CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of electrical Engineering
Department of Radioelectronics

Machine Learning Techniques for Efficient Image Quality Assessment
Metody strojového učení pro efektivní hodnocení kvality obrazu

Master's thesis

Bc. Jiří Šebek

Master programme: Electronics and Communications
Branch of study: Media and Signal Processing
Supervisor: Ing. Karel Fliegel, Ph.D.

Prague, March 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Šebek** Jméno: **Jiří** Osobní číslo: **434637**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

Studijní obor: **Audiovizuální technika a zpracování signálů**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Metody strojového učení pro efektivní hodnocení kvality obrazu

Název diplomové práce anglicky:

Machine Learning Techniques for Efficient Image Quality Assessment

Pokyny pro vypracování:

Podejte přehled současných metod pro hodnocení kvality obrazu případně videa s využitím metod strojového učení ML (machine learning). Soustřeďte se zejména na nejnovější poznatky v oblasti aplikace hlubokého učení DL (deep learning). Na vybraném problému a testovacích datech analyzujte účinnost zkoumaných metod v porovnání s klasickými přístupy.

Seznam doporučené literatury:

[1] Bovik, A.: Handbook of Image and Video Processing, Elsevier Academic Press, 2005.
[2] Gastaldo, P., Zunino, R., Redi, J.: Supporting visual quality assessment with machine learning, Eurasip Journal on Image and Video Processing, 2013.
[3] Gu, J., Meng, G., Redi, J.A., Xiang, S., Pan, C.: Blind Image Quality Assessment via Vector Regression and Object Oriented Pooling, IEEE Transactions on Multimedia, 20(5), 2018.
[4] Bianco, S., Celona, L., Napoletano, P., Schettini, R.: On the use of deep learning for blind image quality assessment, Signal, Image and Video Processing, 12(2), 2018.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Karel Fliegel, Ph.D., katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **15.02.2019** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **20.09.2020**

Ing. Karel Fliegel, Ph.D.	doc. Ing. Josef Dobeš, CSc.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Podpis studenta

Datum převzetí zadání

Declaration

I hereby declare that I have written this master thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, May 2020

.....

Bc. Jiří Šebek

Abstract

Multimedia technology has experienced rapid growth mainly due to transformation of the mobile market towards smartphones with highly capable cameras and increasing popularity of streaming services. With the mass production and sharing of images and video, to maintain good user experience, a robust and universal quality assessment metric is required. Traditional full reference metrics has limited use and no reference metrics are very challenging. This has began to change due to extensive research of deep learning and neural networks, enabled by higher computation power and it's availability. This thesis discusses the basic principles of neural networks, especially convolutional, approaches to image quality assessment based on them, how they perform against state-of-the-art full reference metrics, what are the main challenges and how to tackle them. The practical part of this thesis contains author's approach to neural network and how it performs on various image databases. Thesis includes a comparison with traditional full reference metrics.

Keywords: *deep learning, neural networks, no-reference image quality assessment*

Abstrakt

Multimediální technologie zažily rapidní rozvoj především díky transformaci trhu s mobilními telefony směrem k telefonům chytrým s velmi schopnými kamerami a růstu popularity streamovacích služeb. Spolu s masovou produkcí a sdílením fotografií a videí, pro zachování dobré uživatelské zkušenosti, roste potřeba robustní a univerzální metriky pro hodnocení kvality obrazu. Tradiční metriky s plnou referencí mají omezené možnosti použití a metriky bez reference představují nemalou výzvu. Tato situace se začíná měnit díky rozsáhlému výzkumu v oblasti hlubokého učení a neuronových sítí, umožněnému rozvojem a stále lepší dostupností výpočetní techniky. Tato práce pojednává o základních principech fungování neuronových sítí se zaměřením na konvoluční sítě, metody hodnocení kvality obrazu na nich založených, jak schopné jsou ve srovnání s nejmodernějšími plnoreferenčními metrikami, jejich hlavní nedostatky a problémy a jak je možné je řešit. Praktická část práce obsahuje autorem vytvořenou neuronovou síť a její fungování s různými databázemi fotek. Práce také obsahuje srovnání této metody s tradičními.

Klíčová slova: *hluboké učení, neuronové sítě, hodnocení kvality obrazu bez reference*

Acknowledgements

The writing of this thesis was many times very challenging and could not be done without certain people I would like to express my thanks to. To Ing. Karel Fliegel, Ph.D. for finding time in his busy schedule, always being helpful and patient. To my parents Eva and Pavel for providing me everything I needed and more, especially during the crisis. To my friends who never turned their back on me. And to the internet community, whose guidance and tutorials which helped me understand the topics more thoroughly.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

In recent years, the photo and video industry has experienced huge transformation. Our smart phones has higher computation power than computers that helped man get to the moon, the most used camera in the world is nowadays Apple iPhone, television broadcast runs in 4K high dynamic range resolution and cars can navigate themselves in between highway lanes. More pictures and videos are produced and streamed than ever before and along with them the need for automatic tools of quality assessment grows. Especially for streaming services, for the best end-user-experience, choosing and setting the codecs to be small enough for average internet user to flawlessly play and quality enough to prevent distortion from breaking the immersion is tough.

In field of image quality assessment there has been a shift towards algorithms that can learn what makes the image look good. In the 2000s the first algorithms using learned features were developed and are used as a standard metric to this day. Few years later the complexity of proposed algorithms grew, following the increase in computation power and taking advantage of statistical modelling of natural scenes. These algorithms attempted to tackle quality assessment without knowing how the image should look when pristine, with results drawing close to the full-reference metrics, but rarely matching them. We analyze the most significant ones in Chapter 2.

With increasing computation power and affordability, the family of machine learning algorithms became massively popular. It was mainly due to three aspects: the graphics processing units' power grew and consumer-grade cards became powerful enough to drive some scientific processes, large-scale publicly available datasets came to be, with millions of labeled images and major breakthrough models were introduced, like AlexNet [1] and VGG-16 [37]. Big tech companies like Google and lately Amazon saw the potential of these algorithms and actively participated in the development, pushing it even further. More about how neural networks work and how they can be developed can be found in Chapter 3.

The most used task for deep learning is object recognition. Using the vast amounts of data, the deep networks learns the distinguishing features based on the labeled images to later find similar patterns when put to test. The task of image quality assessment is fundamentally similar, since every distortion has its unique specifics. The problem arises in the amount of data required to properly train robust network. While there are millions of images labeled in terms of content, the datasets with images labeled with quality are many times smaller. Another challenges are presented in the making the network robust to perform across all kinds of images, multiple distortions present at once and more. Some of the most recent and promising approaches are presented in Chapter 4.

The practical part of this thesis include authors attempt to create functional convolutional neural network, inspired by approaches from Chapter 4 and theoretical information from Chapter 3. Testing and validation of the network are done on one of the newer databases and evaluations are done on some of the well-known and used databases. In chapter 7, the author draws conclusions and outlines possible future works to expand on topic.

Chapter 2

Contemporary BIQA Methods

Image quality assessment (IQA) can be divided into three subgroups according to the type of input data. First we have full reference IQA, where we work, as name suggests, with both original pristine image and its corrupted version. Here the task is fairly simple, using various metrics we can define differences between the two and quantify how much is the copy true to the original, either from technical point of view like signal-to-noise ratio or by incorporating algorithms modelling human visual system (HVS) to simulate which changes are more apparent to our vision and therefore corrupting the image more severe.

While full reference IQA methods usually yield the best results, the problem arises from the lack of reference, either complete or partial. Reduced-reference IQA aims to get the most possible information from parts of the original image or only some information about it.

The main challenge lies in assessing image quality with zero information about how the image should look. Quality and reliable no-reference IQA would find its place primarily in streaming services and websites to improve their quality of experience. In this chapter we are going to cover some standardized full reference methods, many of them being used commercially, and state-of-the-art approaches to no-reference methods, which are profusely used as benchmarks for most of the newly presented IQA approaches.

