for convolutional neural networks applied to object recognition in images.

[8] Anderson de Andrade. Best practices for convolutional neural networks applied to object recognition in images.

[9] Sergio Decherchi, Paolo Gastaldo, Rodolfo Zunino, Erik Cambria, and Judith Redi. Circular-elm for the reduced-reference assessment of perceived image quality. *Neurocomputing*, 102:78–89, 2013.

[10] Jia Deng, Alex Berg, Sanjeev Satheesh, H Su, Aditya Khosla, and L Fei-Fei. Imagenet large scale visual recognition competition 2012 (ilsvrc2012), 2012.

[11] Li Deng and Dong Yu. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

[12] Weiguang Ding, Ruoyan Wang, Fei Mao, and Graham Taylor. Theano-based large-scale visual recognition with multiple gpus. *arXiv preprint arXiv:1412.2302*, 2014.

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[14] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[15] Ian J. Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013.

[16] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[17] DANIEL Graupe, Ruey Wen Liu, and GEORGE S Moschytz. Applications of neural networks to medical signal processing. In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 343–347. IEEE, 1988.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[20] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[22] Recommandation ITU-T. P910. *Subjective video quality assessment methods for multimedia applications*, 2008.

[23] Recommandation ITU-T. P913. *Methods for the subjective assessment of video quality, audio quality and audiovisual quality of Internet video and distribution quality television in any environment*, 2014.

[24] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[25] Le Kang, Peng Ye, Yi Li, and David Doermann. Convolutional neural networks for no-reference image quality assessment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[26] Le Kang, Peng Ye, Yi Li, and David Doermann. Simultaneous estimation of image quality and distortion via multi-task convolutional neural networks. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2791–2795. IEEE, 2015.

[27] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[30] LISA Lab. Convolutional neural networks (LeNet) - DeepLearning 0.1 documentation. http://deeplearning.net/tutorial/lenet.html, 2015.

[31] Qiang Li and Zhou Wang. Reduced-reference image quality assessment using divisive normalization-based image representation. *IEEE Journal of Selected Topics in Signal Processing*, 3(2):202–211, 2009.

[32] log0. Differences between L1 and L2 as Loss Function and Regularization. http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/, 2013.

[33] Xin Lu, Zhe Lin, Hailin Jin, Jianchao Yang, and James Z Wang. Rapid: Rating pictorial aesthetics using deep learning. In *Proceedings of the ACM International Conference on Multimedia*, pages 457–466. ACM, 2014.

[34] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[35] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *Image Processing, IEEE Transactions on*, 21(12):4695–4708, 2012.

[36] Pedram Mohammadi, Abbas Ebrahimi-Moghadam, and Shahram Shirani. Subjective and objective quality assessment of image: A survey. *arXiv preprint arXiv:1406.7799*, 2014.

[37] Anush Krishna Moorthy and Alan Conrad Bovik. Blind image quality assessment: From natural scene statistics to perceptual quality. *Image Processing, IEEE Transactions on*, 20(12):3350–3364, 2011.

[38] Alexander Mordvintsev. Inceptionism: Going Deeper into Neural Networks. https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html, 2016.

[39] Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2408–2415. IEEE, 2012.

[40] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Doklady an SSSR*, volume 269, pages 543–547, 1983.

[41] Nikolay Ponomarenko, Vladimir Lukin, Alexander Zelensky, Karen Egiazarian, M Carli, and F Battisti. Tid2008-a database for evaluation of full-reference visual quality assessment metrics. *Advances of Modern Radioelectronics*, 10(4):30–45, 2009.

[42] ITURBT Recommendation. 500-11,âmethodology for the subjective assessment of the quality of television pictures,â recommendation itu-r bt. 500-11. *ITU Telecom. Standardization Sector of ITU*, 2002.

[43] Judith A Redi, Paolo Gastaldo, Ingrid Heynderickx, and Rodolfo Zunino. Color distribution information for the reduced-reference assessment of perceived image quality. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(12):1757–1769, 2010.

[44] Judith A Redi, Yi Zhu, Huib de Ridder, and Ingrid Heynderickx. How passive image viewers became active multimedia users. In *Visual Signal Quality Assessment*, pages 31–72. Springer, 2015.

[45] Abdul Rehman and Zhou Wang. Reduced-reference image quality assessment by structural similarity estimation. *IEEE Transactions on Image Processing*, 21(8):3378–3389, 2012.

[46] Ann M Rohaly, Philip J Corriveau, John M Libert, Arthur A Webster, Vittorio Baroncini, John Beerends, Jean-Louis Blin, Laura Contin, Takahiro Hamada, David Harrison, et al. Video quality experts group: current results and future directions. In *Visual Communications and Image Processing 2000*, pages 742–753. International Society for Optics and Photonics, 2000.

[47] Sebastian Ruder. Blog: An overview of gradient descent optimization algorithms. http://sebastianruder.com/optimizing-gradient-descent/index.htmlminibatchgradientdescent, 2016.

[48] Daniel L Ruderman. The statistics of natural images. *Network: computation in neural systems*, 5(4):517–548, 1994.

[49] Michele A Saad, Alan C Bovik, and Christophe Charrier. Blind image quality assessment: A natural scene statistics approach in the dct domain. *Image Processing, IEEE Transactions on*, 21(8):3339–3352, 2012.

[50] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv preprint arXiv:1602.07868*, 2016.

[51] Hamid R Sheikh, Zhou Wang, Lawrence Cormack, and Alan C Bovik. Live image quality assessment database release 2, 2005.

[52] D Amnon Silverstein and Joyce E Farrell. The relationship between image fidelity and image quality. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 881–884. IEEE, 1996.

