Multipurpose Image Quality Assessment for Both Human and Computer Vision Systems via Convolutional Neural Network

by

Han Yin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2017

Abstract

Computer vision algorithms have been widely used for many applications, including traffic monitoring, autonomous driving, robot path planning and navigation, object detection and medical image analysis, etc. Images and videos are typical input to computer vision algorithms and the performance of computer vision algorithms are highly correlated with the quality of input signal. The quality of videos and images are impacted by vision sensors; environmental conditions, such as lighting, rain, fog and wind. Therefore, it is a very active research issue to determine the failure mode of computer vision by automatically measuring the quality of images and videos.

In the literature, many algorithms have been proposed to measure image and video qualities using reference images. However, measuring the quality of image and video without using a reference image, known as no-reference image quality assessment, is a very challenging problem. Most existing methods use a manual feature extraction and a classification technique to model image and video quality. Internal image statics are considered as feature vectors and classical machine learning techniques such as support vector machine and naive Bayes as the classifier. Using convolutional neural network (CNN) to learn the internal statistic of distorted images is a newly developed but efficient way to solve the problem. However, there are also new challenges in image quality assessment field. One of them is the wide spread of computer vision systems. Those systems, like human viewers, also demand a certain method to measure the quality of input images, but with their own standards.

Inspired by the challenge, in this thesis, we propose to build an image quality assessment system based on convolutional neural network that can work for both human and computer vision system. In specific, we build 2 models: DAQ1 and DAQ2 with different design concept and evaluate their performance. Both models can work well with human visual system and outperform most former state-of-art Image Quality Assessment (IQA) methods. On computer vision system side, the models also show certain level of prediction power and reveal the potential of CNNs in facing this challenge. The performance in estimating image quality is first evaluated using 2 standard data-sets and against three state-of-the art image quality methods. Further, the performance in automatically detecting the failure mode computer vision algorithm is evaluated using Miovision's computer vision algorithm and datasets.

Acknowledgements

I would like to thank all the people who made this thesis possible.They are my supervisors and my colleagues in the Miovision Technology CV Research group.

First of all, I gratefully acknowledge my supervisors, Prof. Naik and Dr. Mishra. Prof. Naik provides me the chance of being a research student and guide me to the principle of doing research. Dr. Mishra acts as both my co-supervisor and my colleague. He offers me a lot of help in my work as well as in my study.

I would like to give special thanks to my colleagues in Miovision CV team: Andrew, Mohan, Justin, Nick, Joe and Dave. They are always ready to help me when I have problem in building models or coding.

I would also like to thank Prof. Shen for guiding me through thesis writing and encouraging me all the time.

Table of Contents

List of Tables

List of Figures

List of abbreviations

ANN Artificial Neural Network 2, 3, 5, 12

BP Back Propagation 13, 14

CNN Convolutional Neural Network viii, 2–5, 15, 17, 19–21, 24, 30, 31, 36, 49, 50, 53

DAC Decoupled Active Contour, the name of a computer vision system which detects vehicles, made by Miovision Technologies Inc. ix, x, 4, 24–28, 31, 36, 37, 45, 49–52, 54

DAQ Deep Algorithm Quality, the name of our model vi, vii, ix, x, 3, 24, 25, 30–33, 36–38, 40–45, 47–51, 53, 54, 59, 62

DMOS Differential Mean Opinion Score 2, 25, 27, 36

IQA Image Quality Assessment viii, ix, 1–5, 7–10, 12, 20, 24, 25, 30, 31, 34, 37, 42, 44, 49, 53, 54

LCC Linear Correlation Coefficient vii, ix, x, 42–45, 47–49, 53, 54

MAE Mean Absolute Error 2–4, 20, 36

MCC Matthews Correlation Coefficient ix, 3, 4, 28, 29, 36

MOS Mean Opinion Score 2, 7–9, 25, 27, 36

MSE Mean Square Error viii, 9–11

ReLU Rectified Linear Unit, a non-linear layer in CNN viii, ix, 18, 19, 32, 35–37

SROCC Spearman's Rank Correlation Coefficient vii, ix, x, 42–45, 47, 48, 53, 54

Chapter 1

Introduction

1.1 Image Quality Assessment and Neural Networks

Image quality assessment (IQA) is a challenging task in signal processing and computer vision field. Digital images are widely used for a variety of purposes nowadays. However, these images can never be perfect. Distortions might be brought in during collecting, compressing, transmitting, storing and all other processing done on the image. To ensure the end user getting an image with sufficient quality, an assessment of image quality is needed. With the growing number of images, it is necessary to develop algorithms that automatically evaluate the quality of images rather than hire more human annotators. Since then IQA has become an important but challenging field in digital image processing system. Depending on the accessibility of reference image, IQA is divided into three categories:

- Full-Reference IQA (FR-IQA) [7, 8]:

  The algorithm gets access to both the distorted image and the reference image which is considered as perfect quality.

- Reduced-Reference IQA (RR-IQA) [9]:

  The algorithm gets access to the distorted image and some abstract features obtained from the reference image.

- No-Reference IQA (NR-IQA) [10]:

  The algorithm only gets access to the distorted image.

Among the FR-IQA techniques, Visual Information Fidelity (VIF) [11] and Feature Similarity Index Metric (FSIM) [12] provide a high degree of correlation with human visual perception of image quality. Although remarkable advancements have been made in the area of FR-IQA and RR-IQA methods, these methods are not well implemented in industry since the ground truth is not accessible in most production environments. Therefore, NR-IQA techniques are more practical, which is the focus of our work.

On the other hand, the definition of "quality" is an abstract concept that depends on the usage of image. Many different measurements are developed to quantize the quality of image. Common measurements can be divided into subjective scores and objective scores [13]. Subjective scores like MOS (Mean Opinion Score) and DMOS (Differential Mean Opinion Score) are based on the feeling of human viewers. Objective scores, from simplest methods like SNR, PSNR, MAE (Mean Absolute Error) to sophisticated methods like FSIM (Feature Similarity) and SSIM (Structural Similarity), are based on their own algorithms that simulate human behavior.

When computer vision systems are considered, more challenges and opportunities are brought to IQA. Measuring the quality of image for computer vision systems could be as useful as measuring image quality for human beings. By knowing the quality of input images, user can get aware of bad inputs and take action to remove or correct them, and thus prevent failure before it happens. Users can also estimate the accuracy of output according to the quality of input. Moreover, based on the common feature of low-quality inputs, engineers may discover the defect and weakness of the computer vision system and make improvement. However, the IQA of image quality for computer vision are facing new challenges. An image good for human visual system might not as good for computer vision systems and vice versa. Unlike human visual systems, which share the same physiological structure, each computer vision system has its own design and may bring about a unique standard of image quality. Making a specialized quality standard for each of them individually is an elaborate work, and thus it is meaningful to find a common way to generate unique standards for different computer vision systems.

In order to cope with the unknown quality standard and the variety, Artificial neural network (ANN) and Convolutional neural network (CNN) can be applied.

ANN is a historical concept which has become reality in recent years, thanks to the dramatic improvement of computing power. ANN works as a simulation of biological neural networks in animal brains. Different from traditional state-of-the-art algorithms, researchers do not directly work out the regression function but design the architecture of ANN instead. Then, ANN itself learns the regression function from a labeled training dataset by back-propagation. This feature gives ANN superior capability in dealing with

Table 1.1: Experimental Models

	Type	Depth	Input Size	Quantization
DAQ1	Patch Based Classifier & Weight Learner	3	$32 \times 32 \times 1$	MAE
DAQ2	Single CNN	5	$144 \times 96 \times 3$	MCC

complex problems in which a regular pattern is hard to find for human beings.

CNN is a special ANN that has each neuron connected to a spatial domain of certain size and shares weights across different neurons (usually all neurons). Nowadays, CNN has been widely used in image recognition, neutral language processing and artificial intelligence.

Back to our problem. In the past few years, neural network has also been brought into image quality assessment and showed its power in NR-IQA. Moreover, neural network has the flexibility to fit into different visual systems without changing its architecture. By changing the training data set, a neural network can act as a standard generator, which learns a new unique quality standard each time during training. Hence, we propose to design a neural network DAQ (deep algorithm quality) as a generalizable quality assessment tool under both human visual system and computer vision systems.

1.2 Contribution

In this thesis, we review the state-of-the-art IQA techniques as well as the challenges. To address the challenge of IQA for computer vision systems, we propose 2 IQA algorithms based on CNN , which can provide IQA to both human visual and computer vision systems, named as DAQ1 (Deep Algorithm Quality 1) and DAQ2.

DAQ1 follows patch based image classifier design. It employs a shallow CNN combined with a patch weight learner for better simulating human visual system. On the other hand, DAQ2 is a larger and deeper CNN that takes the whole distorted image as its input. DAQ1 and DAQ2 also quantize the image quality for computer vision system in a different way with respect to their architecture.

Both DAQ1 and DAQ2 achieve competitive result on simulating human visual system, compared to existing methods. Although the result on computer vision system is not good as that on human visual system, both model still show certain prediction power. Using CNNs is presumably a good way to measure image quality for computer vision systems.

1.3 Thesis Outline

The reminder of this thesis is organized as follows:

Chapter 2 gives a general background introduction to IQA and neural networks. In addition, we also review and discuss the related works in IQA field.

In Chapter 3, we show the problem formulation and experiments set up in detail. We aim at building a general proposed IQA model for both human visual system and computer vision system. Specifically, we quantize image quality for computer vision system using Mean Absolute Error (MAE) and Matthews Correlation Coefficient (MCC). Then we build 2 models based on different neural network architectures to test whether CNN can fulfill the task.

Chapter 4 is the evaluation of our model on benchmark datasets. We test each model on both IQA for human and IQA for computer vision system. LVIE-R2 and TID-2008 datasets are used as benchmark for human visual system. Miovision DAC detector and Miovision dataset are used as benchmark for computer vision systems.

Chapter 5 discusses the experiment results and draws conclusions.

Chapter 2

Literature Review

The background knowledge and related works are introduced in this chapter.

2.1 Background

The related background knowledge is introduced in this section. Subsection 2.1.1 gives a brief introduction to IQA and Subsection 2.1.2 explains the basic concept of Artificial Neural Network (ANN) and Convolutional Neural Network (CNN).

2.1.1 Image Quality Assessment

IQA is a complex problem in computer vision field. As the use of digital images grows significantly in past decades, maintaining the quality of images soon becomes a problem. Digital images are vulnerable to distortions on its way to end users. Fig. 2.1 shows some possible procedures that may cause distortions. During collection, the quality of image maybe affected by shooting condition like low illumination, jitter of lens and photography parameters. After that, the raw images are compressed for storage. Distortions also come from lossy compressions like JPEG. Some enhancement made to digital images, for instance contrast adjustment and sharpening may also bring certain distortion. After that, when images are transmitted to end users, mainly through the Internet, it may further suffer from the package loss and bit error during transmissions.

As end users require to get images with sufficient quality, the IQA techniques need to be implemented, which has three main kinds of usages. [14]:

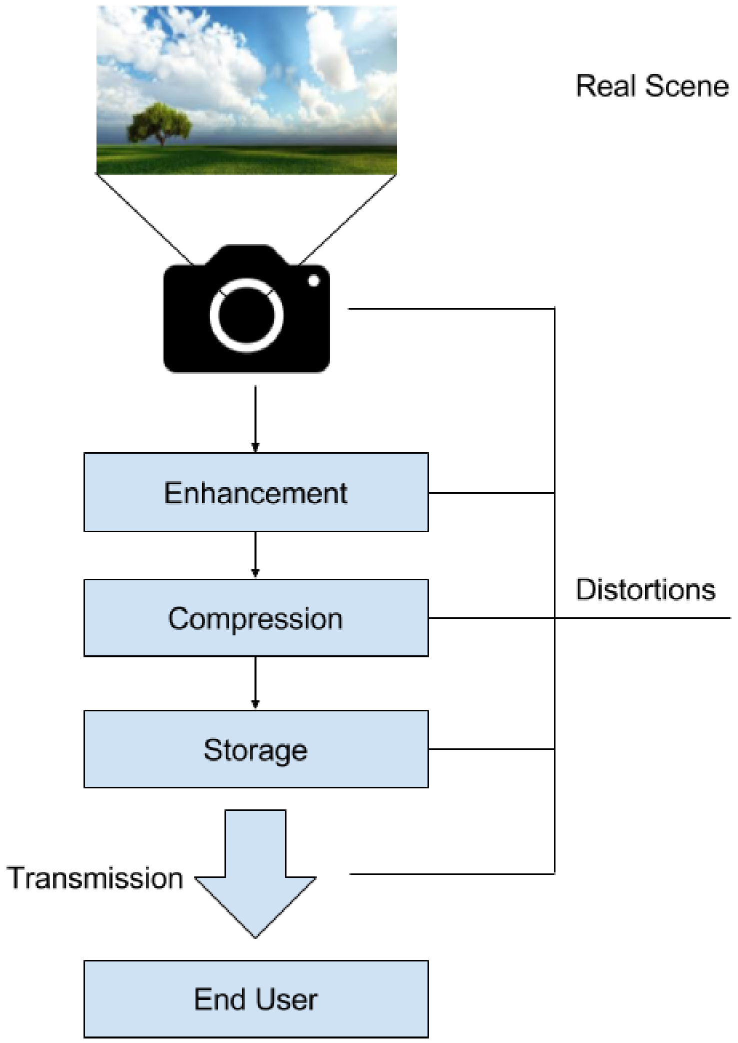


Figure 2.1: Digital images suffer from distortions on every step.

Table 2.1: ACR Scale

Rating	Label
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

1. IQA can be used to monitor image quality in quality control systems. For example, an image acquisition system can use a quality metric to monitor and automatically adjust itself to obtain the best quality image data. A network video server can examine the quality of the digital video transmitted on the network to control and allocate streaming resources. In light of the recent gigantic growth of Internet video sources, this application is quite important.

2. Second, IQA can be employed to benchmark image-processing systems and algorithms. For instance, if a number of image denoising and restoration algorithms are available to enhance the quality of images captured by digital cameras, a quality metric can be deployed to determine which provides the best quality results.

3. Third, IQA can be embedded into image-processing and transmission systems to optimize the systems and the parameter settings. For example, in a visual communication system, an image quality measurement can assist in the optimal design of the prefiltering and bit assignment algorithms at the encoder and of optimal reconstruction, error concealment, and postfiltering algorithms at the decoder.

Therefore, researchers in image processing and computer vision have worked on this subject, and various methods of IQA have been put forward.

Since in most cases, the end user of digital images is a human, the easiest and perhaps the most accurate way of measuring the quality of images is to have human annotators to grade the image. Thus subjective measurements appears. The most widely used subjective measurement is MOS (Mean Opinion Score). MOS is represented by a single rational number, usually from 1 to 5, where 1 stands for the lowest quality and 5 stands for the highest quality. The commonly used ACR (Absolute Category Rating) quality test method defines the score as Table 2.1 shows.

Usually, each test image is evaluated by multiple human annotators and the final MOS score is computed as:

$$MOS = \frac{\sum_0^N R_n}{N}, \tag{2.1}$$

where N is the number of human annotators and R_n is the rating given by the nth annotator. Thanks to the common structure of human visual systems and the extraordinary accuracy of it, MOS is a highly precise measurement of image quality and often works as a standard one.

However, the defect of MOS is also obvious: Human annotator is inefficient and expensive, which makes MOS powerless facing the millions of images that arise in the information age. A faster and economic method is needed. Therefore, objective measurements are invented.

The mean idea of objective measurements is to simulate human visual system using a series of algorithms, thus IQA can be done automatically by computers instead of human. A common classification of objective measurements is based on the access of an original image. An original image is a sample image which is considered as perfect quality. Fig. 2.2 shows an example, (a) is the original image and (b), (c) and (d) are distorted version of it. Objective measurements are classified into 3 categories [14]:

1. Full-Reference (FR) IQA, or image similarity/fidelity measurement considers the original image as fully accessible. Distortions can be determined by comparing the original image and distorted images. FR-IQA methods are often used as a benchmark algorithm that tests whether an image compression or enhancement procedure maintain the quality of original image. However, in quality control systems, especially in industrial environment, a perfect original image does not exist, which makes FR-IQA limited in usage.