2.1 Modern Full-Reference Methods

2.1.1 Structural Similarity Index (SSIM)

One of the standardized and highly used methods is Structural Similarity Index. Using the HVS model, it examines the image as a whole, comparing it with the original, non-

distorted image and provides quality score. This method was developed in Laboratory for Image and Video Engineering (LIVE) at The University of Texas at Austin and is being further developed with many variations today.[4] The HVS is highly adapted to extract structural information from the visual scene, and therefore the measurement of structural similarity or distortion should give us a good approximation of perceived quality. The Structural Similarity Index is used to measure similarity between two images, requiring an original, distortion-free reference image.[4]

The formula for SSIM is as follows:

$$SSIM(\boldsymbol{x}, \boldsymbol{y}) = [l(\boldsymbol{x}, \boldsymbol{y})^{\alpha} \cdot c(\boldsymbol{x}, \boldsymbol{y})^{\beta} \cdot s(\boldsymbol{x}, \boldsymbol{y})^{\gamma}], \tag{2.1}$$

where c stands for contrast, l for luminance and s for structure, $SSIM$ being a weighted combination of those. Setting the weights , , to 1, the formula can be reduced to

$$SSIM(\boldsymbol{x}, \boldsymbol{y}) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{2.2}$$

where $\mu_{x,y}$ is the average of x, y, $\sigma_{x,y}$ variance of x, y and σ_{xy} covariance of x and y, $c_1 - (k_1, L)^2$, $c_2 - (k_2, L)^2$ two variables to stabilize the division with weak denominator, L is the dynamic range of pixel values $(2^{numofbitsperpixel} - 1)$ and $k1 = 0.01$, $k2 = 0.03$ by default. The index's value can be $[0, 1]$, the higher the better while 1 can be reached only when comparing two identical images and 0 indicates no similarity whatsoever. For an image, it is typically calculated using a sliding Gaussian window of size 11x11 or a block window of size 8x8. [4]

You can see how SSIM correlates with your perception of good or poor quality image in Figure 2.1, in contrast with MSE.

2.1.2 Multi-Scale SSIM

Improvement to nowadays' standard quality metric SSIM was brought up by incorporating multiple scales of the image and combining the output evaluations of each one. After evaluation of the reference image and its distorted version, both images are low-pass filtered and downsampled by factor of 2, then evaluated again. This process is repeated $M - 1$ times until reaching M-th scale. Resultant index is obtained by combining all indexes throughout all iterations.

Wang et. al. [43] proposed this approach in 2003 with the only concert being calibrating the parameters, which turned out to be far from predictable. Still, the MS-SSIM

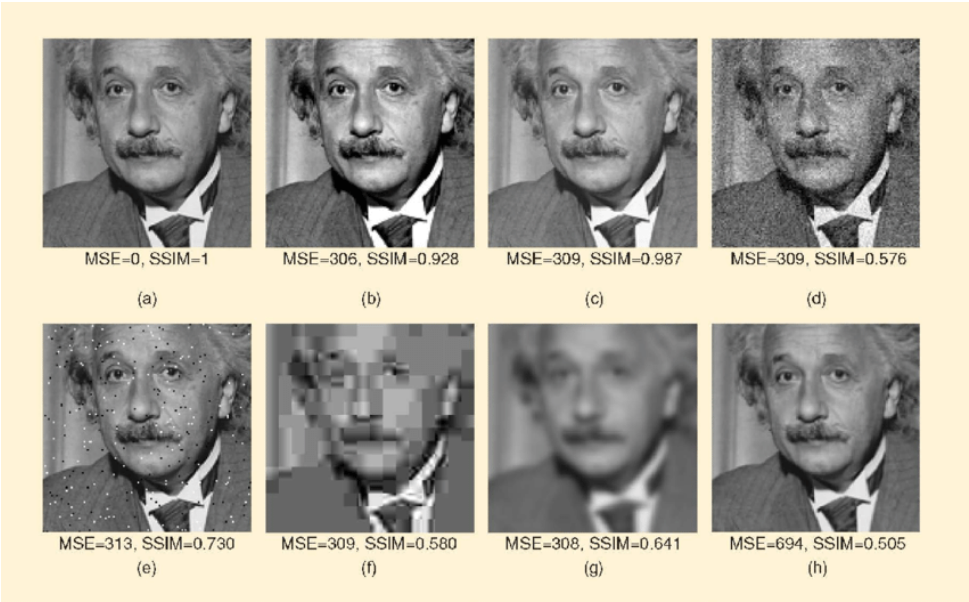

Figure 2.1: Different results of MSE and SSIM. (a) Reference image. (b) Mean contrast stretch. (c) Luminance shift. (d) Gaussian noise. (e) Impulsive noise. (f) JPEG compression. (g) Blurring. (h) Spatial scaling (taken from [35]).

proved itself to be very competitive and many papers nowadays incorporates it in comparisons with newer approaches.

2.1.3 Visual Information Fidelity (VIF)

In contrast with SSIM, this full reference IQA index uses natural scene statistics (NSS) besides HVS to determine the visual quality. Images and videos of all possible signals for small subspace that can be classified as 'natural'. Researchers have developed sophisticated methods of determining the characteristics of these, which come very handy since all kinds of distortion more or less deform the image in a way that the image produces less 'natural' statistics, even if we can clearly tell the contents of the image, as our brain 'fixes' the distortion using experience and interpolation.

The algorithm employs 3 models: NSS modelling, distortion modelling and HVS. Overview of those working in conjunction can be seen in Figure 2.2. The source model, employing NSS, uses Gaussian scale mixture (GSM) to statistically model wavelet coefficients of a steerable pyramid decomposition of an image. The model is described below for a given sub-band of the multi-scale multi-orientation decomposition, with $\mathcal{C}$ representing a given sub-band. After the channel splits into two, distortion modelling is employed in one of them, combining signal attenuation and additive noise in wavelet domain, represented by $\mathcal{D}$. $\mathcal{E}$ and $\mathcal{F}$ represent signals after final application of the HVS modelling. Since several aspects have already been accounted for in the initial processing, the additional

modeling is based on hypothesis that the uncertainty in the perception of visual signals limits the amount of information that can be extracted from the source and distorted image. [32]

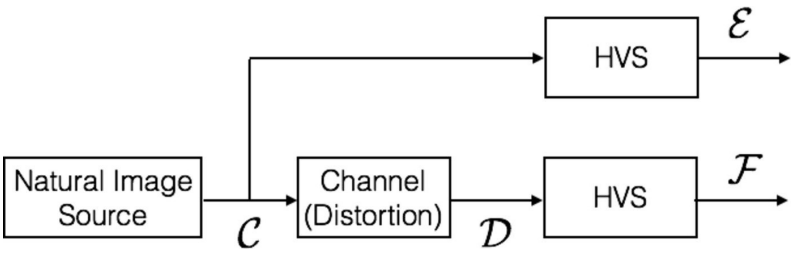

Figure 2.2: VIF algorithm block scheme (taken from [32]).

The Spearman's rank order correlation coefficient (SROCC) between the VIF index scores of distorted images on the LIVE IQA Database and the corresponding human opinion scores is evaluated to be. This suggests that the index correlates very well with human perception of image quality.Moreover, VIF is one of the most significant metrics used today, among others it is deployed in the core of Netflix's video quality monitoring system, analyzing all of the Netflix encoded streamed video. [16] [33]

2.2 No-Reference IQA Methods

All of the standardized methods share one limitation - they require a reference pristine image to provide quality drop index. This makes them useless when e. g. measuring the quality of a live stream so we can immediately improve the quality. In addition to that, Mittal et. al [20] states that full-reference and reduced-reference IQA can only accurately measure fidelity, which does not necessary mean quality, since the original image can be objectively bad or distorted. Researchers are therefore focusing on field of study called Blind- or No-Reference Image Quality Assessment (BIQA/NR-IQA) to tackle these problems. While there is currently no standardized method providing truly reliable results, many of them have been proposed in past decade. Following sections will introduce some of the most commonly used in comparisons and benchmarks.

2.2.1 DIIVINE

One of the first algorithms aiming to better the NR-IQA field was developed in 2011, largely base on natural scene statistic modeling [21]. The basic premise is that natural scenes possesses certain statistic qualities and parameters that are altered with distortion, resulting in an unnatural image and by characterizing these unnatural parameters the

distortion can be classified and it magnitude can be determined. In addition, unlike many previous algorithms, this would work for any distortion and/or their combination. The result is 2-stage algorithm utilizing the principles of how human brain processes and filters images and statistical features characteristic to natural images.