[53] Phil Simon. *Too Big to Ignore: The Business Case for Big Data*, volume 72. John Wiley & Sons, 2013.

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[55] Hyun Ah Song and Soo-Young Lee. Hierarchical representation using nmf. In *Neural Information Processing*, pages 466–473. Springer, 2013.

[56] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[57] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[58] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[59] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[60] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[61] Zhou Wang and Xinli Shang. Spatial pooling strategies for perceptual image quality assessment. In *2006 International Conference on Image Processing*, pages 2945–2948. IEEE, 2006.

[62] Wikipedia. Artificial neural networks - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Artificial_neural_network, 2016.

[63] Wikipedia. Deep learning - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Deep_learning, 2016.

[64] Wikipedia. Image quality - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Image_quality, 2016.

[65] Wikipedia. Overfitting - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Overfitting, 2016.

[66] Peng Ye, Jayant Kumar, Le Kang, and David Doermann. Unsupervised feature learning framework for no-reference image quality assessment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1098–1105. IEEE, 2012.

[67] Peng Ye, Jayant Kumar, Le Kang, and David Doermann. Real-time no-reference image quality assessment based on filter learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 987–994, 2013.

[68] Peng Ye, Jayant Kumar, Le Kang, and David Doermann. Real-time no-reference image quality assessment based on filter learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 987–994, 2013.

[69] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.

[70] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[71] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[72] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

[73] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[74] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. Fsim: a feature similarity index for image quality assessment. *Image Processing, IEEE Transactions on*, 20(8):2378–2386, 2011.

Appendix A

Glossary

In this appendix, we give an overview of the frequently used terms.

CNN: Convolutional Neural Network

IQA: Image Quality Assessment

FR-IQA: Full-Reference Image Quality Assessment

RR-IQA: Reduced-Reference Image Quality Assessment

NR-IQA: No-Reference Image Quality Assessment

Conv layer: Convolutional layer

FC layer: Fully Connected layer

Pool unit: Pooling unit

GCN unit: Global Contrast Normalization unit

LCN unit: Local Contrast Normalization unit

LRN unit: Local Response Normalization unit

Training set: Set of examples used for CNN training.

Validation set: Set of examples used for CNN performance evaluation during the training process.

Test set: Set of examples used for CNN performance evaluation at the end of training process.

1 Epoch: Training over 1 training set

1 Iteration: Training over 1 training example

Appendix B

Hardware Specifications

In this appendix, we give an overview of the hardware specifications utilized in this thesis project.

All experiments for the training of CNN's were executed on NVIDIA GPU's that helped us in achieving over 15 times faster executions than CPU's. To provide even further speed up, CPU's were utilized for preparing new minibatches in parallel to the GPU operations on previous minibatches. Two clusters were used for the execution of the experiments conducted in this thesis project.

B.1 INSY cluster

INSY is a cluster from Intelligent Systems Department at TU Delft. It contains total 304 CPU cores, 3 NVIDIA Quadro K2200 GPU's, and 1.705 TB memory. Each of the offered GPU's contains local 4GB GPU memory. It is a cluster of all Linux servers and uses the SLURM scheduler[69] for efficient workload management. This cluster was utilized for conducting the design space exploration of CNN as presented in Chapter 4.

B.2 Cartesius cluster

Cartesius is a Dutch national supercomputer that contains total 40,960 cores, 132 NVIDIA Tesla K40 GPU's, and 117 TB memory. Each of the offered GPU's contains 12GB GPU memory. It is a cluster of all Linux servers and uses the SLURM scheduler[69] for efficient workload management. This cluster was utilized for conducting the performance evaluation of IQA-DCNN and IQA-DCNN-s as presented in Chapter 5.

Appendix C

Software Specifications

In this appendix, we give an overview of the specifications of software that were utilized in this thesis project.

We used two different set of software tools, one for the design space exploration and performance evaluation of CNN, and another for generating the feature visualization of IQA-DCNN. The specification of these two software tools is described in the following.

C.1 CNN design and performance evaluation

For the training of CNN, we modified the code from [12], that was originally written for the implementation of AlexNet[29] with the ILSVRC-2012 dataset. The code was written in Python and besides standard python libraries, below mentioned libraries that are specific to deep neural networks were used:

1. **Theano**[58]: It is a Python library that facilitates an efficient way of evaluating, defining, and optimizing mathematical expressions involving multi-dimensional arrays. It provides good support for coding deep learning systems.

2. **cuDNN**[5]: cuDNN is the NVIDIA library that is used in various deep learning applications. It provides an optimized version of some the operations like convolution, which is one of the main parts of convolutional neural networks.

3. **Pylearn2**[15]: Pylearn2 is a machine learning library. Most of its functionality is built on top of Theano. It is a wrap around of GPU code from Alex Krizhevsky cuda-convnet project. It provides optimized convolution operation and modules to do overlapped pooling and response normalization.

Both Pylear2 and cuDNN libraries provide the functionality for performing optimized convolution operation on GPU, but it was observed that cuDNN convolution is faster than Pylearn2 convolution, thus former was used utilized in this project.

C.2 CNN feature visualization

For generating the visualizations of learned features, we made use of Caffe framework[24], which is deep learning framework especially used for the applications of CNN's.

As mentioned in Chapter 6, we generated two feature visualization of IQA-DCNN. For these two visualization techniques, we utilized two different sources over which our code was created.

1. **Synthetic-Max**: The code for generating Synthetic-Max visualizations was modified from [38].
2. **Image-Max**: The code for generating Image-Max visualizations was modified from [71].