2. Reduced-Reference (RR) IQA is more like an extension of FR-IQA. RR algorithms are based on some certain features extracted from the original image. As shown in Fig. 2.3, a RR-IQA system transforms extracted features separately with original image and uses the features to evaluate the quality of received image. Transforming only the features saves bandwidth, thus RR-IQA works well on transmission quality assessment where a FR-IQA cannot afford. However, RR-IQA is still limited by the prerequisite that there exists an original image.

3. No-Reference (NR) IQA is designed for such tasks without access to original image. NR-IQA only relies on the inner features of a distorted image, like a human end user

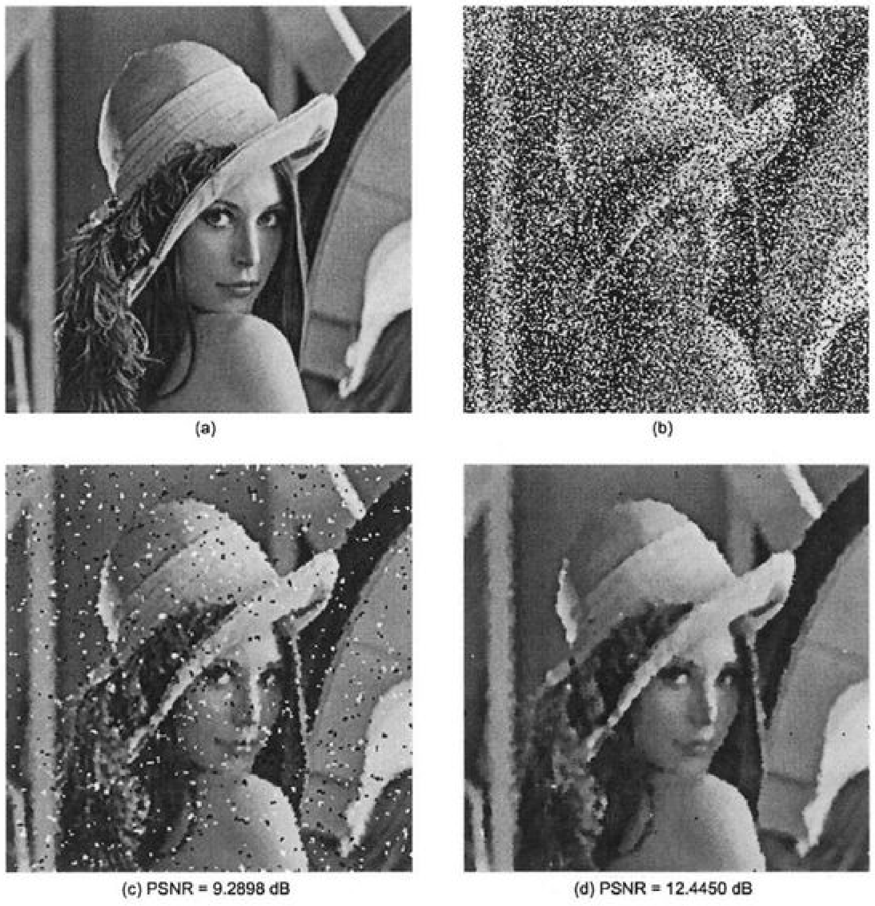


Figure 2.2: Perhaps the most famous reference image: Lenna

does. However, it is proved to be a very challenge task since the procedure taken by human brain is still a mystery to scientists.

The simplest subjective measurement is MSE (Mean Square Error), which is a FR-IQA which still being widely used in today's quality assessment systems owing to its low computational cost, in spite of its poor correlation to MOS. MSE can be defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2, \tag{2.2}$$

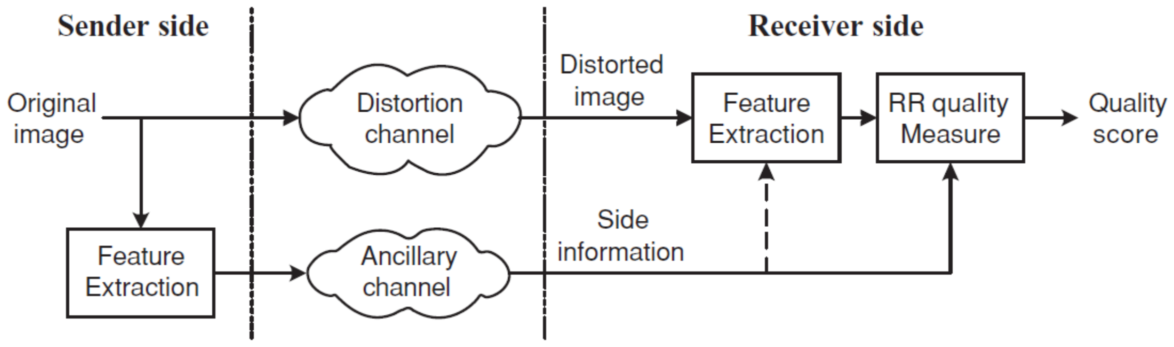


Figure 2.3: A typical diagram of RR-IQA system.

where x is an original image which treated as perfect quality and y is its distorted version. x_i and y_i stand for the ith pixel in the image and N is the total number of pixels.

The defect of MSE is obvious: it concerns no spatial factor. MSE is also known as noise power, only keeps track of the intensity of noise. In fact, the spatial distribution and pattern of noise matters a lot. An example given by [14] is shown in Fig. 2.4. Two different noise signals are added to the same original image. One has the same values on all pixels, the other one has random positive or negative value with the same absolute value. The distorted images (b) and (c) get the same MSE since the absolute error intensity are the same, but have an obvious difference in perceptual quality.

Modern state-of-the-art subjective measurements have much more sophisticated structure. As shown in Fig. 2.5, a variety of preprocessing and channel decomposition method are applied to the image. The decomposed channels are evaluated separately and summed up in the end. Usually the error summation is implemented by a Minkowski metric[15]:

$$E = (\sum_l \sum_k |e_{l,k}|^\beta)^{\frac{1}{\beta}}, \tag{2.3}$$

where $e_{l,k}$ is the normalized error of the k-th coefficient in the l-th channel, and β is a constant exponent typically chosen to lie between 1 and 4.

Plenty of efforts have been made into improving the error weighting and error masking of different channels. However, these state-of-the-art models still suffer from some common defects inherited from MSE because they were built upon following assumptions which might not be true[16]:

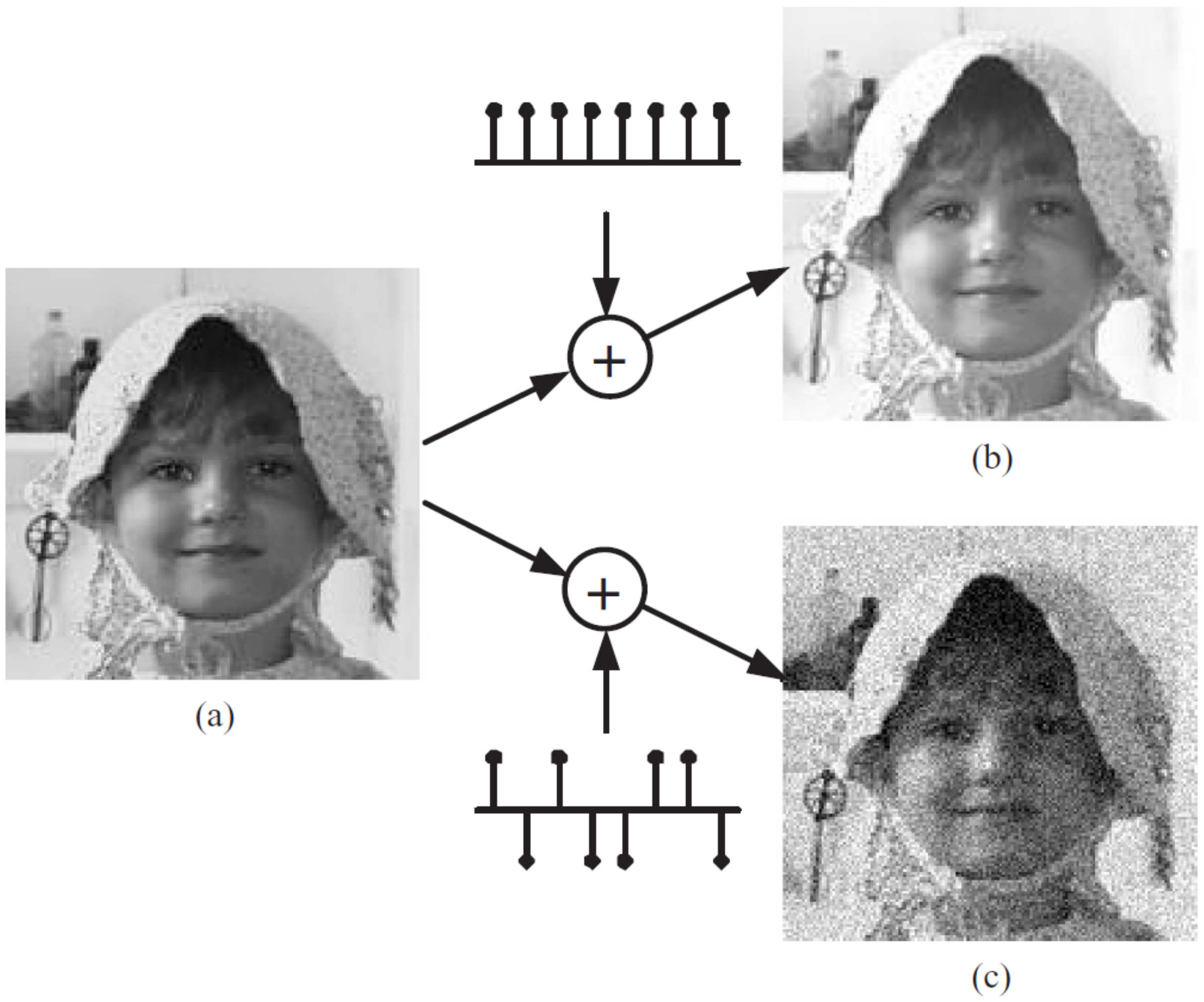


Figure 2.4: Two images with the same MSE may have very different quality.

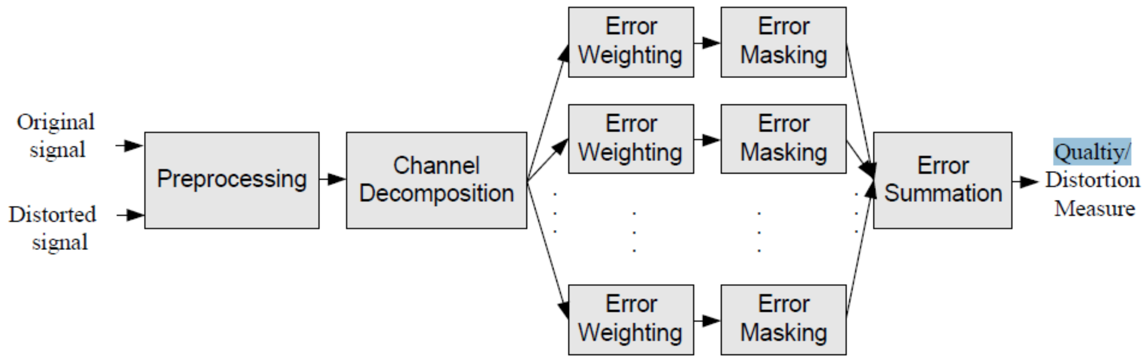


Figure 2.5: Structure of common state-of-the-art IQA system.

1. In each channel, after weighting and masking, the interaction between different coefficients is small enough to ignore;
2. The interaction between channels is small enough to ignore;
3. The perceived image quality is determined in the early vision system. Higher level processes, such as feature extraction, pattern matching and cognitive understanding happening in the human brain, are less effective;
4. Active visual processes, such as the change of fixation points and the adaptive adjustment of spatial resolution because of attention, are less effective.

The suspicion on such assumptions leads to the invention of structure based image quality measurement. Among structure based IQA, convolution neural network is one of the most ingenious and promising method which recently achieves convincing results.

2.1.2 Neural Networks

An Artificial Neural Network (ANN) is not a brand new idea. It was firstly introduced by Warren McCulloch and Walter Pitts [17] in 1943 as a computational model of ”nerve net” in human brain. After that, the concept and architecture of neural networks are further developed by follow-up researchers. For a long time, neural networks were constrained by the performance of hardware. Not until recent decades, the advancement in GPU design and brain science lead to a boom in the development of neural networks.

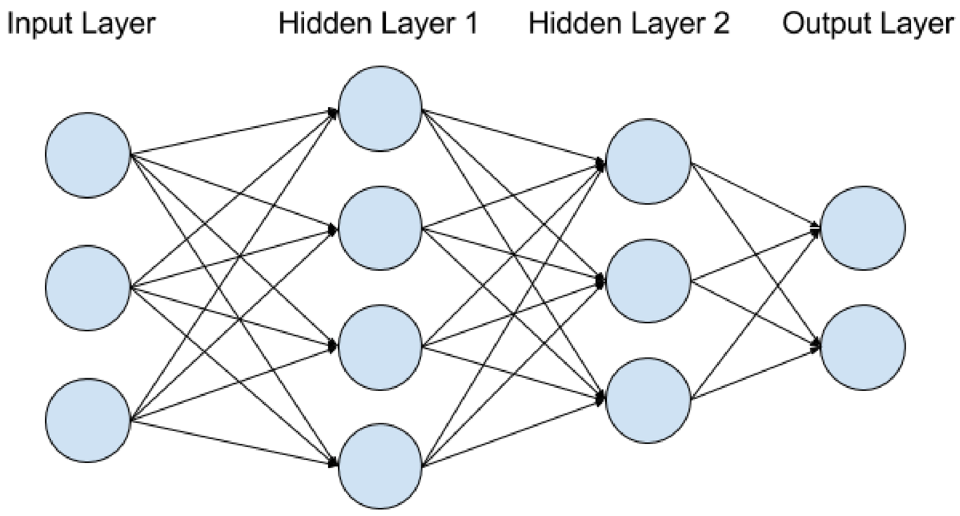


Figure 2.6: The basic structure of a Neural Network: neurons form multiple layers and the neurons in each layer are connected to the neurons in the next layer.

A common modern neural networks consist of a large number of nodes called neurons. Each neuron does a simple calculation, usually $y = Wx + b$, where W is called Weight and b is called Bias. The neurons form multiple layers and the result value y of each neuron is then passed to the neurons in the next layer. The first layer is called input layer as shown in Fig. 2.6. As its name implies, it takes features from outside the network as input. The last layer is output layer and its output value is the prediction given by the neural network.

2.1.3 Training of Neural Networks

At the beginning, Weight Metric and Bias Metric contain random values (or pre-trained value obtained from other benchmark data). To adjust those parameters to fit into a particular task, a training of the neural networks is needed.

The most common and popular method of training neural network is back-propagation (BP) [18]. The goal of back-propagation is to compute the partial derivative, or gradient, $\partial E/\partial w$ of a loss function E with respect to any weight w in the network. The loss function E calculates the difference between prediction of neural network and its expected output, after one or a batch of sample data go through the network. A loss function is usually defined as:

$$E = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2, \tag{2.4}$$

or

$$E = \frac{1}{N}\sum_{i=1}^{N}\left|f(x_i) - y_i)\right|, \tag{2.5}$$

where $f(x)$ is the equivalent function of the whole neural network. Equation 2.4 is called $L2$ loss while equation 2.5 is called $L1$ loss. In practice, $L2$ loss is the most popular one because it is more sensitive to examples that far away from expected output. Thus the trained neural network is hopefully more general. On the other hand, $L1$ loss is not that sensitive to a minority of output that far from the expectation and takes care of the average error of the majority. It is especially useful when training data is not very carefully collected and may contain incorrect samples.