First stage is image scale-space-oriented decomposition, similar to wavelet transform, to form oriented band-pass responses. These coefficients are then used in natural scene statistics model to form feature vectors, which statistically represent distortion in the image. Utilizing these vectors Moorthy et. al. identify the probability that the image is affected by one or more of the distortion categories and build regression model for each distortion category to turn the features into quality score in each one of them. The set of wavelet coefficients is modeled using GSM and the features are extracted using steerable pyramid decomposition [36] and divisive normalisation. These result in following sub-band statistics:

- **Scale and Orientation Selective Statistics**, parametrized by generalised Gaussian distribution (GGD). Divisive normalisation tends to produce coefficients in Gaussian manner for natural images, so in case of distortion the fit using GGD shows clear diferences
- **Orientation Selective Statistics** analyzing the correlations between same-oriented images across scales. Since images are naturally multiscale, some features are shared between various-scale sub-bands. These are compromised when distortion is present, which can be analyzed by GGD
- **Correlations Across Scales** computed using high-pass and band-pass sub-bands of the image, which are filtered using Gaussian window to detect relationship between them
- **Spatial Correlation** using natural images' correlation structure that in most cases varies as function of distance
- **Across Orientation Statistics** including correlations that images exhibit across orientation and their changes with distortion present

Second stage requires training on evaluated data, with no need for reference image. The classifier 'learns' the mapping from feature space into class label and after the calibration is finished, it produces n-dimensional vector that represents probabilities of the input having certain types of distortion. The whole process can be seen in

DIIVINE was evaluated on LIVE IQA database, resulting in statistical superiority to other IQA methods and draw with SSIM, surpassing him in JPEG2000 evaluation but losing in noise and blur distortions. [21]

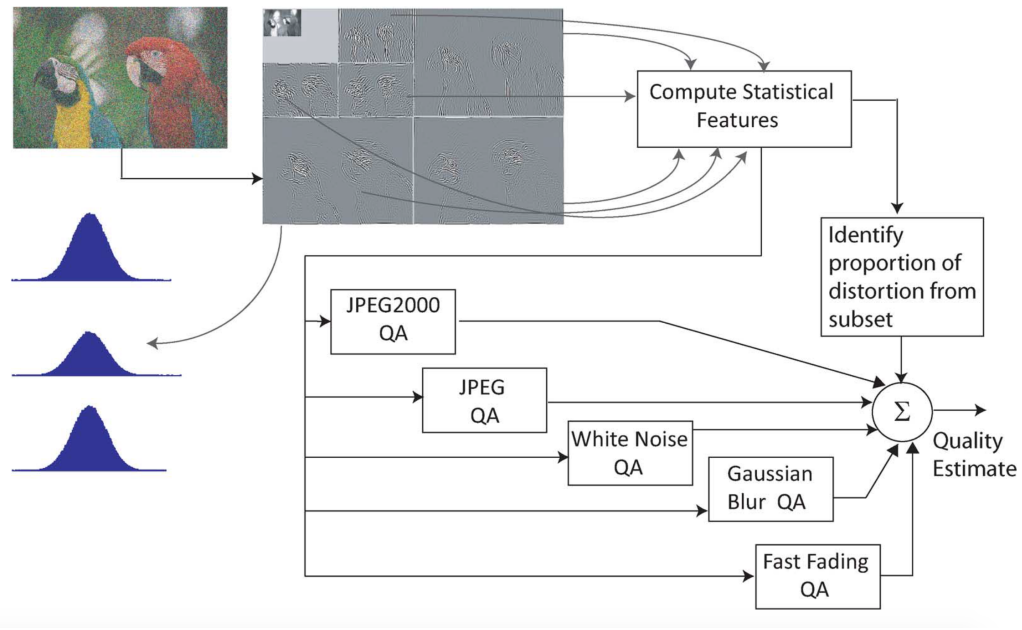

Figure 2.3: Block scheme of DIIVINE 2-framework algorithm (taken from [21]).

2.2.2 BLIINDS-II

This approach also relies on natural scene statistics models and transformation, but unlike DIIVINE's wavelet transformation authors choose discrete cosine transform (DCT) coefficients to simplify and speed up the whole process, simultaneously reducing time required for training the algorithm. BLIINDS-II is a single-stage framework that focuses entirely on natural scene statistics, since DCT computation can be done much faster than modelling functions of the HVS, at those are still far from understood.

The algorithm is based on the changing of NSS parameters with increasing distortion. It is trained using features derived directly from a generalized parametric statistical model of natural image DCT coefficients against various perceptual levels of image distortion. Learned model is then used to predict quality score.

Framework overview can be seen in Figure 2.4. The image destined for evaluation is partitioned into nxn blocks or patches, for which the 2D DCT is computed. The coefficient extraction is performed locally in the spatial domain. For each block is then applied generalised Gaussian density model, as well as for each coefficient. For each block and coefficient is then also obtained generalized Gaussian fit. Third stage of the framework computes functions of the derived generalized Gaussian model parameters. Finally, a Bayesian prediction model is used to maximize the probability that the image has a certain quality score given the model-based features extracted from the image, and to produce quality score.

In conclusion, BLIINDS-II have similar results as DIIVINE, both algorithms out-

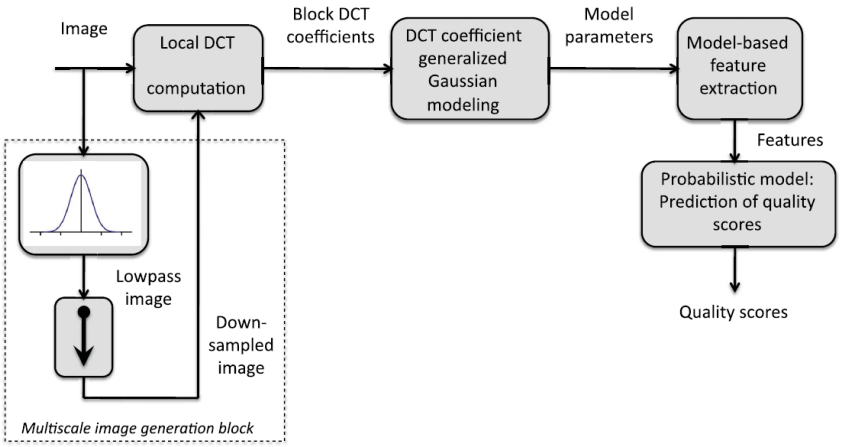

Figure 2.4: Overview of the BLIINDS-II framework (taken from [30]).

perform or match current full-reference metrics with some limitations. DIIVINE needs distortion categories to be specified to achieve state-of-the-art results and therefore is pseudo-distortion-unspecific, but due to that it can predict what type(s) of distortion affect the image as a by-product. BLIINDS-II having only on one QA engine is less computation demanding, cutting the processing time by little more than half [20] due primarily to operating in DCT domain.

2.2.3 BRISQUE

This no-reference IQA, although based on NSS as well, operates in spatial domain. Unlike previously mentioned DIIVINE and BLIINDS-II, BRISQUE is also more straightforward, focusing solely on the 'naturalness' of the image, thus less computation-demanding, meaning it could be more usable for real-time applications.

The key principle of the spatial domain approach is computing locally normalized luminances via local mean subtraction and divisive normalization. These strongly tend to be unit normal Gaussian characteristic for natural images. With this we get mean subtracted contrast normalized coefficients (MSCN). These are, by authors' hypothesis, strongly affected by distortion. If plotted, coefficients produced by pristine image yet again have Gaussian-like distribution, while for example blur creates more Laplacian-like behaviour. MSCN coefficients plotted can be seen in Figure 2.5.

Later, GGD is applied to capture broad spectrum of distorted image statistics, along with statistical relationships between neighboring pixels. While MSCN coefficients are definitely more homogeneous for pristine images, the signs of adjacent coefficients also exhibit a regular structure, which gets disturbed in the presence of distortion. Finally,

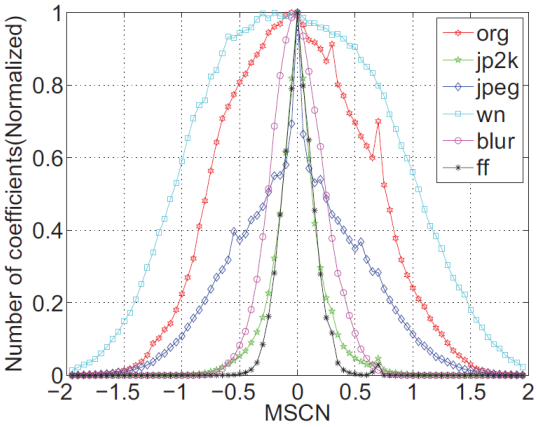

Figure 2.5: Histogram of MSCN coefficients for a natural undistorted image and its various distorted versions (taken from [20]).

similarily to previously mentioned algorithms, natural multi-scale-likeness of images is taken advantage of by extracting features from original image as well as from LP filtered down-sampled one by factor of 2.

Evaluation was onca again performed using LIVE IQA database with 5 kinds of distortion. BRISQUE as well as DIIVINE and BLIINDS-II requires learning to perform, so the database was split 4:1 into training and testing subset with no overlapping images, iterating 1000 times and evaluating the median performance. BRISQUE outperforms all no-reference methods published until that day and except for MS-SSIM [43] even the full-reference methods like SSIM with similarity to Mean opinion score of the image. Concerning computation speed, simple metrics like PSNR still outperforms BRISQUE with more than ten times the speed, but BRISQUE runs over 50 times faster than BLIINDS-II and over 100 times faster than DIIVINE. BRISQUE regularly appears in comparisons of new approaches to IQA as one of the most competitive methods. [20]

Chapter 3

Deep Neural Networks Theory

Neural networks are series of algorithms, modelled to operate similarly to human brain. They take input data, either raw or processed, and performs various numerical, in most cases matrix operations with on them. Special field of neural networks is called deep learning or deep neural networks. They have two distinguishing features: high number of hidden layers and algorithms learning directly from the input data.

The recently gained huge popularity especially in fields of pattern recognition studies is due to their ability to learn basically any type of pattern. Though there are numerous types of networks and each type can be modified virtually infinitely, many principles, algorithms and terminology is mostly shared across all of them. In this chapter we will discuss the theory behind basic neural networks and later focus on a special category used for image recognition – Convolutional neural networks.