Thus the progress of training by BP can be presented as:

1. Put one/a batch of training data through the neural network;
2. Calculate the loss between output and ground truth;
3. Go backward the network and calculate the partial derivative, or gradient, $\partial E/\partial w$ of loss function E with respect to each weight w in the network;
4. Update the weights in the network according to loss, gradient and learning rate (LR);
5. Repeat step 1 to 4 until training ends, usually when a certain number of cycles set by researcher is reached or the loss value is smaller than a threshold;

This method is called "back-propagation" partly because the partial derivative is calculated using chain rule:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}, \tag{2.6}$$

where E is the loss function, w_{ij} is the weight from neuron i to neuron j, o_j is the output of neuron j, net_j is the weighted sum of outputs to neuron j from the previous layer.

We can calculate them one by one from the output layer to input layer and use the result in later layers for calculating the former layers. Therefore, calculating partial derivative is actually quite cheap when doing backward.

A neural network with multiple layer structure has proved its power on image recognition [19]. However, it suffers from "the curse of dimensionality" heavily. It means that the number of parameters in the network goes up quickly when the dimension (resolution) of input image increases. Early neural networks work on low resolution images such as 20×20 and 32×32. Early benchmark datasets, MNIST [20] and CIFAR10 [21] for instance, are also collections of small images with 20×20 and 32×32 pixels. At that time, neural networks can take care of those images with hundreds or thousands of parameters. When the size of target image rise to around 200×200, an input layer with 40000 neurons is needed. Assuming the first hidden layer is fully connected and has the same number of neurons as the input layer, which is quite common in practice, at least 40000×40000 individual parameters is needed in just 2 layers. The mass of parameters not only consumes computing resources, but also causes serious overfitting problems.

Overfitting means that a statistical model tries to describe each training sample rather than to find out regular patterns among the sample collection. Fig. 2.7 shows a simple case of overfitting. The regression function tends to "remember" the distinguishing features of each sample individually but fails to figure out the trend of all samples. Although it passes every sample point and has 0 loss, it is not generalizable to unseen data. Even a linear function has more prediction power than it.

Overfitting usually happens when neural network has too many parameters compared to the number of training samples, which enables the network easily remembering all samples. To limit the number of parameters in neural network, researchers find a way to reuse parameters in different parts of the image, which is a Convolutional Neural Network (CNN).

2.1.4 Convolutional Neural Networks

A CNN is one type of neural network that specially designed for image recognizing. The architecture of CNN comes from the organization of animal visual cortex. Fig. 2.8 shows the basic structure of a convolutional neural network. $X_{i,j}$ represents the pixels in the input image. A is the kernel of the first convolutional layer and is repeatedly used on each block of four pixels. The neurons on the second layer then takes the outputs of the first layer as their inputs and use the same kernel B. Fig. 2.9 shows the mapping between 2 layers. One blocked in the front layer, which is a $m \times n \times d_1$ tensor ($m = n$ in most cases),

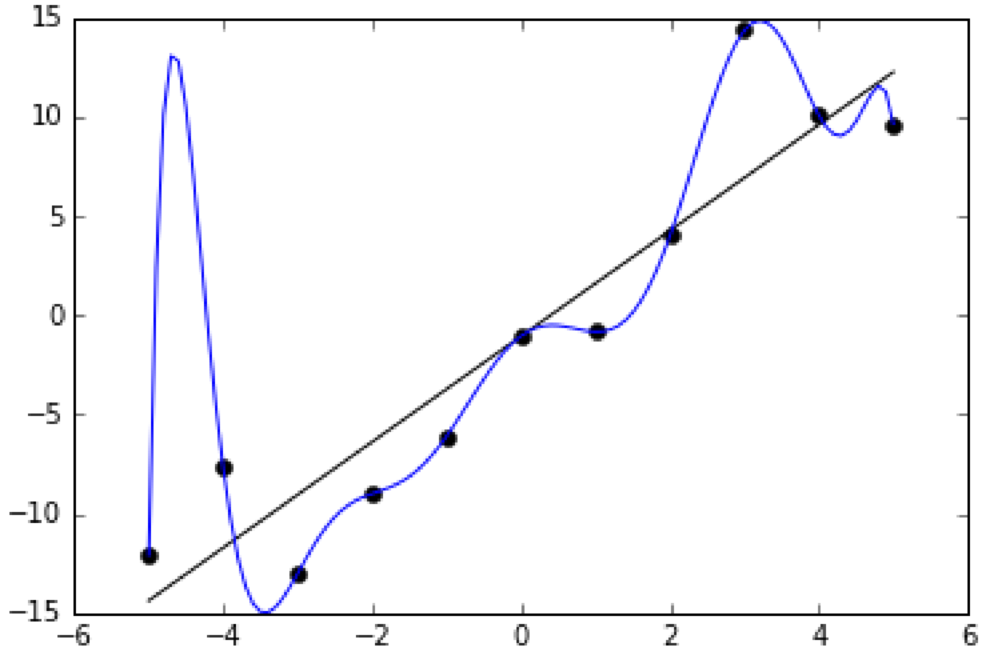


Figure 2.7: The polynomial function overfit the training samples. It "remembers" each sample instead of learns the general relationship [2].

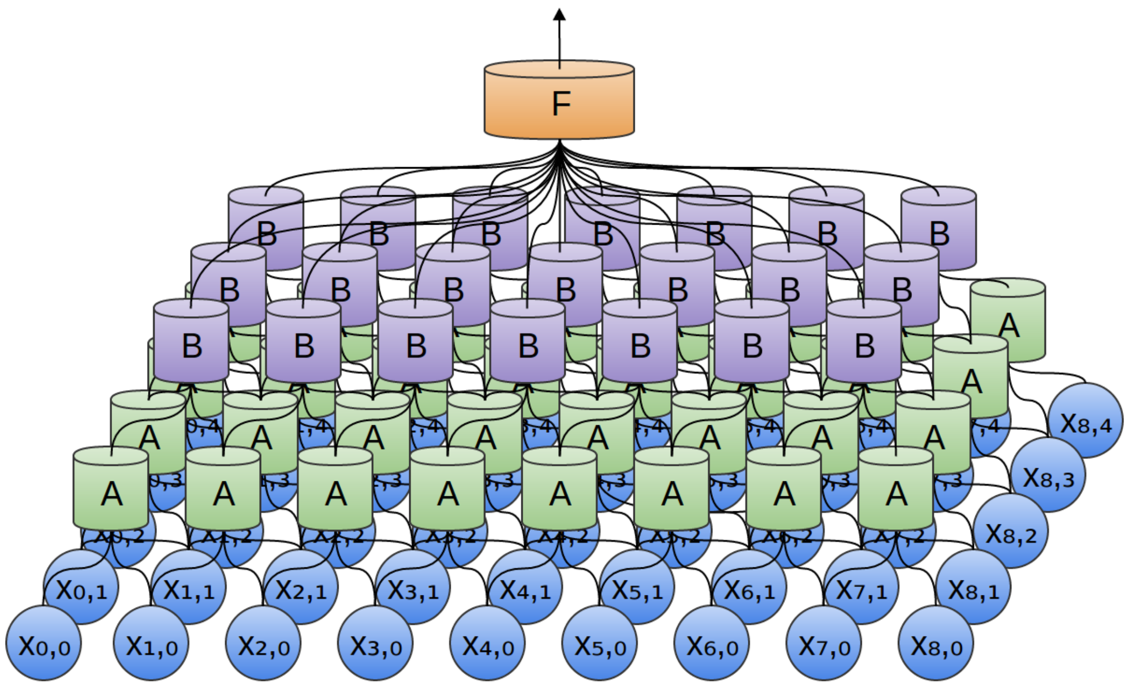


Figure 2.8: CNN reduces the number of parameters needed in high resolution images by sharing the kernel on each layer [3].

is multiplied by a $m \times n \times d_1 \times d_2$ kernel and mapped to a $1 \times 1 \times d_2$ block in the next layer. The $m \times n \times d_1$ block in the front layer is called receptive field, which means all neurons in such block is connected to one neuron in the next layer, and their information is gathered together by a neuron in next layer.

CNN is proved to be very efficient in pattern recognition and other image classification tasks. Its superior performance comes from some particular features. The most important feature is perhaps its spatial invariant. Since the same kernel is used repeatedly in the whole input space, it can detect its corresponding pattern no matter where the pattern shows up. This feature significantly reduces the number of patterns the network needs to learn.

Another important feature is its ability of abstracting and concentrating information. In Fig. 2.8, each neuron A (instance of kernel) on first layer accesses information from 4 pixels. On the second layer, each neuron B connect to 4 neurons in the first layer, which means it can access information gathered from 9 pixels in the input image. As the network

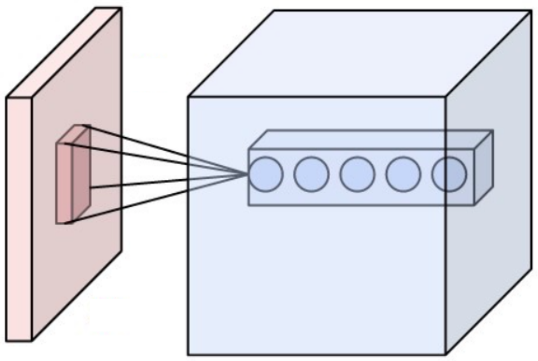


Figure 2.9: Kernel is an operator that maps $m \times n \times d_1$ block in previous layer to an $1 \times 1 \times d_2$ block in next layer [4].

goes deeper, the neurons in later layers get access to larger area of the input image. At last, at the final layer, the network gets an overall abstract sense of the input image. All of these concentration and abstract procedure are learned automatically by back-propagation. It is still a mystery to researches that how those things exactly happen because the mid product of hidden layers are really difficult to understand by human beings.

ReLU Layers

ReLU layers usually stand between 2 convolutional layers. ReLU stands for Rectified Linear Units. ReLU layer applies the non-saturating activation function to the outputs of convolutional layers:

$$f(x) = \max(0, x). \tag{2.7}$$

ReLU layers are very simple but they efficiently add nonlinear properties to the decision making function of the overall network as well as the sigmoid function:

$$f(x) = 1/(1 + e^{-x}), \tag{2.8}$$

and the hyperbolic tangent function:

$$f(x) = tanh(x). \tag{2.9}$$

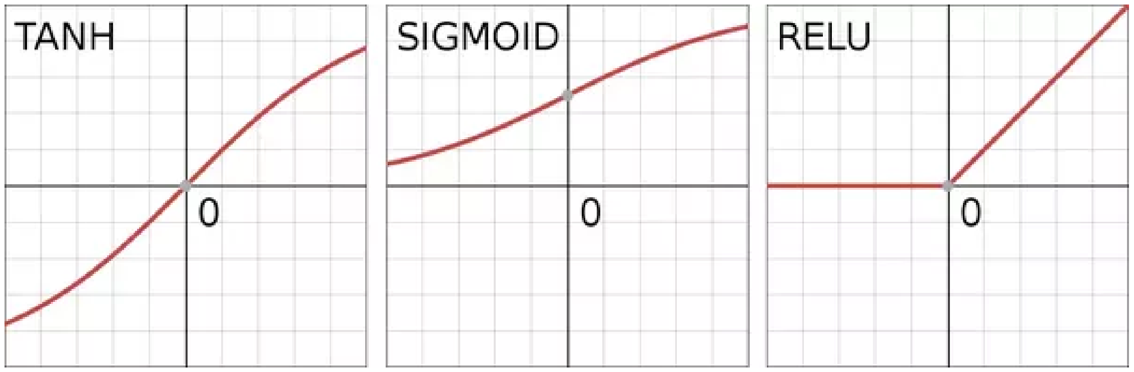


Figure 2.10: Common nonlinear functions used in CNN: ReLU, Sigmoid and hyperbolic tangent.

Fig. 2.10 shows the response of 3 methods. The 3 methods share the same idea of inhibiting negative outputs and amplifying/keeping positive outputs of the early layers, which is a simulation of how human brain cells work. Hyperbolic tangent function and sigmoid function are widely used in old models but ReLU function becomes more preferable recently because it is proved to be much computational cheaper without making any significant differences in accuracy [22].

Pooling Layers

"Pooling" is a nonlinear down-sampling method widely used in CNNs. Fig. 2.11 shows a common max pooling layer with a 2×2 filter size and a stride of 2. The filter move through the entries with a certain stride, pooling layer maps each block in former layer to a single value. Pooling layer concentrates the information in former layer and provides the later layers a larger "vision" in the original image. Also, pooling helps reducing the number of parameters in the network and hence has an effect of overfit control. The most popular pooling methods are max pooling and min pooling, where the filter takes the max or min value in each block as the output. Average pooling, which uses the average of all values in the block as output is also commonly used in old days. However, it has given its place to max pooling since the later one is proved to work better in practice [23].

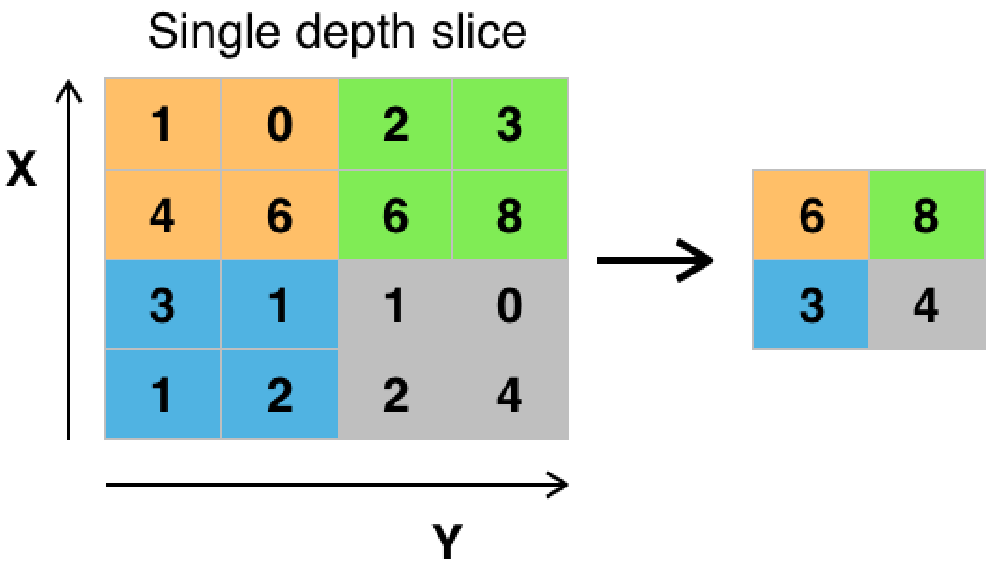


Figure 2.11: A max pooling layer with a 2×2 filter size and a stride of 2 [5].

2.2 Related Works

Kang et al. [6] applied CNN to NR-IQA. In their work, a CNN consisting of 1 convolutional layer, 1 feature pooling layer and 2 fully connected layers is trained to act as a general regression model, as shown in Fig. 2.12. The convolutional layer is used as the feature learner and the fully connected layers are used to learn the regression from features to quality score. The targeted large image is cropped into non-overlapped 32×32 sub-regions called patches and each patch goes through the model individually. During training, every patch is labeled by the score of image it comes from and the loss is defined as the MAE between predicted scores and truth scores. On contrary, during testing the average score of all patches from the same image is considered as the score of the image.