3.1 Basic Theory

The foundation of a human brain are neurons. Connected one to another, they transmit electrical impulses from our 'sensors', such as eyes, ears, nose etc. and pass them onto other neurons. According to the importance – in mathematical language *bias*, if a neuron receives electrical impulse, it either fires another impulse to next neutron, or does nothing. At the end of this neural network, the human brain receives an information based on many sub-decisions and takes adequate action.

Similarly, artificial neural networks created by mathematical algorithms are created by many artificial neurons, connected in layers. The first type of artificial neuron was a perceptron, developed as early as in 1950s. It takes several inputs $x1, x2, \ldots$, and produces single binary output, as we can see in Figure 3.1. Every input has associated

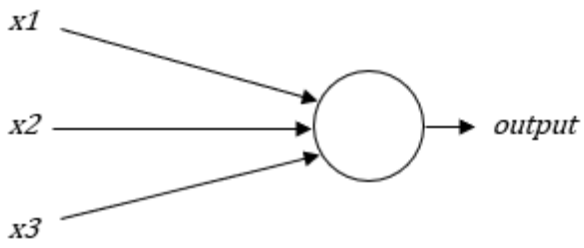

Figure 3.1: Perceptron with 3 inputs

weight to be multiplied with and giving the input 'importance' as with real brain. The mathematical formula of the perceptron is

$$f(x) = \begin{cases} f0, & \sum_j w_j x_j \leq \text{threshold} \\ 1, & \sum_j w_j x_j > \text{threshold} \end{cases}$$

where wj represents all the weights and threshold is a chosen value. In modern literature a different equation is commonly used to describe perceptrons:

$$f(x) = \begin{cases} 0, & \boldsymbol{w} \cdot \boldsymbol{x} + b \leq 0 \\ 1, & \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \end{cases}$$

The weights and inputs are represented as vectors and instead of threshold a bias is introduced, while $b = -threshold$.
Perceptrons form layers of an artificial neural network. In Figure 3.2 we have an example of a neural network with one hidden layer. These are called 'deep' networks and the number of hidden layers depends on the use and choice of a programmer. There are five inputs entering the algorithm, each connected with all neurons of the first layer. Decisions of the first layer is then carried into the hidden layer; each neuron being connected to all four in the hidden layer. This is called *fully connected layer*. Finally, decisions of the hidden layer is evaluated in a single output neuron.

The key feature of every neural network is its ability to learn. For a certain input there are desirable outputs. For more complex cases the weights and biases are set by learning from input values with known output. But for the learning to be effective, it must be done by small amounts and the perceptron's ability to only output 0 or 1 makes it unusable. Instead we use neuron type called *sigmoid neuron*.
Sigmoid neuron has one or more inputs and single output like perceptron, as seen in Figure 3.1. However, the output is modified by a sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3.1}$$

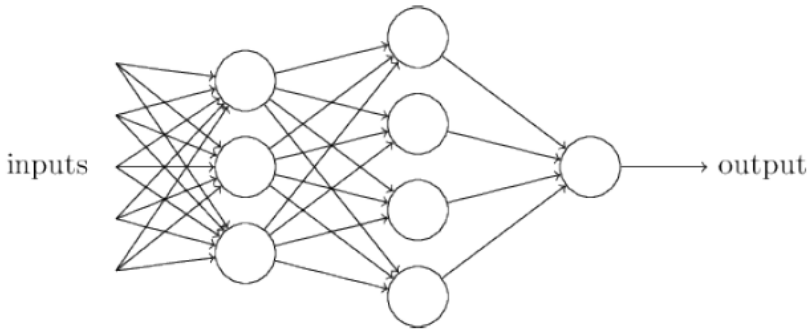

Figure 3.2: Neural network with two hidden layers (taken from [23]).

Multiplying the sum of weights with the sum of values with added bias. The output of sigmoid neuron can be in closed interval $< 0, 1 >$, as illustrated in Figure 3.3. Resulting equation of the sigmoid neuron output is

$$\frac{1}{1 + exp(-\sum_j w_j x_j - b)} \tag{3.2}$$

with wj being the sum of weights and b bias. This is also called *activation function* as it defines the conditions of when the neuron should 'fire'. Sigmoid is one of the most common used, more will be introduced later in this thesis.

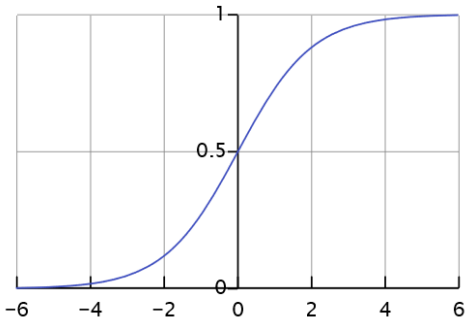

Figure 3.3: Sigmoid function (taken from [23]).

3.1.1 Learning process

When constructed, the neural network does perform well. Individual weights and biases are set in a way that the output value does not correspond with the desired value. We quantify this error and use it to correct the parameters to achieve better accuracy. Since there are hundreds to billions of parameters, depending on the architecture, we calculate the changes in opposite direction to data flow using the dependencies between neurons.

In the initiation process, the weights and biases are usually set to random within a

specified interval. Since it is nearly impossible to determine how they should be set even with simple architectures, the network must work its way to adjust them. Using input data with known output, we can find how badly the network performs. To quantify this amount of error, we define a cost function:

$$C(w,b) \equiv \frac{1}{2n} \sum_x \parallel y(x) - a \parallel^2 \tag{3.3}$$

Here, w stands for all the weights in the network, b for all the biases, n is the total number of training inputs, a is the vector of outputs from the network for x meaning input, and the sum is over all training inputs. C is basically a variant of a mean squared error function, or shortly MSE. [23] This function is non-negative, since every term in the sum is non-negative. The cost function becomes smaller, i.e., $C(w,b) \approx 0$ when $y(x) \approx a$, which is ultimately the goal of learning. Therefore, our next step is to find algorithm to minimize the cost function $C(w,b)$. In other words, we want to find a set of weights and biases which make the cost as close to zero as possible.

For simplification, let's substitute two multi-dimensional variables w, b with two-dimensional one, v. We are now searching for the global minimum of a cost function $C(v)$. Because neural networks have millions or billions of weights and biases, using analytical approach to solve this problem would be extremely complicated. Instead, we use analogy of a valley (our 2D function) and a imaginary ball rolling down in it. We need to find the direction that is the steepest for the ball to reach the minimum. For two-dimensional cost function, the C changes as follows:

$$\Delta C \approx \frac{\delta C}{\delta v_1} \Delta v_1 + \frac{\delta C}{\delta v_2} \Delta v_2. \tag{3.4}$$

The goal is to find Δv_1 and Δv_2 so that the ΔC is negative. We define Δv to be the vector of changes, $\Delta v \equiv (\Delta v_1, \Delta v_2)^T$ and the gradient of C to be the vector of partial derivatives, $\left(\frac{\delta C}{\delta v_1}, \frac{\delta C}{\delta v_2}\right)^T$. Now, we get the gradient equation for two-dimensional C:

$$\nabla C \equiv \left(\frac{\delta C}{\delta v_1}, \frac{\delta C}{\delta v_2}\right)^T. \tag{3.5}$$

The previous equation now can be rewritten as

$$\Delta C \equiv \nabla C \cdot \Delta v. \tag{3.6}$$

The gradient of C relates changes in v to changes in C. This lets us see how to choose to make negative. The neural networks implement one other parameter – the learning

rate . It is small non-negative number to multiply with to reduce the changes in order not to "overshoot" the minimum. Finally, we get the equation

$$\Delta v = -\eta \nabla C \tag{3.7}$$

to calculate how we need to change the weights and biases to reduce the cost function. This is done repeatedly, over chosen number of *epochs*. One epoch embodies running a series of learning and validation data through the neural network. We can see the effects of choosing different learning rates in Figure 3.4. If we set it too high, though in first few epochs the cost function will decrease significantly, it will not get very close to the minimum because of large steps. Oppositely, one can say that very low learning rate would eradicate the 'overshooting' problem and most certainly lead to the minimum, but such rates would require many epochs, would be very time-consuming and may get stuck on quite low accuracy.

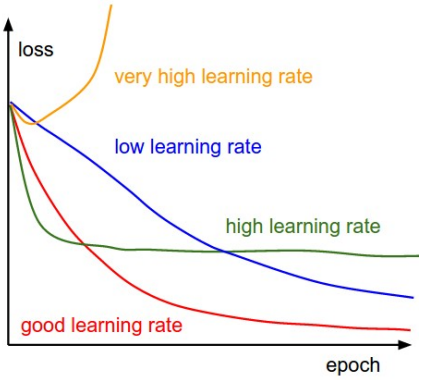

Figure 3.4: Effect of learning rate on cost function (source: http://cs231n.stanford.edu/).

3.1.2 Backpropagation

The algorithm used for calculation of the gradient descend is called backpropagation. Using the output of a neuron cell, known activation function and structure of the architecture, we define the equation combining the cost function and activation function equations.

Before we can define the backpropagation equation, we must index individual layers and neuron cells. First, we define L as a layer index, starting with 1 for the input layer, written as superscript. Neuron cells are indexed with subscript starting at j for the last layer before the outputs, k for the cells before etc. Weight of the connection between cell k in layer l–1 and cell j in layer l is $w_{jk}^{(l)}$. Now, we can define input of a single cell:

$$z_j^{(l)} = \sum_{k=0}^{n-l} w_{jk}^{(l)} a_k^{(l-1)}, \tag{3.8}$$

summing all weights and activation functions connected to cell j in the layer l. The output of this node is the result of a chosen activation function $g^{(l)}$ in the node, therefore:

$$a_j^{(l)} = g^{(l)}\left(z_j^{(l)}\right). \tag{3.9}$$

We can now use this output with the equation for cost function, expressing it as $= C_{0j}\left(\left(a_j^{(l)}\right)\right)$. Substituting for $a_j^{(l)}$, we get $C_{0j}\left(a_j^{(l)}\left(g^{(l)}\left(z_j^{(l)}\right)\right)\right)$. According to the dependencies throughout the equation, we make derivatives:

$$\frac{\delta C_0}{\delta w_{jk}^{(l)}} = \left(\frac{\delta C_0}{\delta a_j^{(l)}}\right)\left(\frac{\delta a_j^{(l)}}{\delta z_j^{(l)}}\right)\left(\frac{\delta z_j^{(l)}}{\delta w_{jk}^{(l)}}\right), \tag{3.10}$$

which results in

$$\frac{\delta C_0}{\delta w_{jk}^{(l)}} = 2\left(a_j^{(l)} - y_j\right)\left(g'^{(l)}\left(z_j^{(l)}\right)\right)\left(a_j^{(l-1)}\right). \tag{3.11}$$

This equation calculates the cost function for a single weight for one single training example. To calculate over n training examples, we calculate the average derivative of the cost function over the examples. Then, we expand the calculations over all remaining weights and calculate it over many epochs, until a pleasant accuracy is reached.

3.2 Convolutional Neural Network

Classical architecture of a neural network is not optimal for images. Since an image is a set of hundreds to tens of thousands of pixels, with each being an individual input, we could either handcraft some important features from them or process the value of every pixel. Both would be extremely computationally demanding and/or inaccurate. Therefore, specialized layers are introduced to process efficiently, with the use of convolution, subsampling and different activation functions. In contrast to older neural networks where features are handcrafted, deep convolutional neural networks learn features by training on large sets of data.

3.2.1 Convolution Layer

In order to recognize various patterns throughout the image, we use a set of filters via convolution operation. These differ from simple shapes such as straight and curved lines, dots, circles etc. to complex ones in the deeper, hidden layers. Each used filter reveals presence or absence of a certain pattern in the image and this information is then transferred to following layer. [23] The main reason we are using convolution instead of flattening the image to a one-dimensional array of pixel values is that the image carries spatial information as well. Flattened array of values would result in the network treating the distant pixel the same way as those next to each other. Using multiplication through a moving window we test response of the whole image to the filter. The resulting convoluted image has a single bias and single weight, meaning that in final decision-making we are adjusting the importance of the filter and not the pixels themselves. [23]

Setting the parameters of the convolution layers must be done manually and is crucial to the performance of the whole network. We pick number of filters, filter size and window stride. Window stride is usually 1 to preserve the most amount of information, filter size depends on resolution of images we are working with, in [23] we for example have 5x5 pixel filters for 28x28 images. Filter size gets lower for each following layer, in contrast with number of filters, which increases. Both is correlated with subsampling in the pooling layers, which follows the convolutional ones. Two-dimensional convolution with stride 1 can be seen in Figure 3.6.

The two-dimensional convolution formula is as follows:

$$g[m,n] = \sum_j \sum_i \omega[i,j] \cdot f[m-j, m-i], \tag{3.12}$$

where g is the filtered image, f the original image and ω the convolution kernel. Convolution is commonly used in other image processing applications, such as sharpening, blurring, edge detection etc. In these the convolution kernel can be set to overlap the original image which preserves the image dimensions. Since this cause some image distortion, it is undesirable for our purposes. Some examples of convolution kernels for these applications can be seen in Figure 3.5.

For purposes of constructing the neural network, the formula to calculate the output size of the convolution is

$$\text{output width} = \frac{W - F_w + 2P}{S_w} + 1 \tag{3.13}$$

$$\text{output height} = \frac{H - F_h + 2P}{S_h} + 1 \tag{3.14}$$

where W, H stands width and height of the original image, for $F_{w,h}$ for filter width and height, P for padding and $S_{w,h}$ for stride.

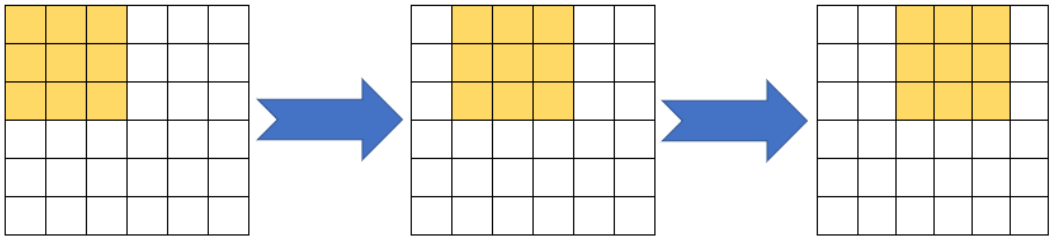

Figure 3.5: 3x3 kernel moving with stride 1 pixel through the image

Results of a convolutional layer is series of filtered images, each having assigned single bias and weight. Each of these provides information about different pattern being present or absent in the image and through the process of learning the network changes the bias and weight of each filter's neuron depending on how it affects the recognition.

3.2.2 Pooling layer

Following the convolutional layer, to reduce the amount of processed data, the pooling layer is introduced. There are two common types of pooling – max pooling and mean pooling. The processing done within this layer is fairly simple; it uses square moving window with a stride equal to its dimension and computes the mean of all values inside the window (mean pool) or takes the highest value as the result (max pool). Although mean pooling preserves more data because the final value comes from averaging all in window, since the convoluted image has higher values where feature detection happened, it is desirable to keep this high value information and discard the rest, reducing noise generated by unwanted detection. Most common pooling is with 2x2 window and stride 2, therefore reducing the dimensions of convoluted image matrices by 2. Pooling was proven to be an effective method of decreasing the computational demands while preserving enough information to make the network accurate. Visualization of pooling process can be seen in Figure 3.6.

3.2.3 Activation Function

We already introduced one of the basic activation functions, the sigmoid. However, in convolutional network architectures, the most common functions used are *ReLu, tanh* and *softmax*. Each one has different purpose and is suitable for different architecture or layer

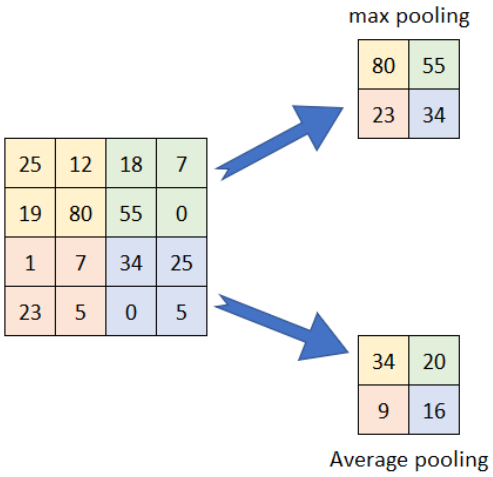

Figure 3.6: Average and Max pooling 2x2

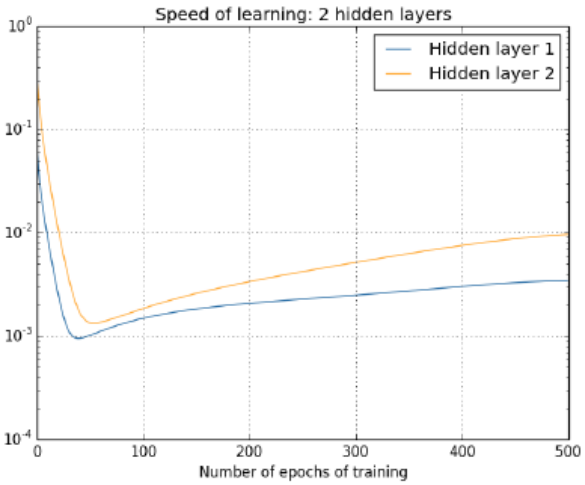

Figure 3.7: Speed of learning of 2 hidden layers (taken from [23]).