LIVE dataset release 2 and TID-2008 dataset are chosen as training and testing data. The network is tested on 5 kinds of distortions: JP2k compression (JP2K), JPEG compression (JPEG), White Gaussian (WN), Gaussian blur and fast fading (FF). The model gets a convincing correlation coefficient between the predicted quality score and the truth quality score. It is worth noting that all 5 kinds of distortions are global distortion that applied equally on every patch. This feature makes it possible to use the quality score of

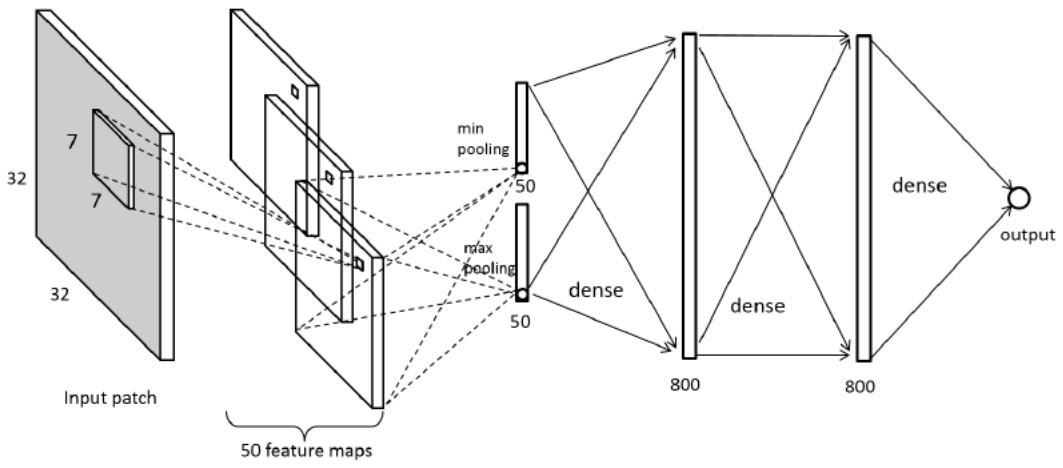


Figure 2.12: CNNIQ Network Architecture [6].

whole image as the score of every single patch during training.

Based on Kang's work, Li Jie, et al [24] approached to improve the performance by introducing an individual weighting score to each patch instead of using a simple averaging. They suggest that human visual system has the property of being more sensitive to contour and edge information in the image and thus the patches with such features should be given more weights. As shown in Fig. 2.13, Prewitt operator combined with a Graph-based image segmentation algorithm is used to determine the weight of each patch. These patch weights are applied to patch scores by a weighted average at the output layer of the network. In this way their network achieves a better overall score than Kang's design on the same distortions.

Also inspired by Kang's work, Li Yuming, et al [25] proposed to improve the performance by building a deeper CNN. Since Kang's CNN only contains one convolution layer, it cannot take the whole large image as a single input and has to divide it into small sub-regions called patches. This brings inaccuracy to the label of each patch since they all use the same score from the original image. Although the distortions are global, patches are not affected in the exactly same degree. By using a deeper CNN with 4 convolution layers, their DCNN network takes the whole original image as the input. They test DCNN on LIVE dataset and DCNN also achieves a competitive result against state-of-the-art traditional algorithms. However, DCNN did not beat Kang's CNN on overall correlation score. The tradeoff in computing time due to a much larger network structure seems not

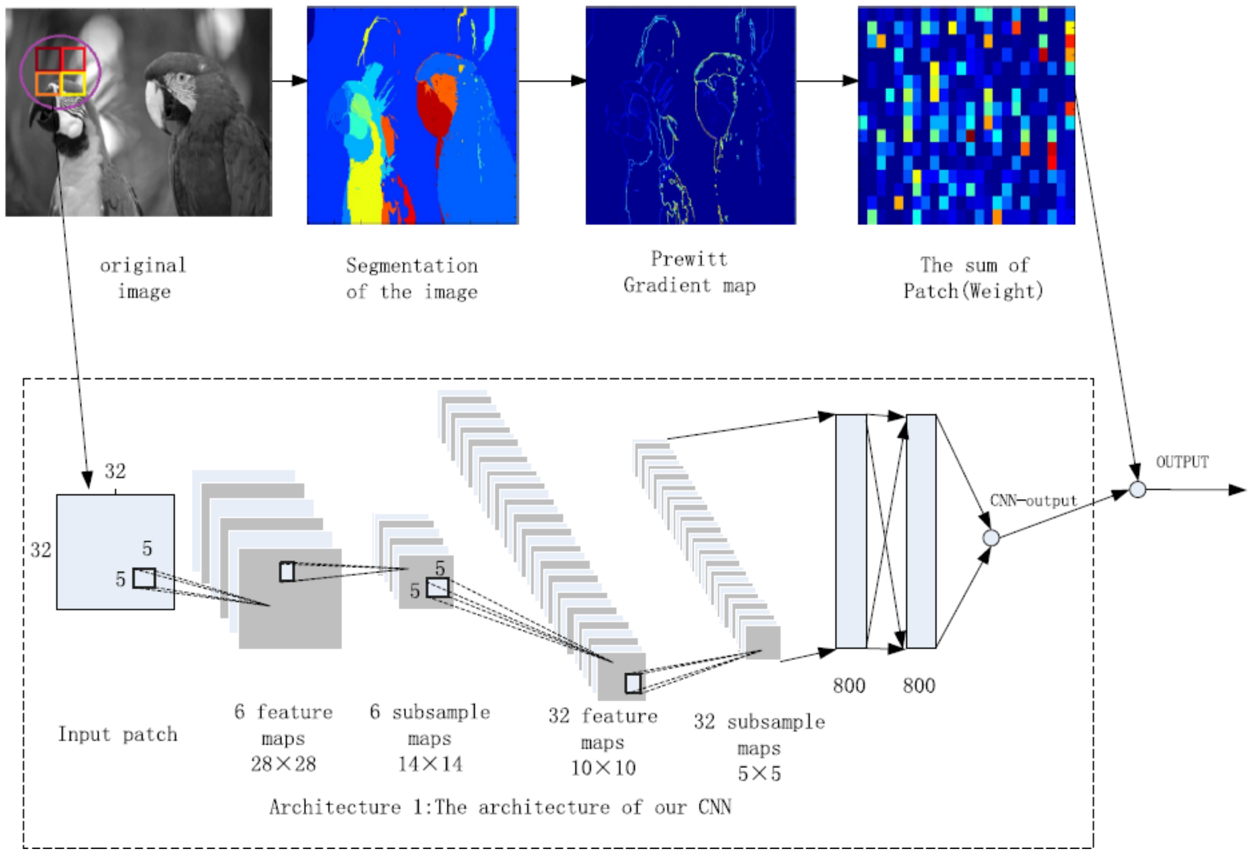


Figure 2.13: Based on CNNIQ network, Li's network applies Prewitt operator and Graph-based image segmentation algorithms to determine the weight for each patch [6].

that worthy.

Chapter 3

Experiment Setup

In this chapter we explain how we setup the experiments in detail. The sections are arranged in the order of experiment procedure.

Section 3.1 introduces the selection of experiment datasets and corresponding computer vision systems. Since our goal is to build a general image quality assessment model for both human and computer vision system, we also need benchmark dataset for both. For human vision system, we use LIVE-R2 and TID-2008, which are common benchmark datasets widely used in IQA experiments, including Kang's work [6]. By using the same benchmark datasets we can get a better comparison between our work and the existing ones. For experiment on computer vision systems, we cooperate with Miovision Technology Inc. and use their DAC (Decoupled Active Contour) detector as our test system. We use the Miovision Dataset, which is the benchmark input dataset for DAC, as our benchmark dataset for computer vision system.

Section 3.2 explains how we define image quality for a computer vision system. The benchmark quality scores commonly used in IQA experiments are usually based on the feeling of human reviewers, we cannot directly use them on computer vision system since they are not necessarily correlated to the performance of computer vision system (e.g. the detect accuracy of DAC detector in our case).

To make the benchmark images suitable for model training, certain pre-processing are needed, which are introduced in Section 3.3.

The design of models is described in Section 3.4. To evaluate the performance of CNN on IQA for computer vision system, we built 2 models for our experiment. The first model, DAQ1 is based on Kang's work [6] and Li Jie's improvement [24]. Kang's CNNIQ network already achieves an excellent score on its test datasets but still has an issue of simple averaging the scores of all patches. Li's work introduces a series of state-of-the-art method to determine the weight of each patch and outperformed the original CNNIQ network. We also propose to enable this feature on DAQ1 model. However, the state-of-the-art method of Li cannot be applied to IQA for computer vision systems since it is fixed. Therefore, instead of using state-of-the-art method, we built another neural network parallel to CNNIQ as our patch weight learner. Two networks are trained alternatively in our experiment. Another model, DAQ2, is inspired by Li Yuming's work [25]. We propose to test the performance of a deeper neural network using DAQ2.

Finally, Section 3.5 shows the implementation of models and Section 3.6 shows the training methods.

3.1 Data Selection

To evaluate the performance of models on measuring image quality for human visual system, LIVE dataset release 2 and TID-2008 dataset, both widely used in IQA experiments, are selected as benchmark dataset.

To evaluate the performance on measuring image quality for computer vision system, we use Miovision dataset as benchmark dataset and DAC model as the test computer vision system.

1. LIVE R2: Live R2 [26, 8], is composed of 844 degraded images derived from 29 reference images. The degraded images are generated by distorting each reference image with 5 types of distortions: JP2k compression (JP2K), JPEG compression (JPEG), White Gaussian (WN), Gaussian blur, fast fading (FF), where for each distortion type, the reference image is degraded with 7 to 8 degrees of degradation. Every one of the distorted images is annotated with Differential Mean Opinion Score (DMOS) in a span of 0 to 100, where 100 is the lowest and 0 is the highest quality.

2. TID-2008: TID-2008 [1] comprises 1700 degraded images derived from 25 reference images. The degraded images are generated by distorting each reference image with 17 types of distortions. All distorted image are annotated with Mean Opinion Score (MOS) in a span of 0 to 10, where 0 is the lowest and 10 is the highest quality.

3. Miovision Data Set: Miovision data set contains photographs taken from real traffic scene as well as the layout of vehicles annotated by human annotators. We picked

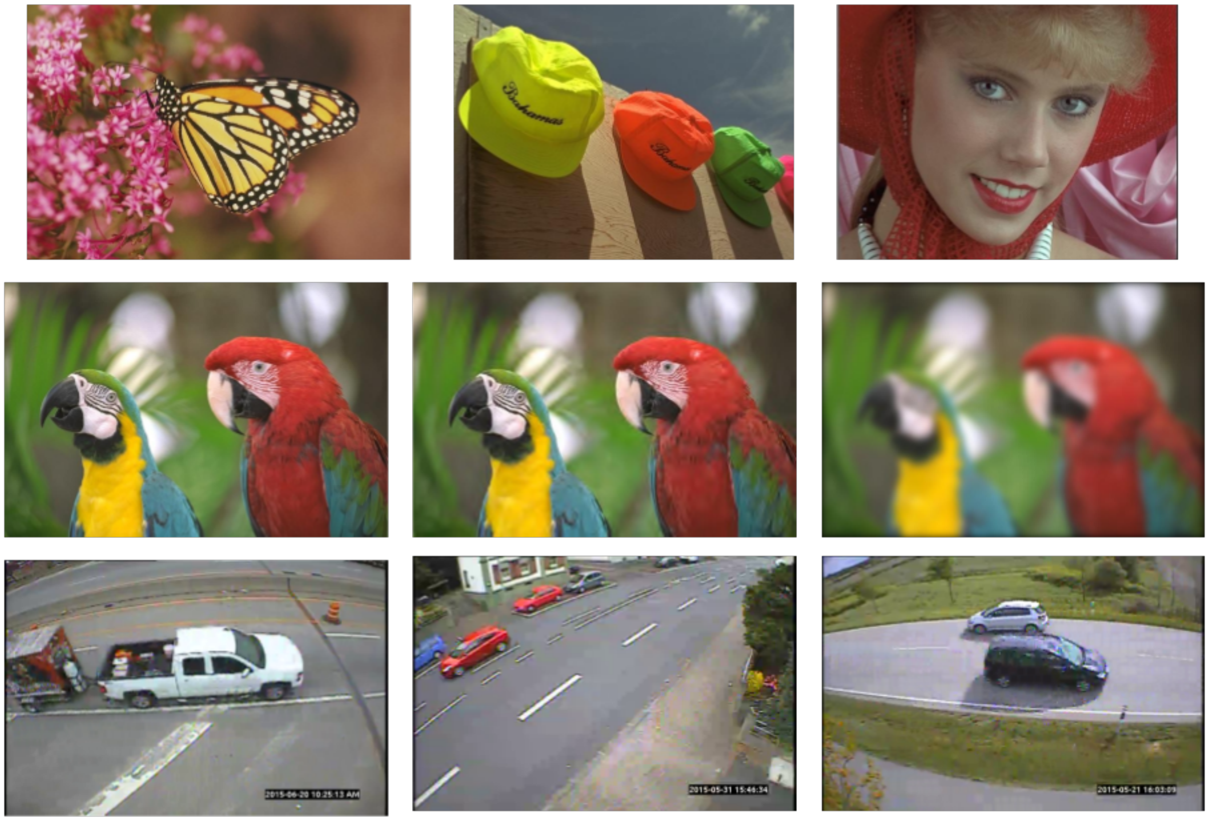


Figure 3.1: Upper: Images in LIVE-R2 and TID-2008 datasets. Middle: Reference image (left) compares to distorted images in LIVE-R2 and TID-2008 datasets. Lower: Images of real road scene from Miovision Dataset.

5000 of them used as experiment images. Another computer vision detection system called Decoupled Active Contour (DAC) is applied to experiment images and generates a set of probability map of where it detects a vehicle as Fig. 3.2 shows. The quality of image is defined as the pixel level error between human annotation and the predicted probability map.

Fig. 3.1 shows samples from benchmark datasets.

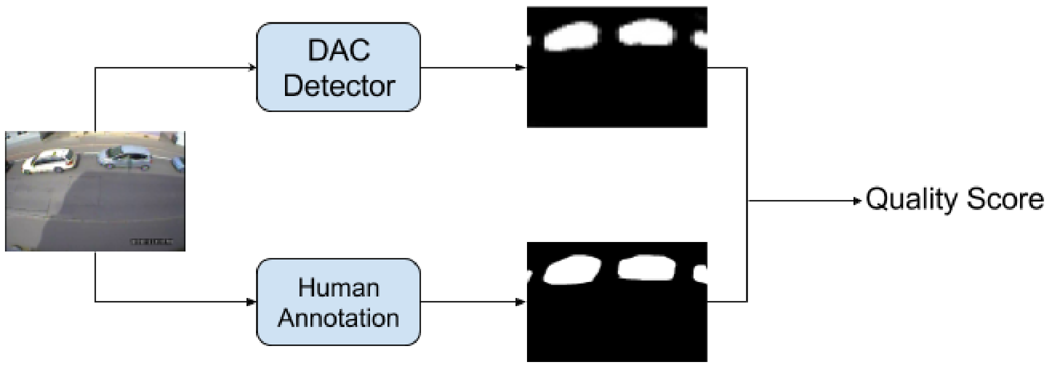


Figure 3.2: The measurement of quality score: A raw image is processed by the vehicle detector, DAC, and compared to annotated data to calculate quality score.

3.2 Quality Quantization

Images from LIVE-R2 dataset and TID2008 dataset are marked by objective score (MOS and DMOS), which is a good quantization of human feeling. Thus, we use it directly in the experiments related to human visual system.

As for computer vision system, there is no ready-made score in Miovision Data Set. Thus, we propose several approaches to quantize the quality. The reason we care about the quality of image is that it affects the information we could get from the image. In other words, image quality is something that measures how our visual system react to the image. In the same way, we can define the image quality for computer vision system as how well the system work on such image.

Therefore, we measure the performance of DAC detector on every image and define it as the quality of image. As shown in Fig. 3.2, every sample image goes through DAC Detector (the computer vision system) and produces a probability map of where is the vehicle and where is the background. Meanwhile, a human annotator views the image and creates a true layout graph of the vehicles.

The probability maps output by DAC detector are 31×31 black and white image that represents the probability of each pixel belongs to a vehicle with a variance from 0 to 1. The human annotations are white colored vehicle outline (with all pixels valued 1) under complete black background (with all pixels valued 0), which have the same size as source images, usually 720×480. To compare the probability map with human annotation, it needs some pre-processing. Firstly, the probability map is resized to the size of human annotation by bilinear interpolation[27, p.123]. Then the resized map is binarized to 0 and 1 with a threshold of 0.5. For each pixel x_i in the resized map:

$$x_i = \begin{cases} 1, & \text{if } x_i >= 0.5, \\ 0, & \text{if } x_i < 0.5. \end{cases} \tag{3.1}$$

After the pre-processing we get a resized map which has the same style with the annotations. Then we measure the performance of DAC detector, which will later be used as image quality score, in 2 different ways.