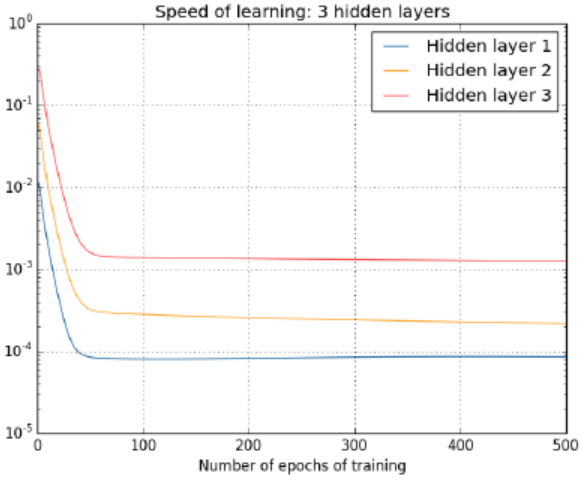

Figure 3.8: Speed of learning of 3 hidden layers (taken from [23]).

Starting with *tanh*, it is very similar to the sigmoid function. In fact, the relation between them is approximately $tanh(x) = 2\sigma(2x) - 1$. The has a sharper transition around zero values and the output is $< -1, 1 >$. This means that both sigmoid and tanh function suffers from *vanishing gradient* problem. As we calculate gradient descent of a cost function 'further away' from the output, with more neuron cells incorporated in the equation, the sigmoid or tanh functions saturate, slowing the process of learning after given number of epochs. We can see development of this trend in Figure 3.7 and 3.8., taken from [23], which is contrary to expectations that deeper networks would reach higher accuracy with every added hidden layer. The figures show how the learning speed decreases differently depending on the number of the hidden layers and their order.

To tackle this problem, some papers proposed [1],[22] using the rectified linear unit (ReLu) activation function. The output of a rectified linear unit with input x, weight

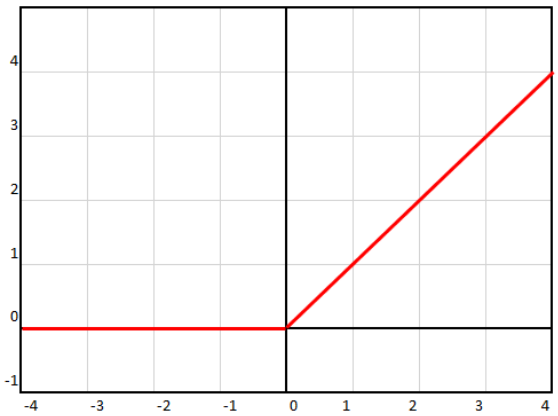

Figure 3.9: ReLu function

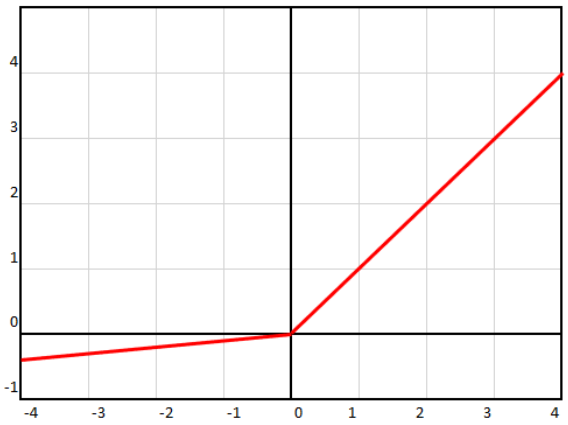

Figure 3.10: Leaky ReLu function

vector w, and bias b is given by

$$f(x) = max(0, w \cdot x + b). \tag{3.15}$$

The function turns all negative inputs into zero and non-negative without change. We can see it graphically in Figure 3.9:

ReLu provides faster learning for deep neural networks and practically cancels out the vanishing gradient problem [1]at ReLu can set parameters for some neuron cells so that they never fire again, effectively 'killing' them. Solution for this problem was introduced in a form of a leaky ReLu. The difference is that the negative values are not zeroed-out but instead they are given proportionally very small negative value. The output values of leaky ReLu are defined by

$$f(x) = \begin{cases} \text{x} & \text{if } x > 0 \\ a \cdot \text{x} & \text{otherwise.} \end{cases}$$

Leaky ReLu is represented graphically in Figure 3.10. Leaky ReLu with different values for a were evaluated against standard ReLu in [2]. The results show slight improvement in learning speed for leaky ReLu with and significant improvement during training and testing. The value of a plays significant role as with low value of $a = 0.01$ the improvement was about 1-2% but for $a = 0.18$ the improvement was about 10%. The paper [2] also mentions Randomized Leaky ReLu, which randomly assigns value to a within defined span in the training phase. This type of non-linearity yielded very similar results as Leaky ReLu.

While ReLu is commonly used nowadays as it replaced sigmoid function, it is not the most suitable for the output layer, where we prefer to see probability-like distribution

to get the clear picture. For this purpose, a *softmax* function was designed to be a new output layer on neural network. Unlike sigmoid or other functions used in output layers, the sum of all output neuron cells in softmax layer is always 1. It also tends to create bigger value difference between the highest value and the rest of them.

Therefore, we can refer to it as a probability distribution, which is mostly convenient for neural networks when output is supposed to be single value. The formula used for softmax function can be written as

$$a_j^L = \frac{e^{x_j^L}}{\sum_k e^{z_k^L}} \tag{3.16}$$

where x_j^L stands for input of j-th neuron cell and $\sum_k e^{z_k^L}$ being sum over outputs over all neuron cells in output layer.

3.2.4 Dropout

Another way to eliminate or lower overfitting problem is to use dropout. According to user-set rate, a layer with added dropout randomly drops neural cells, making the architecture different with every epoch. The dropout rate d represents the probability of a cell being ignored, with $d = 1$ being dropout-free and $d = 0.5$ dropping half of the cells in a layer. As stated in [31] ignificantly improves the generalization of a network no matter the specialization at a cost of slowing the learning time. To reduce the rising time, Srivastava et al. [31] recommend increasing the learning rate 10-100 times. Also, it has been found out that the typical value of dropout depends on the depth of the network, with lower dropout rate requiring deeper nets. Too large d may not prevent overfitting. For real-world inputs the typical dropout rate was set to be 0.8.

3.3 Building Deep CNN

With theoretical basics covered, we can now implement our own neural network. The development in this field went hand in hand with creation of many tools for building neural networks, with most of them being user-friendly and available for free.

3.3.1 Programming Languages and Frameworks

Most popular programming language for neural network implementation is Python , since it is one of the-, if not the most beginner-friendly language and it's globally available for

free. For data science especially, a framework called NumPy was created, simplifying most of the code needed to build neural network. For further simplification, framework named Keras [7] was created, allowing everyone to build a neural network within tens of lines of code. Most of the papers cited in this thesis uses these tools. Additionally, MATLAB released Deep Network Designer, graphical application for creating neural networks by composing blocks corresponding with particular layers, activation functions and more. The application also has revision tool called `analyzeNetwork`, which can determine flaws in code. As a result of the network created by this application, MATLAB automatically creates all scripts and functions required for evaluating the network using data of choice. The whole process of designing network in MATLAB is very intuitive and therefore will be used for the practical part of this, thanks to licensing provided by the Czech Technical University.

3.3.2 Cloud-based solutions

Although the computation power increased to the point where more-less anyone can build, train and use neural network, some projects are still far too large to be handled on consumer-grade computer or even servers in decent time frame. In recent years, the giants of computing industry Google, Amazon and Microsoft have created cloud-based computing solutions, allowing subscribers take advantage of their enormous servers for storage and computation power. They also introduced their own frameworks for Deep learning, namely Google TensorFlow [19] and Microsoft ML.NET [8].

3.3.3 Popular Datasets

One of the reasons of machine learning's growing popularity is the availability of vast amounts of data. This is also the reason while some of the most capable frameworks come from Google and Microsoft. For training of convolutional neural network, we need set of images with corresponding labels, dividing them into categories. One of the largest databases, created for image classification, is ImageNet.org, community-driven downloadable database, currently composed of over 14 million labeled images across 21k synsets. For purposes of recognizing handwritten numbers, a MNIST database has been created, containing over 50k labeled images.

For purposes of image quality assessment however, the amount of data we have today is less than optimal. While recognition is fairly straightforward task for humans, assessing the image quality is very subjective. The most common used method is obtaining opinion score from human subjects, who give the image a score from 1 (excellent) to 5 (bad) either by determining the quality of the image by itself or comparing it to other images. This

process requires controlled environment with expensive equipment and lots of time to be valuable. Therefore, some of the researchers decided to for other methods of assessing the quality of images. The most used and popular datasets with their parameters are listed below.

- **LIVE image database** - developed by Laboratory for Image and Video Engineering at University of Texas, this database is one of the evergreens in modern IQA papers. Usually every initial testing of an algorithm is first done on images from this database, as well as training in case of non-transfer learning approach. There are currently two releases with the newer one having 29 reference color images with typical resolution of 0.4 megapixel, 169 distorted by JPEG compression, 175 by JP2K compression and 145 by white noise, bit error in JP2K and Gaussian blur each. All distortions are done in post processing.
- **LIVE In the Wild Challenge** - this database contains 1162 images taken using typically mobile camera devices and contain authentic distortion caused by technical limits of the camera or poor skills of the cameraman, therefore usually containing multiple distortions. Database was evaluated by over 8000 human observers.
- **TID 2008** - this database originated from Kodak lossless true color image suite by applying 17 types of distortion at 4 level each. Combined the image includes 1700 images evaluated by 838 observers, obtaining over 250 thousand of opinion scores. The distortions include 7 types of noise, JPEG and JP2K compression and transmission errors, mean shift, contrast changes and more.
- **TID 2013** - updated version of TID 2008, adding 7 more types of distortion and one more level to each one, making up a total of 3000 distorted images. Added distortion types include color distortions, dithering, chromatic aberrations and more.
- **CSIQ** - database developed at Shizuoka University containing 30 pristine image turned into 900 distorted ones. The applied distortion are JPEG and JP2K compression, adaptive and pink Gaussian noise, Gaussian blur and global contrast decrements, all within 5 levels of severity. The database was rated by 35 observers and obtained over 5000 opinion scores.
- **Waterloo Exploration** - this database, unlike others, was not evaluated by human subjects using opinion score. Instead, the 4 744 pristine images and 94 880 distorted ones were tested on 20 different full- and no reference algorithms, using pristine/distorted image discriminability test, list-wise ranking consistency test and the pairwise preference consistency test. The distortions used were JPEG and JP2K compression, white Gaussian noise and Gaussian blur, each in 5 different levels [18].

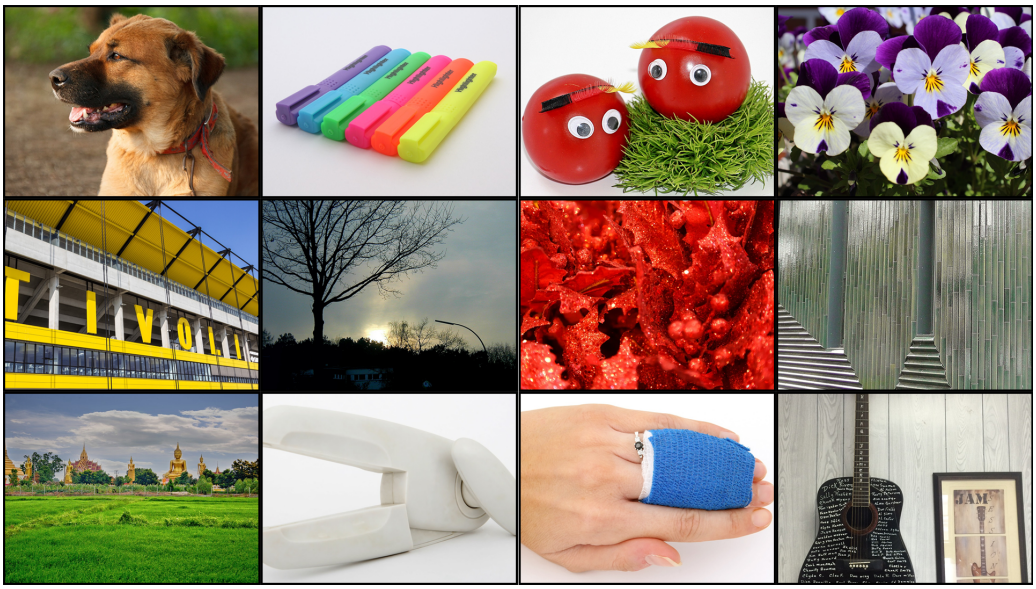

Figure 3.11: Selected pristine images from database Kadid 700k

- **KADID 10k** - full name Konstanz artificially distorted image quality database - this database was published in 2019 and unlike the previously mentioned was used only a few times. This database contains 81 pristine images to which 25 types of distortion, including noises, blurs, color distortions and more, were applied in 5 different levels, resulting in 10 000 distorted images. The parameters of the distortion algorithm were chosen manually so that the resulting image quality perceptually corresponded to rating from 1 to 5 by standards of opinion score. Using crowdsourcing, the authors obtained 30 opinion scores for each image.

- **KADID 700k** - a set of 140 000 images was picked from Pixabay database (the 81 images selected for KADID 10k were the best looking of those). From those, another 700 000 distorted images are generated with same types of distortion, but unevenly, creating 5 distorted variations of every image, each containing one random distortion type with level of degradation corresponding to MOS 1 to 5. This makes the KADID 700k the most extensive database ever created [15]. Researchers can download the pristine 140 000 images, having size of estimately 45 gigabytes and then run MATLAB script to generate the 700 000 images, which take over 300 gigabytes of space.

All mentioned databases are publicly available for download and testing, with detailed instructions included in the package and additional information or research papers on the website. You can find links to all of them in the Appendix - External links.

Figure 3.12: Selected images with the most severe distortion settings from database Kadid 700k

Chapter 4

Neural Networks based BIQA Methods

For the last few years, the research in deep learning field has grown vastly and image quality assessment has become widely explored field of study. Expanding the basic architectures of convolutional neural networks by adding processing to input, output or modifying data flow within the net, many studies are outperforming classic non-machine learning approaches. Universal metrics for evaluating the methods are those which determine similarity between mean opinion score of tested images given by human subjects and that which was given by program, with aim to be identical. In this section we will describe various types CNN-based image quality assessment methods, divided to subsections based on how the network is implemented, what is the output, how the training is done, or the data flow is modified.

4.1 Preprocessing of the Input Data

4.1.1 Image Normalization

Preprocessing the image by normalizing them reduces or completely ditches the data not required for quality assessment, or those which have very little impact. Inspired from [20] and [41], local contrast normalization is implemented in [13] with a window of 3x3 pixels, since it was found out that larger windows worsen the performance as well as non-local contrast normalization. Local contrast normalization is frequently used in CNN object recognition problems as it makes the algorithms more robust to luminance and contrast variations.

In [11] the normalization is done by turning input images into grayscale and a low-

pass filtered version of the image is subtracted from the original. This is done due to fact that the low frequency information are not affected by distortion; for example Gaussian noise add high frequency components, blur removes high frequency details and JPEG and JP2K block artifacts introduce high-frequency edges. Also, HVS is little to not sensitive to low frequency changes. Kim et. al. [11] state that the normalization does cut some information out of the network, which is compensated by adding two handcrafted features in the second training stage.

The choice of turning image from color into grayscale pops up in almost every paper. While some claim that the color information is important and it makes the network perform better, others oppose that our perception of color quality is very subjective and color fidelity to real world plays little to no role in the network feature extraction.

4.1.2 Gradient of the image

Some distortions are strongly associated with colors and intensities of the pixel. Changes in the high frequency information due to these distortions are significantly reflected in image gradient. This fact is explored in [25] by creating two identical streams of neural network, one processing patches of the original image and the second one the patches of a gradient of mentioned image. Authors claim that this approach increases sensitivity of the network to blur and compression types of distortion, since the loss of high frequency details is in these cases the most significant. The network processes features of the image stream and gradient stream independently up to two final fully-connected layers where those are aligned next to each other, creating the final opinion score. You can see the input images and their corresponding gradients in Figure 4.1. The gradients of the images distorted by blur shows significantly less detail than the sharp ones.

Figure 4.1: Input images of a neural network and their corresponding gradient (taken from [25]).

4.1.3 Color Adjustments

Another approach of preprocessing images lies in changing the color space of an image. Digitally stored images are standardly stored in RGB additive color space [10], where three digits from 0 to 255 represent the intensity, saturation and hue. For the task of IQA this approach may not be ideal since determining these values can not be done directly from the numbers and because these are dependent on all three numbers, i. e. changing one of the component numbers affect luminance, saturation and hue all at once.

Solution to this problem is presented in [40]. Authors create multi stream neural network processing the original image as well as image transformed into three more color spaces. The first color space is CIE LAB, consisting of one luminance and two color components. LAB can describe wider gamut of colors and non-linear relationship of luminance-color components is similar to the way HVS perceives this information. CIE LMS, also resulting from studying HVS, represents color in a similar way to LAB, having one luminance achromatic component and two chromatic. The last used color space is CIE HSV, representing colors as hue, saturation and value components.

The multi stream neural network has each channel of each color space as a single input, fusing the features from each processed image at the end of the network. Visualisation of the different color spaces and it's components can be seen in Figure 4.2.