The simplest method is using the number of unmatched pixels as the quality score. In this way the best score is 0 and the worst score is the total number of pixels:

$$Q = \sum_{0}^{N} |x_i - y_i| \tag{3.2}$$

where Q is the quality score of source image we defined, N is the total number of pixels, x_i is the value of ith pixel in human annotation and y_i is the value of corresponding pixel in the resized probability map.

In this way, the range of score fluctuates depending on the size of image. Therefore, computing the average error rate of pixels would be a better quantization that ranged from 0 to 1, where lower is better. A drawback of these methods is that images with more area occupied by vehicles are more likely to have more unmatched pixels. The reason is that the layouts of vehicles drawn by human annotators are not exactly the true layouts and more vehicles always lead to more misclassified pixels.

Another method is to treat the detection as a binary classification that each pixel is either belongs to vehicles or background. Then the quality score can be generated by statistical analysis methods such as F_1 score (the harmonic mean of precision and recall) and MCC (Matthews correlation coefficient) score [28]:

$$Q = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \tag{3.3}$$

where TP, TN, FP, FN stand for true positive, true negative, false positive and false negative, respectively. In our case, we define pixels with value 1, which represent vehicle, as positive and pixels with value 0, which represent background, as negative.

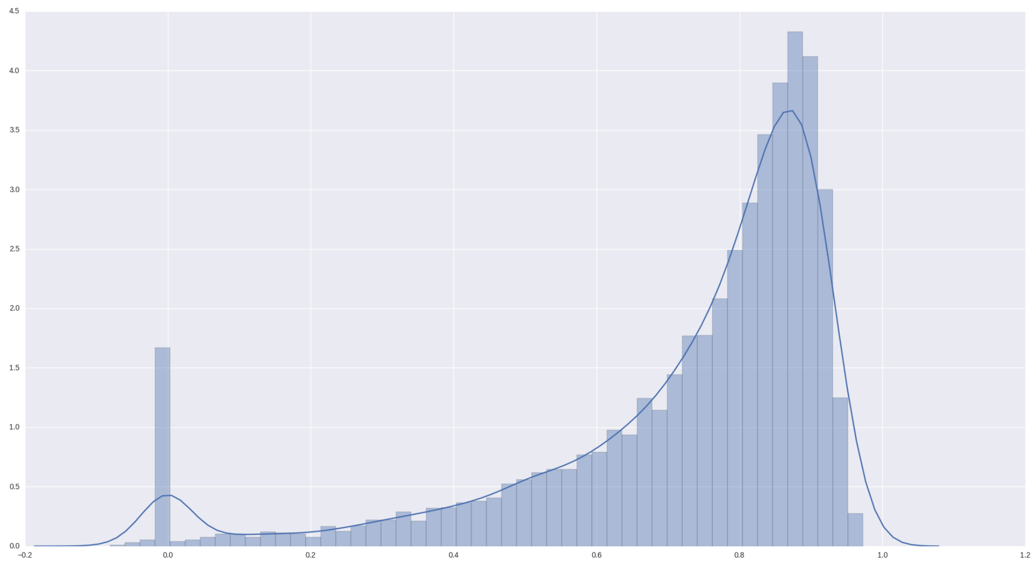


Figure 3.3: The distribution of MCC score on all images in Miovision Dataset, most images are of good quality.

These methods get rid of the correlation between area of vehicle and quality score. However, they have a problem with images that contain no vehicle. In this case, there is no positive sample and true positive rate is 0. Thus, the quality score is always 0 no matter what the prediction is. Fig. 3.3 shows the distribution of the MCC scores of Miovision Data Set. A lot of images are quantized to 0 score due to the problem, though most of them only have a few unmatched pixels.

In experiments, we tried both straight comparison and statistical analysis to generate quality score. The architecture of network also adjusted according to the score measuring method.

3.3 Data Pre-Processing

Before being input to the models, images are pre-processed for better learning effects.

3.3.1 Data Augmentation

To enlarge the experiment dataset and avoid overfit problem, data augmentation methods are applied to input images during training.

- Rotation:

  Input images are rotated randomly by 0, 90, 180, 270 degree. We only use these 4 kind of degrees to ensure the image keeping all information. Thus, it does not affect the quality score of the image.

- Flipping:

  Input images are random flipped horizontally or vertically. The scores remain the same.

- Shifting and Scaling:

  Shifting and Scaling are also common data augmentation methods. Both of them are widely used in training image recognition CNNs. However, DAQ models are aimed at IQA and applying them will affect the quality of images. Therefore, Shifting and Scaling are not used in our experiments.

3.3.2 Local Normalization

Following the instructions in [6], local normalization is applied to all input images of DAQ1 model. DAQ2 model directly takes the raw image. In our experiments, local normalization takes a window size of 7×7:

$$y_{ij} = \frac{x_{ij} - \bar{x}_{ij}}{\sqrt{\frac{1}{N}\sum_{m=i-3}^{i+3}\sum_{n=j-3}^{j+3}(x_{mn} - \bar{x}_{ij}) + 3}}, \tag{3.4}$$

where x_{ij} is an arbitrary pixel in raw image and y_{ij} is its normalized value. The constant +3 in the denominator is just a small constant to make the denominator not 0. $\bar{x_{ij}}$ is the arithmetic mean of all pixels inside the 7×7 window and $N = 49$ is the total number of pixels in the window. Therefore,

$$\bar{x}_{ij} = \frac{\sum_{m=i-3}^{i+3}\sum_{n=j-3}^{j+3} x_{mn}}{N}, \tag{3.5}$$

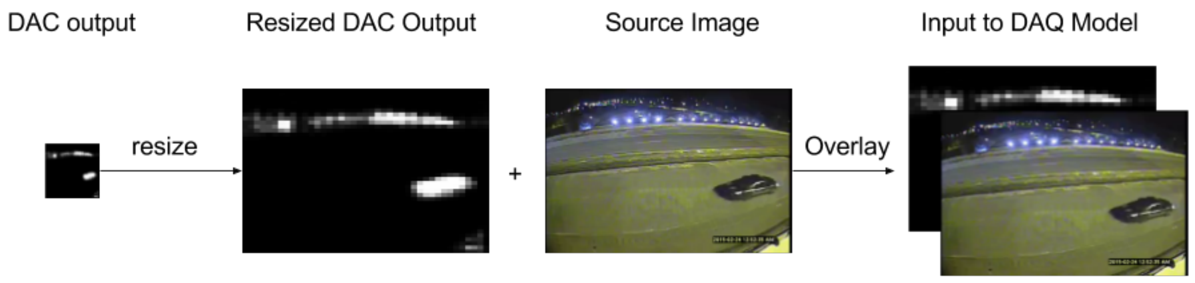


Figure 3.4: DAC detector output is resized and binded to the source image. The combination is then used as the input of our model.

3.3.3 Data Combination

To provide additional information for learning computer vision system's behavior (DAC detector's behavior in our case), source images are binded with the corresponding output of DAC detector before used as the input of DAQ1 and DAQ2 model. This method is only applied to the experiments on computer vision system since the output of human visual system is not accessible. This makes the models used in computer vision system IQA experiments slightly different from the models used in human visual IQA experiments. The first layer of CNNs has one more channel in depth for the output of DAC detector while all other layers remain the same.

As shown in Fig. 3.4, the output of DAC detector is a 31×31 probability map which shows the probability of each pixel being part of a vehicle. The output image is first resized to the size of source image and then overlapped with the source image as an additional channel.

3.4 Model Architecture

In the experiments we designed 2 different CNN. The first model is a very shallow network with only 1 convolution layer. It takes small sub-regions cropped from original images as input and uses average pixel error rate as the quality score. Its small size saves computing resources and enables fast running speed. These advantages make it suitable to work as an adjunct part of other computer vision systems for detecting bad inputs. The second model is a deeper network with 3 convolution layers. Its deeper structure allows it to take the whole image as the input. This enables it to use statistical analysis as the score measuring method.

3.4.1 DAQ1 Model

Motivated by Kang's [6] CNNIQ network and Li's patch weight design [24], DAQ1 uses a novel neural network to perform an importance sampling over a large number of patches to learn the patch weight based internal statistics. Under certain conditions, human see two patches differently while the current neural networks give the same score to both patches because they have the same content. Kang's [6] deep network can be more effective if the weights of patches could be given according to their importance rather than taking the average. Therefore, in Li's work they suggest that human visual system is more sensitive to contour and edge information thus they use Prewitt operator to determine the weight of each patch.

However, human vision is a complex system with a variety of habits that affect the sensitivity to an image. The sub-regions with certain contents might act as a more important role than others: A sub-region with an object such as a face in it would draw more attention than a random sub-region with random pattern. Also, patches with the same content might have different importance in different images: A tree in the dessert is more striking than a tree in the forest. These habits, apart from contours and edges, also affect people's feeling of the quality of an image. Under such conditions, Li's weight generating algorithm cannot cover enough features.

Furthermore, when applied to computer vision systems, the pattern related sensitivity does not follow human habits. Every computer vision system has unique architecture and parameters and in return has its own habits. Designing a weight generator for each of them separately would be a hard and elaborate work. Therefore, we use another neural network to learn the patch weights automatically during training.

The architecture of DAQ1 is described in Fig. 3.5. It consists of two sets of sub networks: local patch score learner (CNNIQ [6]) and patch weight learner. Every input image, already processed as Section 3.3 introduces, is cropped into a number of subregions called patches ($P_1 \cdots P_n$). Each patch goes through score learner and weight learner simultaneously and gets its score S_n and weight W_n. At last the score of image is computed by the weight average of all patches. In our experiments, the patch size is 32×32.

The score learner follows the original set up of CNNIQ network in Fig. 2.9, which consists of a convolutional layer, a max-pooling, a min-pooling, two fully connected layers and three rectified linear units(ReLU). The convolutional layer uses a filter of size 7×7

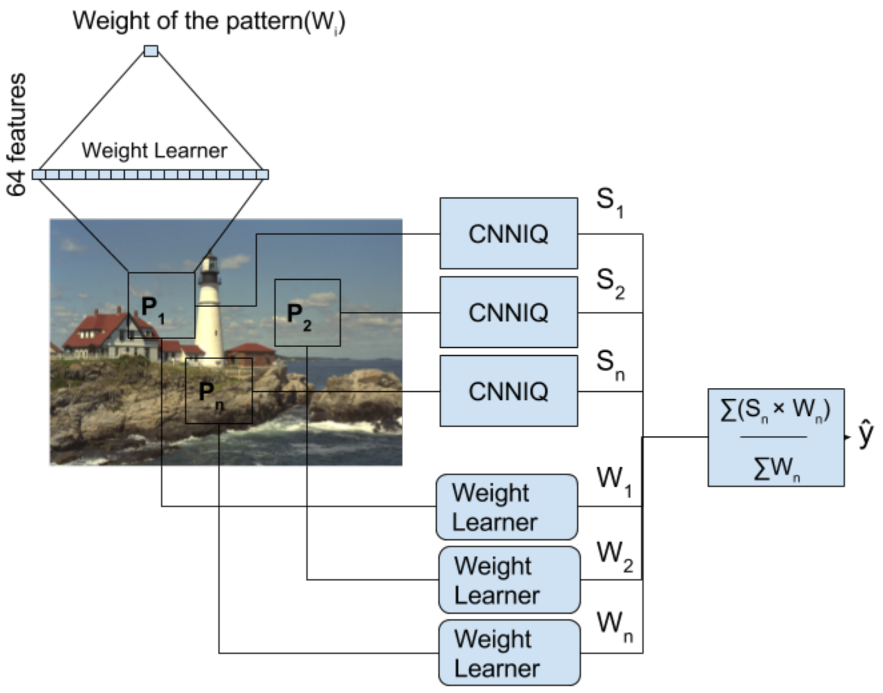


Figure 3.5: DAQ1 architecture. DAQ1 has two sub-networks: patch score learner CNNIQ [6] and patch weight learner.

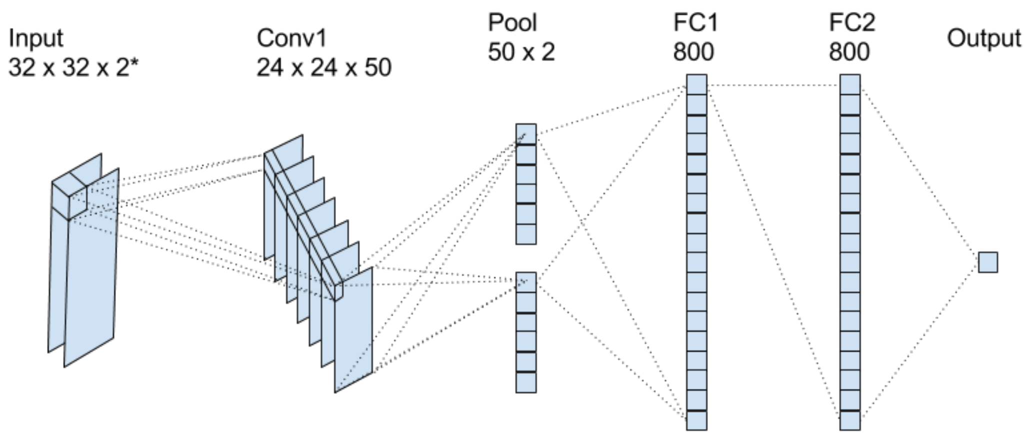


Figure 3.6: The architecture of Score Learner. *Input is $32 \times 32 \times 1$ for human visual IQA and $32 \times 32 \times 2$ for computer vision IQA.

and stride of width one. The size of the two fully connected layers are: $(100, 800)$ and $(800, 1)$.

The patch weight learner is a smaller network that uses two fully connected layers of size $(1024, 64)$ and $(64, 1)$.

The complete progress can be described as follows: For each input image, it is firstly cropped into n patches P_1, $P_2 ... P_n$ of size 32×32. Then each patch P_i goes through Score Learner and Weight Learner in parallel.

The architecture of the Score Learner is shown in Fig. 3.6. The input patch P_i is $32 \times 32 \times 1$ for human visual system and is $32 \times 32 \times 2$ for computer vision system with an addition channel for probability map information.

The first layer of Score Learner network, Conv1, is a common convolutional layer with a kernel size of 7×7:

$$y_{ijk} = \sum_{x \in P_{ijk}} w_{mnp} \cdot x_{mnp} + b_k, \tag{3.6}$$

where y_{ijk} is the output of neuron N_{ijk} on Conv1 at position $[i, j, k]$, P_{ijk} is the receptive field of N_{ijk}, x_{mnp} is the entry at position $[m, n, p]$ in P_{ijk}, w_{mnp} is the weight value of kernel at position $[m, n, p]$ and b_k is the corresponding bias value of kth feature (or kth layer of neurons in Conv1). The output of Conv1 has $24 \times 24 \times 50$ entries.

A pooling layer is set after Conv1. It consists of a max pooling and a min pooling. Unlike usual poolings with small filter size like 2×2 or 3×3, the max pooling takes a single max value from a whole layer through width and height dimension and so does the min pooling. Therefore both max pooling and min pooling have 50 entries.

$$z_k = \max(\{y_{ijk}\}), k \in [0, 49], \tag{3.7}$$

where z is the output of max pooling, i, j, k represent the position in width, height and depth dimension. The min pooling works as well:

$$z'_k = \min(\{y_{ijk}\}), k \in [0, 49]. \tag{3.8}$$

After the pooling layer, there are 2 common fully connected layers, FC1 and FC2, each has 800 entries. The max pooling and min pooling layers are joint together as a 100 entries vector and fully connected to FC1. A ReLU function $h(x)$ is applied to the input of FC1 and FC2 as the rectified linear unit to enable sparse activation:

$$h(x) = \max(x, 0). \tag{3.9}$$

Therefore the activation of FC1 becomes:

$$p_i = \sum_j w_{ij} \cdot h(z_j) + b_i, \tag{3.10}$$

where p_i is the output of ith FC1 neuron, z_j are the entries of the joint pooling layer, w_{ij} and b_i are the corresponding weight and bias of FC1. FC2 works in the same way.

$$q_i = \sum_j w_{ij} \cdot h(p_j) + b_i. \tag{3.11}$$

Finally the output of Score Learner becomes:

$$S_i = \sum w_j q_j + b_j, \tag{3.12}$$

where S_i is the score of ith patch and j is the serial number of FC2 outputs.

The weight Learner is a simple fully connected layer with 64 neurons that takes the 32×32 patch as the input. Since there should be no correlation between the weight of patches and the output of DAC detector, Weight Learner only takes the normalized image channel as input:

$$W_i = \sum w_j x_j + b_j, \tag{3.13}$$

where W_i is the weight of ith patch and x_j is the jth pixel value of the patch.

At last the patch weights are applied to patch scores and the image quality $\hat{y}$ is computed by a weighted average of all patches:

$$\hat{y} = \frac{\sum (S_i \times W_i)}{\sum W_i}. \tag{3.14}$$

During training DAQ1 uses MAE between ground truth perceptual quality y and estimated quality $\hat{y}$ as loss value:

$$AE = |y - \hat{y}|, \tag{3.15}$$

where y and $\hat{y}$ are MOS or DMOS score depending on the image data set used.

However, this network structure does not fit the quality score generated by statistical analysis such as F1 and MCC score, because they are not computed by a linear function and are not additive. A weighted average cannot be applied to such scores.

3.4.2 DAQ2 Model

The basic idea of this model comes from Krizhevsky's work [7], a small but efficient CNN which is used for object recognizing and achieved high accuracy. Naturally, we think that if a CNN could learn to recognize a car, it can learn to determine whether a car is well detected.

The DAQ2 network consists of 3 convolutional layers and 3 fully connected layers (including output layer) as shown in Fig. 3.7. Between every two convolutional layers, there is a ReLU layer, a pooling layer and a normalization layer, shown as Fig. 3.8. The pooling layer applies a max pooling with a filter size of 3×3 and has a stride of 2 in height and width, a stride of 1 in depth.

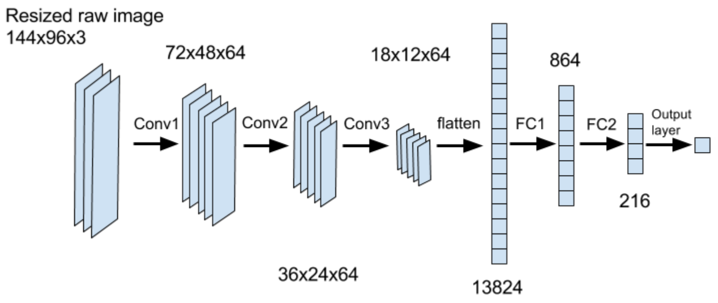


Figure 3.7: DAQ2 neural network. Using features from the 3 channel inputs (RGB), this network is trained to predict an estimated glsMCC score. (In experiments for computer vision IQA, input is 144 × 96 × 4. The additional channel is DAC detector output. See 3.3.3 for detail).

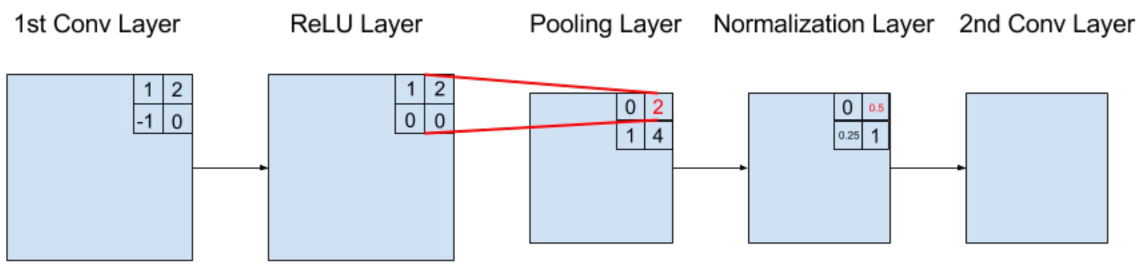


Figure 3.8: ReLU layers, Pooling layers and normalization layers are set between 2 convolutional layers.

Normalization Layers

Local response normalization [22] is applied to the output of pooling layers. Local response normalization follows the equation:

$$b^i_{x,y} = a^i_{x,y}/(k + \alpha \sum_{j=max(0,i-n/2)}^{max(N-1,i+n/2)} (a^j_{x,y})^2)^\beta, \tag{3.16}$$

where $a^i_{x,y}$ is the output of kernel i at position (x, y) and N is the total number of kernels. k, n, α, β are hyper-parameters that control the normalization. Local response normalization creates a competition status between neighboring neurons. Neurons with greater outputs inhibit neurons with smaller outputs in depth direction, which mimics the competition between brain cells of living creatures.

In the experiment, the parameters are set to $k = 2, n = 5, \alpha = 10^{-4}, \beta = 0.75$, following the instruction of the successful CIFAR-10 recognizing network [22].

3.5 Implementation of Models

Both DAQ1 and DAQ2 models are implemented using Tensorflow 1.0 framework [29]. A variety of frameworks have been developed for building machine learning models nowadays, including the most popular ones: Caffe [30], Theano [31], CNTK [32], Torch [33] and Tensorflow. Tensorflow is an open source framework presented by Google in 2015. It is a relatively new framework when we start our experiment. Compared to the early built models like Caffe, which is the first and most widely used industry-grade deep learning tool kit, Tensorflow have many advantages that make the design of new model much easier.

In Caffe, the smallest building block of networks is a layer. To build a new network, one has to define every unique layer inside the network including its full forward, backward, and gradient update methods, then add all layers to a separate proto file to form the network. This layer-wise design reduces the flexibility of framework when building new models. Tensorflow builds networks based on computational graphs instead. A computational graph is a symbolic graph of vector operations. Fig. 3.9 shows very simple computational graph which does some linear computation: $Output = W(x + y) + b$. This computational graph can be implemented by the following code:

```
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
```

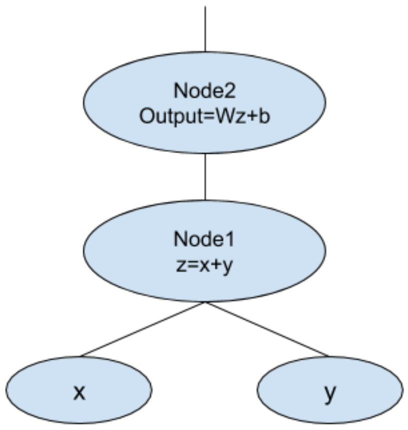


Figure 3.9: A simple computational graph of linear calculation.

```
z = x + y
W = tf.Variable([1], tf.float32)
b = tf.Variable([1], tf.float32)
Output = W * z + b
```

By simply changing the operation in each node here to matrix operations, a computational graph can represent a neural network with multiple layers. In this way, we can build a new network easily in Tensorflow. Newer frameworks like Theano and Torch also make use of symbolic graph while programming a network. However, Torch only provides Lua interface and Theano only provides python interface while Tensorflow supports both Python and C++. Therefore, Tensorflow is better for doing experiment in a script language as well as making further performance optimization in C++.

Considering the fact that our experiment would be performed on GPU, we only care about the performances of the frameworks on GPU. Since most deep learning frameworks including those we mentioned make use of NVIDIA CUDA Deep Neural Network library (CuDNN) [34], and call CuDNN for real computation, they get similar performance on GPU. Therefore, with advantages in other aspects, we pick Tensorflow as our framework used in the experiments.

The models are constructed as code in Appendix A and B.

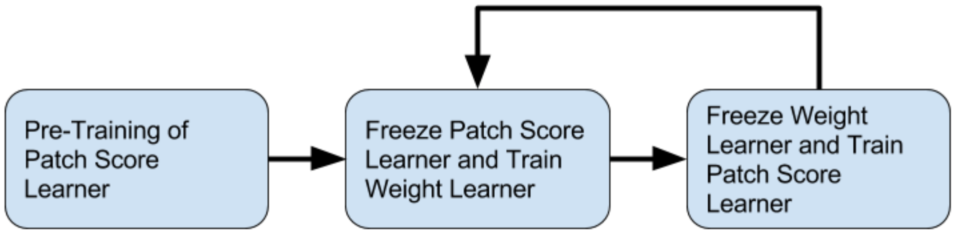

Pre-Training of Patch Score Learner
Freeze Patch Score Learner and Train Weight Learner
Freeze Weight Learner and Train Patch Score Learner

Figure 3.10: DAQ2 uses a train-freeze alternating training method between patch score learner and patch weight learner.

3.6 Model Training

3.6.1 Training of DAQ1

As Fig. 3.10 shows, DAQ1 uses an alternating optimization technique.

1. A CNNIQ network is pre-trained without a weight learner and used as the Patch Score Learner.
2. The parameters of Patch Score Learner are frozen and the Weight Learner is trained based on the prediction of Patch Score Learner.
3. The parameters of Weight Learner are frozen and the Patch Score Learner is trained with weighted average.
4. Repeat (2) and (3) until a minimum loss value or the maximum number of iteration is reached.

In the experiment, both models are implemented using TensorFlow framework. Both CNNIQ network and weight learner were trained by gradient decent optimizer with a learning rate of 10^{-4}. CNNIQ was firstly pre-trained by 200000 steps and then each network was trained by 1000 steps in turn for 20 times each. During the pre-training of

CNNIQ network, input patches were chosen randomly from all images in training set and each labeled by the score of corresponding image. During the alternating training, all patches used in one training step are chosen non-overlapped from the same image for the network to compute the weighted average.

3.6.2 Training of DAQ2

The training of DAQ2 is simpler. All images are resized to 144×96 and input to the model. The model is trained by gradient decent optimizer with a learning rate of 10^{-4} until 200000 steps.

Chapter 4

Results Evaluation

The performances of the proposed models are evaluated in two experimental set-ups. We first evaluate the effectiveness of DAQ models by evaluating the image quality blindly using two standard data-sets: LIVE-R2 and TID-2008. Second, the performance of DAQ models for computer vision based object detector is evaluated in production environment using Miovision's datasets.

The quantitative performance of DAQ models is evaluated using two metrics: 1) linear correlation coefficient (LCC); 2) Spearman's rank-order correlation coefficient (SROCC). Both LCC and SROCC are widely used performance evaluation matrices in IQA field, therefore we chose them for a better comparison with former works. For a fair evaluation, we have followed the evaluation criteria in reference [6], where LIVE-R2 data set is split into training (60%), validation (20%) and testing (20%) sets. The performance of DAQ models is compared against three NR-IQA methods (CNNIQ [6], CORNIA [35], BRISQUE [36]) and two FR-IQA methods (SSIM and PSNR).

4.1 Predicting Human Visual System

4.1.1 Performance on LIVE-R2

The performance of DAQ models compared to two reference based and three non-reference-based methods is demonstrated in Fig. 4.1 and 4.2. DAQ1 outperforms all other compared methods both in LCC and SROCC metrics. It achieves an overall LCC score of 0.955 and SROCC score of 0.961 while the highest scores achieved by existing models are 0.953 and

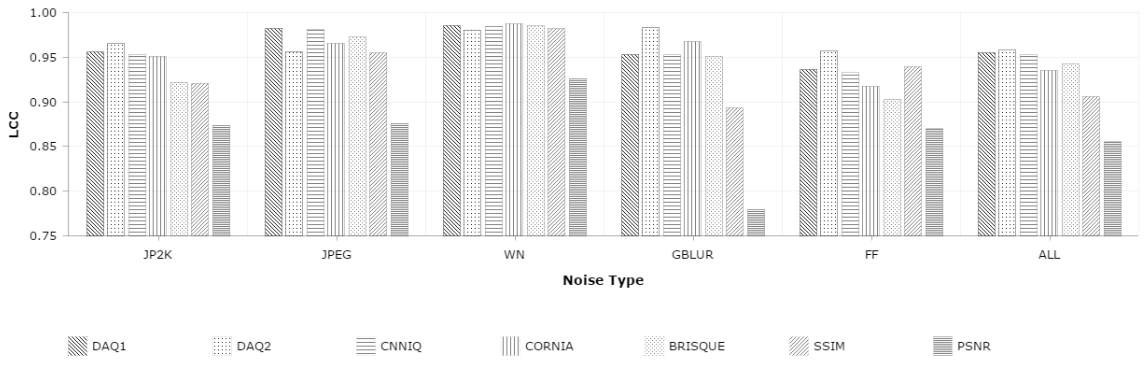


Figure 4.1: LCC of DAQ1 and DAQ2 compared to CNNIQ, CORNIA, BRISQUE, SSIM and PSNR for LIVE R2. DAQ outperforms all other methods. X-axis Labels: JPEG2000 (JP2K), JPEG, White Noise (WN), Gaussian Blur (GBLUR), Fast Fading (FF), Overall performance (ALL).

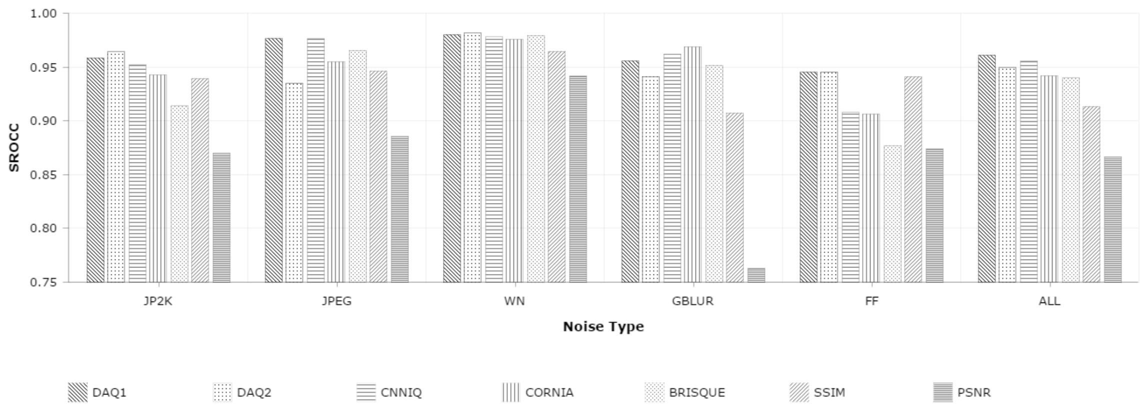


Figure 4.2: SROCC of DAQ1 and DAQ2 compared to CNNIQ, CORNIA, BRISQUE, SSIM and PSNR for LIVE R2. DAQ1 outperforms all other methods. DAQ2 also achieves a competitive score. X-axis Labels: JPEG2000 (JP2K), JPEG, White Noise (WN), Gaussian Blur (GBLUR), Fast Fading (FF), Overall performance (ALL).

Table 4.1: LCC of DAQ1, DAQ2 and other comparison methods on LIVE-R2 dataset

category	DAQ1	DAQ2	CNNIQ	CORNIA	BRISQUE	SSIM	PSNR
JP2K	0.956	0.965	0.953	0.951	0.922	0.921	0.873
JPEG	0.982	0.956	0.981	0.965	0.973	0.955	0.876
WN	0.985	0.980	0.984	0.987	0.985	0.982	0.926
GBLUR	0.953	0.983	0.953	0.968	0.951	0.893	0.779
FF	0.936	0.957	0.933	0.917	0.903	0.939	0.870
ALL	0.955	0.958	0.953	0.935	0.942	0.906	0.856

Table 4.2: SROCC of DAQ1, DAQ2 and other comparison methods on LIVE-R2 dataset

category	DAQ1	DAQ2	CNNIQ	CORNIA	BRISQUE	SSIM	PSNR
JP2K	0.958	0.964	0.952	0.943	0.914	0.939	0.870
JPEG	0.977	0.935	0.977	0.955	0.965	0.946	0.885
WN	0.980	0.988	0.978	0.976	0.979	0.964	0.942
GBLUR	0.956	0.941	0.962	0.969	0.951	0.907	0.763
FF	0.945	0.945	0.908	0.906	0.877	0.941	0.874
ALL	0.961	0.950	0.956	0.942	0.940	0.913	0.866

0.956 respectively. DAQ2 also achieves a great LCC scores of 0.958. It achieves a SROCC score of 0.950, which is not as good as DAQ1 but still enough competitive. The detailed results are given in Tables 4.1 and 4.2. The differences between scores are very small since existing state-of-the-art models already achieved a very high score on both LCC and SROCC. The comparison here is only to show that DAQ models are competitive enough on doing human vision system IQA while being capable of doing computer vision system IQA.

4.1.2 Cross Data Validation on TID-2008

The generalization performance of DAQ1 is validated using the TID2008 image set while the model is trained using the LIVE-R2 dataset. TID-2008 has 17 different types of distortions, whereas, LIVE-R2 has only 5 types of distortions and 4 of them are shared by TID-2008 dataset; Therefore, we choose only the 4 shared distortions from TID2008 dataset. The performance of DAQ models in terms of LCC and SROCC compared to three other, namely, CORNIA, BRISQUE and CNNIQ is shown in Table. 4.3. DAQ1 outperforms the three state-of-the-art methods in cross data set evaluation, which shows strong generalization capability of DAQ1. DAQ2 also performs well on TID-2008 dataset.

Table 4.3: LCC and SROCC on TID2008 [1] dataset

	DAQ1	DAQ2	CORNIA	BRISQUE	CNNIQ
LCC	0.913	0.887	0.880	0.892	0.903
SROCC	0.936	0.890	0.890	0.882	0.920

4.1.3 Weight Learner Evaluation

To evaluate the performance of weight learner, we visualized its output on some LIVE benchmark images as shown in Fig. 4.3. The 32 $\times$ 32 patches are generated by a stride of 8, overlapping each other. The intensity of each pixel are calculated as the sum of overlapped weights predicted by weight learner, brighter pixels indicate the area is given more weight on average. The visualized weights are shown right to the original image. We can tell from Fig. 4.3 that the Weight Learner has learned to give more weight to areas containing certain edges, which draws more attention from human vision.

4.2 Predicting Computer Vision System

To evaluate the effectiveness of the two DAQ models in evaluating the performance of computer vision algorithms, we choose one of Miovision's object detectors called Decoupled Active Contour (DAC). DAC was run on 5000 frames of Miovision data set and a probability map was generated for each image. Those probability maps were compared to the vehicle layout annotated by human and the pixel level error was recorded as the quality score. Finally, as shown in Fig. 3.2, these recorded quality score were used as the ground truth image quality for training DAQ. DAQ was trained on 3000 frames, validated and tested on 1000 frames. The test performance is shown in Fig. 4.4. DAQ1 achieves LCC and SROCC score of 0.80 and 0.82 respectively. DAQ2 achieves a much better score of 0.89 (LCC) and 0.90 (SROCC).

Fig. 4.4 shows the experiment result of DAQ1. It is not a perfect result but still shows some predicting power on the failure detection. The score lower than DAQ1 achieved in former experiments may be caused by the different definition of "quality". The quality of benchmark images is based on the level of several kinds of global distortions while the quality of DAC model signal is based on the performance of DAC model. Therefore the former quality is mainly decided by local pattern while the later quality may be affected by color, shape and contrast in a much wider area, which can not be easily handled by a small network like DAQ1.

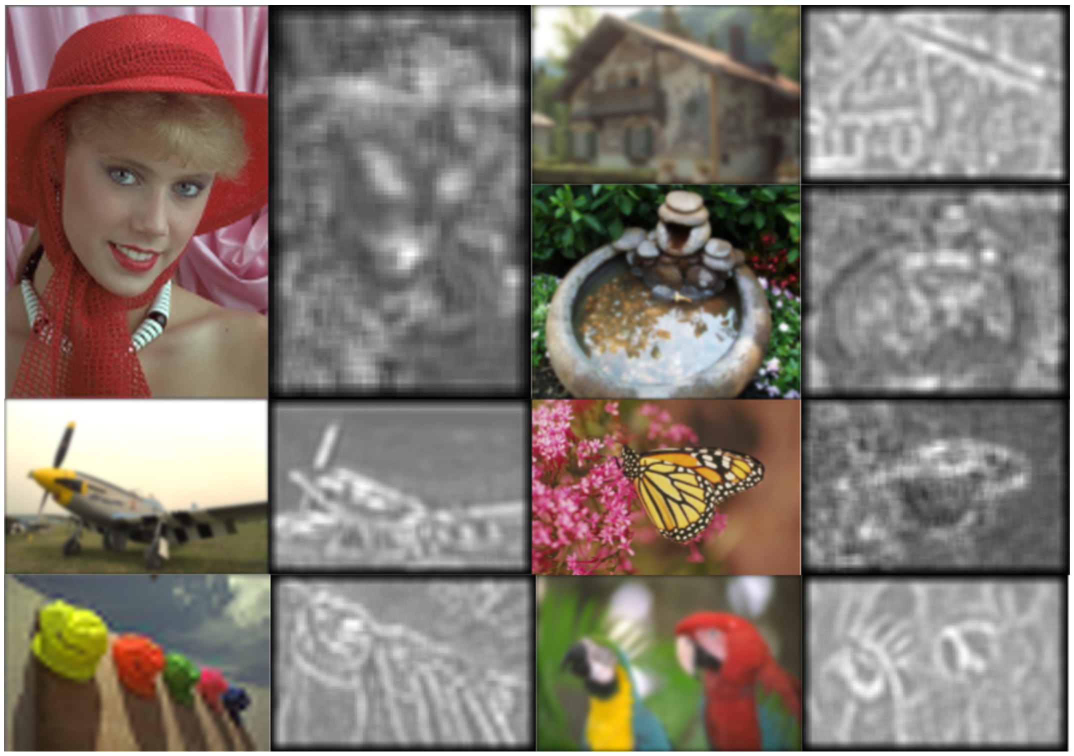


Figure 4.3: Examples of images of various distortion types and the predicted important regions. Predicted weights are shown right to the related image.

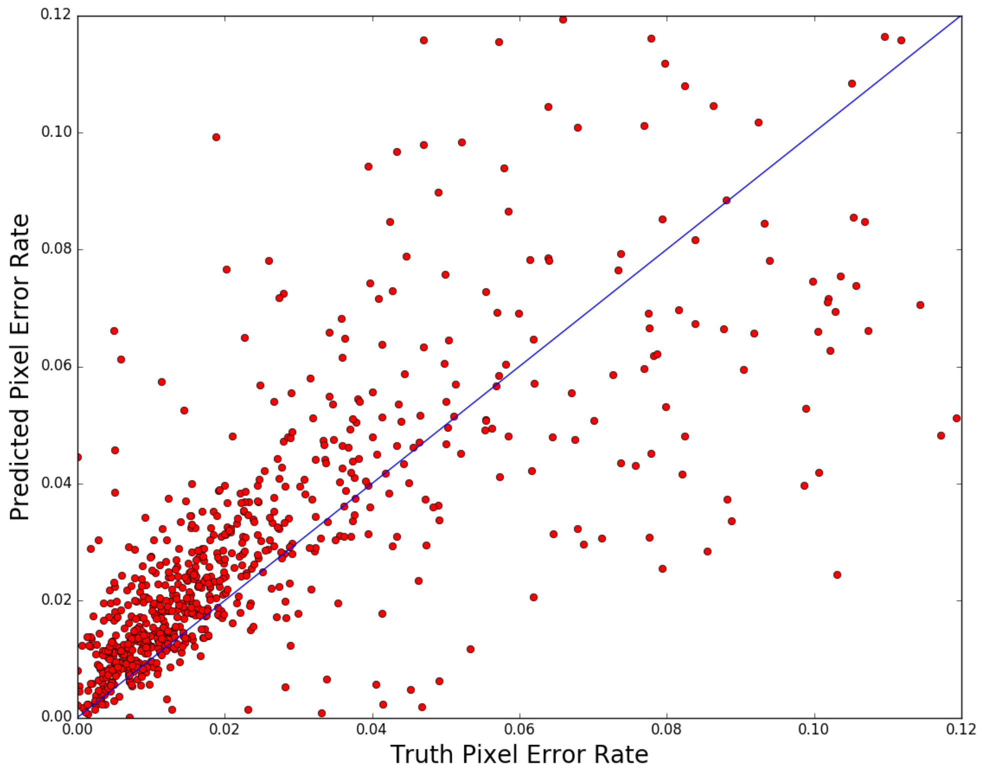


Figure 4.4: True pixel error rate versus predicted error rate of a 1000 sample test data set of DAQ1. The ground truth and predicted pixel error rate have LCC of 0.80 and SROCC of 0.82 (the diagonal line is the theoretical perfect result).

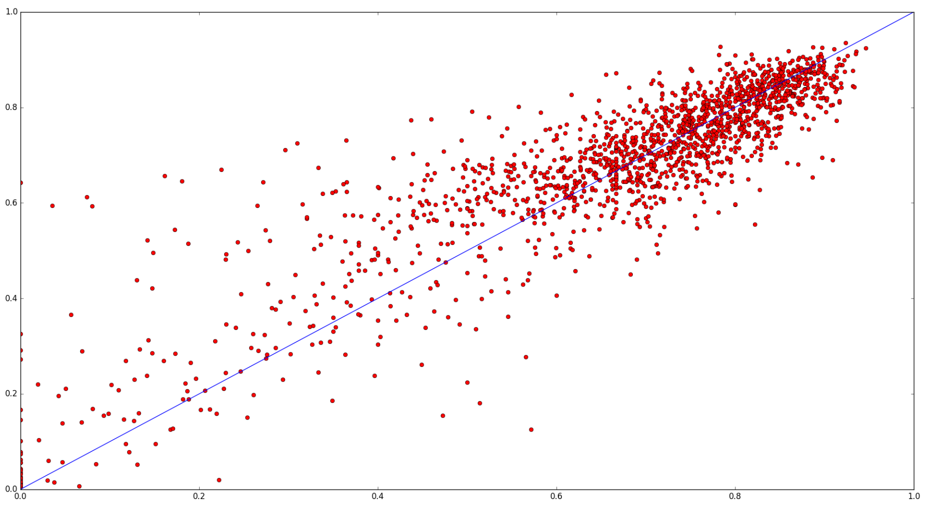


Figure 4.5: True pixel error rate versus predicted error rate of a 1000 sample test data set of DAQ2. The ground truth and predicted pixel error rate have LCC of 0.89 and SROCC of 0.90 (the diagonal line is the theoretical perfect result).

One thing worth mentioning is that the weight learner didn't improve the performance of the pre-trained CNNIQ network during the experiment and was finally trained to produce the same result of the pre-trained CNNIQ network by learning a uniform weight for all patches. This uniform weight for all patches might be caused by the assumption that the DAC detector has a similar way of thinking as human annotators, which affected more by certain patterns but less by others. On the contrary, the DAC detector might pay same attention to all patches thus they have equivalent weights. However, it is also possible that the current weight learner is too small to catch the features of a computer vision system.

Fig. 4.5 shows the experiment result of DAQ2. It achieves much better results than DAQ1. One possible reason is that DAQ2 is a deeper and larger network than DAQ1, and thus it can handle the complex behavior of DAC detector. Also, the better performance may be achieved by the use of glsMCC score rather than average pixel error rate. Perhaps glsMCC score reflects the true quality of image better and brings less confusion to the learning.

Another problem revealed in the experiment is the imbalance of data set. As we can tell from Fig. 3.3, DAC detector works very well on most input images and hence the quality score given to the images are usually quite high. This lead to a lack of bad samples during both training and testing, and thus the models may not act as well as they do on human visual systems.

4.2.1 Feature Learning Evaluation

In the experiments, we combined information from source images with information from computer vision system output. It achieves great effect in increasing performance of the models. In early stage experiments of computer vision system IQA, models are trained using source image only, without combining DAC detector outputs, just like what we do in human IQA experiments. In that way, DAQ1 achieves a LCC of around 0.7 and DAQ2 achieves a LCC of around 0.8. With the data combination, DAQ1 achieves a LCC of around 0.8 and DAQ2 achieves a LCC of around 0.9. The data combination brings a significant increase of about 0.1 in LCC. However, we have to make sure this improvement is based on meaningful features learned instead of end up with some worthless regression function as simple as giving images with more vehicles higher score. Since DAC detector usually works very well and we get a biased dataset with much more high scores than low scores, such problem could happen.

Although understanding what the whole CNN learns has never been easy, we could find clues to our question by simply examining the features learned by the first convolutional

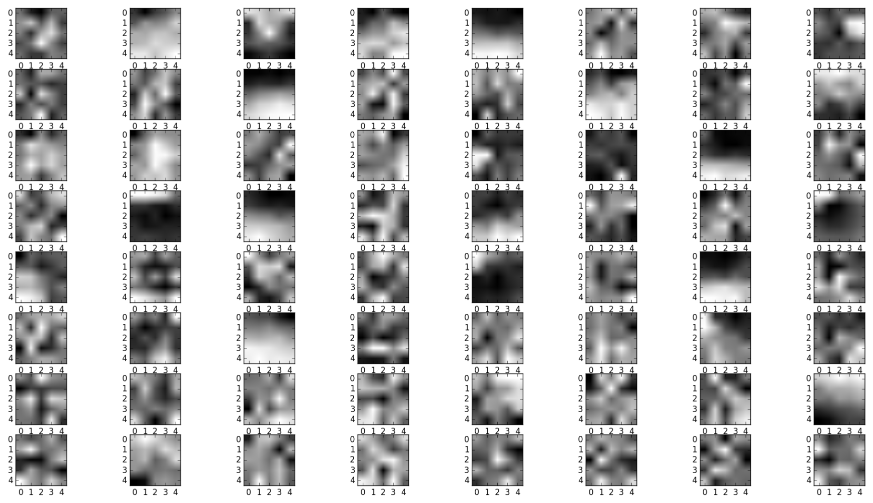


Figure 4.6: 64 features learned from DAC outputs by Conv1 of DAQ2 model.

layer of CNN, which is usually intuitive enough. Following the instructions given by [6] and [37], we visualize the 64 filters learned by the first convolutional layer of DAQ2 model by displaying their weights as images. We choose DAQ2 model instead of DAQ1 model as an example because it takes raw image as input rather than a normalized one, which makes the features closer to human sense. Each filter of Conv1 layer is a $5 \times 5 \times 4$ tensor where the first 3 channels in depth are learned from source images and the 4th channel is learned from DAC detector outputs. Fig.4.6 shows the features learned from DAC detector outputs. Each pattern represents the weights in one of 64 filters Conv1 learned during training, where white is positive value and black is negative value. Fig. 4.7 shows the features learned from source images in the same way. These filters are originally in R, G, B channels and converted to black and white for a better comparison with Fig. 4.6. In this way, every filter is divided into 2 parts: feature learned from DAC outputs and feature learned from source images. In Fig. 4.7 and Fig. 4.6, features at the same position belong to the same filter.

The features learned by DAQ2 are mainly noisy patterns, very similar to those learned by Kang's network [6], which proves result of training to some extent. In addition, by comparing the features learned from source images and from DAC outputs we observe that in multiple cases features from the same filter are complementary to each other. Fig. 4.8

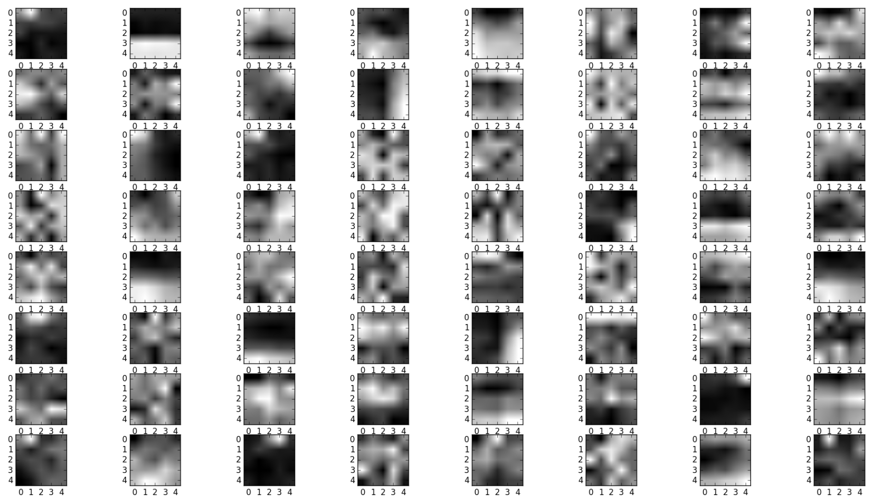


Figure 4.7: 64 features learned from source images by Conv1 of DAQ2 model.

shows some of them. The complementary characteristic appears in both simple features like those at (1, 2) and complex features such those at (2, 0). Each pair of those features makes up a filter that detects the appearance of contrary pattern in source image and DAC output. Intuitively, we may think of these filters as a similar decision making process as we human do when evaluating a detector output by comparing it to the source image and finding the differences. This proves the effectiveness of providing DAC outputs as additional input from another aspect.

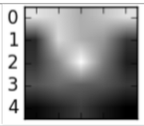
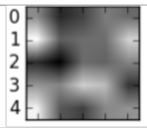
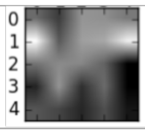
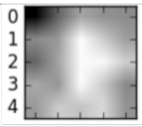
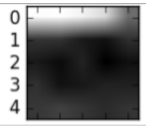
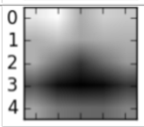
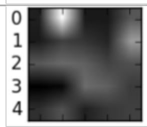
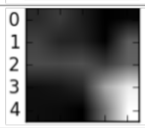
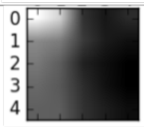
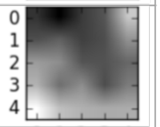
DAC Output Filter					
Source Image Filter					
Position	(2, 0)	(7, 7)	(5, 3)	(1, 2)	(1, 3)

Figure 4.8: Certain filters learned from source image and DAC outputs are complementary to each other. This characteristic even appears in complex features like the one at (2, 0).

Chapter 5

Conclusion and Future Work

This thesis has proposed novel IQA models based on convolutional neural networks. Firstly, the existing works in the IQA field and the development of convolutional neural networks have been reviewed. Then, the inherent challenges of IQA have been discussed, and the differences of IQA for human viewers and computer vision systems have been highlighted. To build a multipurpose IQA model for both human viewers and computer version systems, two models have been developed by implementing the conventional neural networks, which has been shown to outperform the state-of-the-art IQA methods in existing literature.

The first model, DAQ1, has been built upon CNNIQ network, which has been proved to have great performance on IQA. In the DAQ1 model, CNNIQ network has been combined with an additional weight learner network that works in parallel, and the weighted average has been applied to all patches. Furthermore, a novel alternative training method has been implemented in the experiment. In this way, DAQ1 has outperformed original CNNIQ network and other state-of-the-art methods. Specifically, DAQ1 can achieve both LCC and SROCC score around 0.8 in computer vision system IQA experiments, which reveals the potential of CNNs in computer vision system IQA.

DAQ2 model has employed a general CNN architecture with more layers and larger perception field compared to DAQ1. DAQ2 has also achieved great correlation coefficient on human IQA and a more competitive score (around 0.9 in LCC and SROCC) in computer vision system IQA.

For human IQA, both models have been evaluated according to LIVE-R2 and TID-2008 datasets. 100 instances of each model have been trained individually on LIVE-R2 dataset. For the training of each instance, LIVE-R2 dataset has been randomly divided into 60% training data, 20% evaluation data and 20% testing data. Every instance has been tested using the corresponding testing data as well as data with same distortion types in TID-2008 dataset, for a cross validation. The final LCC and SROCC scores have been computed by averaging the result of 100 runs. At last, both models have achieved correlation scores around 0.96, slightly outperforming the original CNNIQ network.

For computer vision system IQA, the Miovision's DAC detector and Miovision dataset have been adopted in the experiments. Both models have shown a prediction power on computer vision system IQA. DAQ2 has outperformed DAQ1 with a correlation score around 0.9 versus around 0.8. The possible reasons have been analyzed in 4.2.

For the future work, the weight learner on computer vision system IQA is worth improving. As mentioned in Sections 4.1.3 and 4.2, the current weight learner produces meaningful outputs for human visual system, but gives all patches the same weight in experiment on computer vision system. As it may be caused by the limited capability of the current weight learner, a new weight learner with more neurons and layers is a potential solution. Besides, the weight learner itself has shown the potential of predicting human focus points in images. It can be further developed into a detector for possible focus points, which is useful in advertising and video making.

Furthermore, the experiment is based on a single computer vision system DAC. To evaluate the generalization ability of the models, further experiments on more computer vision systems are needed. Since it's the first time to train a model for a computer vision system, the benchmark data are not perfectly collected, and the training and testing datasets are not well balanced. We have way more good samples than bad samples. This imbalance problem could commonly happen on other computer vision systems too, because current computer vision systems usually have very high accuracy. Hence we need either to balance the dataset in some way or to rethink our quality quantization method.

References

[1] 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.

[2] Ghiles. Overfitted regression, 3 2016. [Online; accessed April 23, 2017].

[3] Christopher Olah. Conv nets: A modular perspective, Jul 2014.

[4] Aphex34. Convolutional layer, 12 2015. [Online; accessed April 23, 2017].

[5] Aphex34. Max pooling, 12 2015. [Online; accessed August 1, 2017].

[6] L. Kang, P. Ye, Y. Li, and D. Doermann. Convolutional neural networks for no-reference image quality assessment. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1733–1740, June 2014.

[7] Marius Pedersen and Jon Yngve Hardeberg. Full-reference image quality metrics: Classification and evaluation. *Found. Trends. Comput. Graph. Vis.*, 7(1):1–80, January 2012.

[8] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, 13(4):600–612, April 2004.

[9] Zhou Wang and Eero P Simoncelli. Reduced-reference image quality assessment using a wavelet-domain natural image statistic model. In *Proc. of SPIE Conf. Human Vision and Electronic Imaging, San Jose, CA*, pages 149–159. Society of Photo-Optical Instrumentation Engineers, 2005.

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

[11] Hamid R Sheikh and Alan C Bovik. A visual information fidelity approach to video quality assessment. In *The First International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, pages 23–25, 2005.

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

[13] Kim-Han Thung and Paramesran Raveendran. A survey of image quality measures. In *Technical Postgraduates (TECHPOS), 2009 International Conference for*, pages 1–4. IEEE, 2009.

[14] Zhou Wang and Alan C Bovik. Modern image quality assessment (synthesis lectures on image, video, and multimedia processing). *San Rafael, CA: Morgan Claypool*, 2006.

[15] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[16] Zhou Wang, Alan C Bovik, and Ligang Lu. Why is image quality assessment so difficult? In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 4, pages IV–3313. IEEE, 2002.

[17] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986.

[19] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, 2012.

[23] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. *Artificial Neural Networks–ICANN 2010*, pages 92–101, 2010.

[24] Jie Li, Lian Zou, Jia Yan, Dexiang Deng, Tao Qu, and Guihui Xie. No-reference image quality assessment using prewitt magnitude based on convolutional neural networks. *Signal, Image and Video Processing*, 10(4):609–616, 2016.

[25] Yuming Li, Lai-Man Po, Litong Feng, and Fang Yuan. No-reference image quality assessment with deep convolutional neural networks. In *Digital Signal Processing (DSP), 2016 IEEE International Conference on*, pages 685–689. IEEE, 2016.

[26] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik. LIVE Image Quality Assessment Database Release 2, April 2014.

[27] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes in C*. Cambridge university press Cambridge, 1 edition, 1992.

[28] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[29] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[30] 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.

[31] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[32] Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Huaming Wang, et al. An introduction to computational networks and the computational network toolkit. *Microsoft Technical Report MSR-TR-2014–112*, 2014.

[33] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[34] 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.

[35] 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.

[36] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a completely blind image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2013.

[37] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

Appendix A

DAQ1 Source Code

A.1 Score Learner(CNNIQ)

```
# conv1
kernal1 = tf.Variable(initial_value=tf.truncated_normal(shape=[7, 7, 2,
 ↪  50], stddev=0.004), trainable=True,
collections=['cvpr'], name='k1')
conv1 = tf.nn.conv2d(input=patch, filter=kernal1, strides=[1, 1, 1, 1],
 ↪  padding='VALID')
bias1 = tf.Variable(tf.constant(0.0, shape=[50]), trainable=True,
 ↪  collections=['cvpr'], name='b1')
active1 = tf.nn.bias_add(conv1, bias1)

max_pool = tf.reduce_max(active1, reduction_indices=[1, 2])
min_pool = tf.reduce_min(active1, reduction_indices=[1, 2])

# fc1

weights_fc1_1 = tf.Variable(initial_value=tf.truncated_normal([50, 800],
 ↪  stddev=0.04), trainable=True,
collections=['cvpr'], name='f1w1')
bias_fc1_1 = tf.Variable(tf.constant(0.0, shape=[800]), dtype=tf.float32,
 ↪  trainable=True, collections=['cvpr'],
name='fb1')
```

```
weights_fc1_2 = tf.Variable(initial_value=tf.truncated_normal([50, 800],
 ↪  stddev=0.04), trainable=True,
collections=['cvpr'], name='f1w2')
bias_fc1_2 = tf.Variable(tf.constant(0.0, shape=[800]), dtype=tf.float32,
 ↪  trainable=True, collections=['cvpr'],
name='fb2')
active_fc1 = tf.nn.relu(tf.add(tf.add(tf.matmul(max_pool, weights_fc1_1),
 ↪  bias_fc1_1),
tf.add(tf.matmul(min_pool, weights_fc1_2), bias_fc1_2)))

# fc2
weight_fc2 = tf.Variable(initial_value=tf.truncated_normal([800, 800],
 ↪  stddev=0.08), trainable=True,
collections=['cvpr'], name='f2w1')
bias_fc2 = tf.Variable(initial_value=tf.constant(0.0, shape=[800]),
 ↪  dtype=tf.float32, trainable=True,
collections=['cvpr'], name='f2w2')
active_fc2 = tf.nn.relu(tf.add(tf.matmul(active_fc1, weight_fc2),
 ↪  bias_fc2))
active_fc2_drop = tf.nn.dropout(active_fc2, 0.5)

# output
weight_output = tf.Variable(initial_value=tf.truncated_normal([800, 1],
 ↪  stddev=1.0 / 400.0), trainable=True,
collections=['cvpr'], name='opw1')
bias_output = tf.Variable(initial_value=tf.constant(0.0, shape=[1]),
 ↪  dtype=tf.float32, trainable=True,
collections=['cvpr'], name='opb1')
output = tf.add(tf.matmul(active_fc2_drop, weight_output), bias_output,
 ↪  name='output')
```

A.2 Weight Learner

```
patch_weights_var = {
'patch_kernal': tf.Variable(initial_value=tf.truncated_normal(shape=[32,
 ↪  32, 1, 64], stddev=0.001),
collections=['patch_weights'], name='patch_kernal'),
'patch_bias': tf.Variable(initial_value=tf.constant(0.0, shape=[64]),
 ↪  collections=['patch_weights'],
name='patch_bias'),
'patch_fc_weights': tf.Variable(initial_value=tf.truncated_normal([64, 1],
 ↪  stddev=1 / 64),
collections=['patch_weights'], name='patch_fc_weights'),
'patch_fc_bias': tf.Variable(initial_value=tf.constant(0.0, shape=[]),
collections=['patch_weights'],
name='patch_fc_bias')
}

def apply_patch_weights(self, image_patches, patch_weights_var):
  patch_conv = tf.nn.conv2d(image_patches,
   ↪  patch_weights_var['patch_kernal'], [1, 1, 1, 1],
  padding='VALID')
  patch_active = tf.reshape(tf.nn.relu(tf.nn.bias_add(patch_conv,
   ↪  patch_weights_var['patch_bias'])),
  shape=[-1, 64])
  patch_weights = tf.reshape(tf.nn.relu(
  tf.add(tf.matmul(patch_active, patch_weights_var['patch_fc_weights']),
  patch_weights_var['patch_fc_bias'])),
  shape=[-1])
return patch_weights
```

Appendix B

DAQ2 Source Code

```
# conv1
kernal1 = tf.Variable(initial_value=tf.truncated_normal(shape=[5, 5, 4,
 ↪  64], stddev=1e-4))
conv1 = tf.nn.conv2d(image, kernal1, [1, 1, 1, 1], padding='SAME')
bias1 = tf.Variable(initial_value=tf.constant(0.0, shape=[64]))
biased1 = tf.nn.bias_add(conv1, bias1)
active1 = tf.nn.relu(biased1)

pool1 = tf.nn.max_pool(active1, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
 ↪  padding='SAME')
norm1 = tf.nn.lrn(pool1, depth_radius=4, bias=1.0, alpha=0.001 / 9.0,
 ↪  beta=0.75)

# conv2
kernal2 = tf.Variable(initial_value=tf.truncated_normal(shape=[5, 5, 64,
 ↪  64], stddev=1e-4))
conv2 = tf.nn.conv2d(norm1, kernal2, [1, 1, 1, 1], padding='SAME')
bias2 = tf.Variable(initial_value=tf.constant(0.1, shape=[64]))
biased2 = tf.nn.bias_add(conv2, bias2)
active2 = tf.nn.relu(biased2)
norm2 = tf.nn.lrn(active2, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75)
pool2 = tf.nn.max_pool(norm2, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
 ↪  padding='SAME')
```

```
# cov3
kernal3 = tf.Variable(initial_value=tf.truncated_normal(shape=[5, 5, 64,
 ↪  64], stddev=1e-4))
conv3 = tf.nn.conv2d(pool2, kernal3, [1, 1, 1, 1], padding='SAME')
bias3 = tf.Variable(initial_value=tf.constant(0.1, shape=[64]))
biased3 = tf.nn.bias_add(conv3, bias3)
active3 = tf.nn.relu(biased3)
pool3 = tf.nn.max_pool(active3, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
 ↪  padding='SAME')
norm3 = tf.nn.lrn(pool3, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75)

# fc1
reshaped_feature = tf.reshape(norm3, [self._batch_size, -1])
dim = reshaped_feature.get_shape()[1].value
weights1 = tf.Variable(initial_value=tf.truncated_normal(shape=[dim, 864],
 ↪  stddev=0.04))
biases_fc1 = tf.Variable(initial_value=tf.constant(0.1, shape=[864]))
active3 = tf.nn.relu(tf.add(tf.matmul(reshaped_feature, weights1),
 ↪  biases_fc1))
weight_decay1 = tf.mul(tf.nn.l2_loss(weights1), 0.0008)
tf.add_to_collection('losses', weight_decay1)

# fc2
weights3 = tf.Variable(initial_value=tf.truncated_normal(shape=[864, 216],
 ↪  stddev=0.04))
biases_fc3 = tf.Variable(initial_value=tf.constant(0.1, shape=[216]))
active5 = tf.nn.relu(tf.add(tf.matmul(active3, weights3), biases_fc3))
weight_decay3 = tf.mul(tf.nn.l2_loss(weights3), 0.004)
tf.add_to_collection('losses', weight_decay3)

# output
weights_output = tf.Variable(initial_value=tf.truncated_normal(shape=[216,
 ↪  self._classes], stddev=1 / 216.0))
biases_output = tf.Variable(initial_value=tf.constant(0.0,
 ↪  shape=[self._classes]))
output = tf.inv(tf.add(tf.exp(tf.neg(tf.add(tf.matmul(active5,
 ↪  weights_output), biases_output))), 1),
name='output')
```