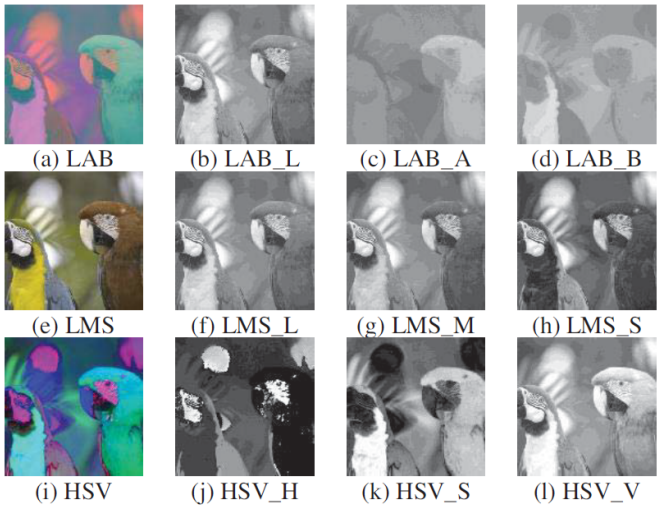

Figure 4.2: Color spaces and their components (taken from [40]).

4.2 Learning the Models

One of the biggest problems of deep learning is need for vast amounts of data. For image classification, datasets must be evaluated by many human observers to provide reliable

ground truth. Unlike image recognition databases, most databases for image quality assessment, until very recently, consisted of mere thousands, in some cases even hundreds of images. To tackle this problem, there are generally two ways: augmenting the datasets the databases or using pretrained models trained on different databases, such as image recognition, where millions of images are labeled.

4.2.1 Augmenting the datasets

The more common way of augmenting existing image databases is slicing the image into patches. It is also useful to have image of square aspect ratio and amount of pixel forming the side to be , with usually being from 5 to 8. The patches are created as overlapping or non-overlapping. While this can increase the database size dramatically, the main problem is with the MOS, since the distortion is rarely homogeneous. Although we can label all the patches with the same MOS as the full image, the training result may be affected. In [34], this problem is tackled by creating large patches, randomly selecting areas about 80% of the size of original image, to preserve variety and context information. A different way of tackling this problem is proposed in [39], using context-aware algorithms to determine the relevance of the score of each patch, resulting in higher accuracy. Another used augmenting method is flipping the images horizontally and vertically and rotating by 90, 180 or 270, with the MOS remaining the same for all listed variants. This method provides images with the MOS of same reliability as the original image, but can only expand the database 6 times, which may not be sufficient.

4.2.2 Transfer Learning

This method takes advantage of the fact that image quality assessment and image recognition are fundamentally similar processes. Since convolutional neural networks are being used for image recognition for some time now, there are databases of millions of images that have already been fed into these networks, mainly AlexNet [1] and VGG16 [37]. With this fine-tuning, the whole models can be used as a backbone to more complex structures, they can be modified, and their learned features can be extracted from any layer. It was shown that in many cases the features learnt for purposes of classifying images can be used for image quality assessment [26], [29], [40]. More on the use of this method is described in following sections. The downsides of this method can be that the images must be either downsized to math the resolution of the CNN, which in case of [1] and [37] is fairly small and some distortion types may be altered, or turn into patches, which can oppose problems mentioned in previous subsection.

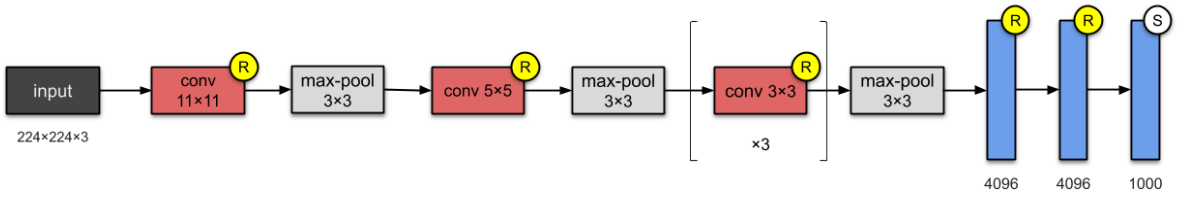

Figure 4.3: Architecture of AlexNet, taken from [1]. R means that the layer is followed by ReLu function, S stands for Softmax function.

4.3 Stream Complexity

Another commonly explored attribute of a neural network is the complexity. The network can be theoretically infinite in terms of simultaneous streams of feature extraction and their processing, as well as with number of layers. However, with increasing the complexity we also increase the computational demands and the link between complexity and performance is not linear.

4.3.1 Depth of the Network

The number of hidden layers has a critical effect of the performance of a neural network. One of the first well-performing nets AlexNet [1] had 5 convolutional layers, 3 pooling layers and 3 fully connected layers, which meant more than double from the previously published network and the first one to be called 'deep'. The second significant deep network VGG16 [37] had 16 hidden layers (hence the name), which was one of the reasons it performed better. After that, the number of proposed network architectures grew and the number of hidden layers was pushed forward. One of the deepest used is the InceptionResNetV2 with over 500 hidden layers [38]. Interestingly, this network has merely a third of the trainable parameters compared to older VGG16, meaning that the layers contain less filters and dimensions.

In IQA research papers the most used architectures are the simpler ones, as they are modified or serve as a part of new network. The depth is certainly affecting the performance. The networks proposed in all papers have a minimal depth of 10 hidden layers and the shallower ones were always combined with image pre-processing or additional feature processing. But to determine exactly how deep the network should be in combination with other parameters is far too complicated.

4.3.2 Multiple-Stream Network

Natural images have many aspects of quality, such as contrast, exposure etc., which can be seen with a naked eye and represented by various software using algorithms, but are not directly represented in the image file. In order to explore those 'hidden' features and simultaneously not losing other information due to conversion, many papers have decided to create multiple stream networks. The common approach is that the architecture is built and used multiple times simultaneously, with differently pre-processed input image. The conjunction of these streams usually occur before fully-connected layers where features are combined, resulting in a single opinion score number.

To explore the differences in pre-processing images and how they would perform when combined, Yuan et. al. [40] proposed multi-stream network with maximum of 15 streams in the most complex architecture variation. Image preprocessing is done by conversion into different color spaces, as mentioned before in more detail. The network in every stream is based either on AlexNet [1] or VGG16 [37], with 5 convolutional layers and 5 fully connected layers instead of the original 2. The learned features in the last layer of every stream are concatenated into single fully connected layer, the dimension is reduced using Principal Component Analysis to remove redundancy and the resulting quality score is obtained using SVR algorithm. Having tried many combinations of backbone pre-trained networks and image pre-processing, the paper states that the best result are produced by VGG16-based network pre-trained on the combination of ImageNet and Places365 databases and using all three color spaces, each with all components.

In [25] this method is used in proposal of two-stream convolutional network. The architecture is the same, one of the nets gets fed patches of the original image and the other one patches of the gradient of the image, in both cases these were made by splitting the image into 3 by 3. In addition to that, this architecture includes region-based fully convolutional network at the end of each of the streams, that concatenates the feature maps of the patches the way the were made from the original image. Finally, the data are concatenated into fully connected layers and form the image score as an output.

4.4 Processing Data from Layers

4.4.1 Feature Extraction

One of the key advantages of convolutional neural networks, as mentioned previously, is its ability to create feature maps with automatically generated filters, which are learned by specific certain types of training data. Typically, we pass these into fully connected

layers and reducing their dimensionality to single array of numbers, which result in desired output(s). However, we can also step into the data flow and extract the learned features from any desired layer and use those for additional processing and/or decision-making. This process is commonly used with networks pre-trained on large databases, such as ImageNet, Places365 or their combination. Although these are made for image recognition, not quality assessment, studies [insert many sources] shown that their features are applicable for IQA as well. The performance differences will be discussed in following sections.

One of the more complex approaches incorporating extracting of features to aid the assessment is proposed in [34]. The features are extracted from deep fully convolutional neural network (FCNN) made of 10 layers at four places to provide hierarchical set of features, from shallow to deep. According to the layer where they were extracted, features are down sampled to match the last layer of the FCNN using convolution. Then they are fed to the final regression net, transformed into one-dimensional vectors using fully connected layers and concatenated into one. From this vector, the final quality score is obtained. Architecture of this proposal is shown in Figure 4.4.

Another approach using features derived from different convolutional layers is proposed in [14]. The goal is to tackle the lack of training data, especially of non-synthetic distortion created by lens, bad processing of the camera or lack of photographer experience, and thus prevent overfitting. As a baseline, pre-trained ResNet-50 [12] is adopted and within the groups of the net the features are extracted through encoder block, consisting of convolutional and global average pooling (GAP) layers, as seen in Figure 4.5, to unify the dimensions of the features, specifically into 256-dimensional feature vectors, later concatenated and through fully connected layer delivering final image score. Original GAP and fully connected layers of the ResNet-50 are removed. For testing, multiple combinations of used features were chosen, with the most effective one being the combination of all four, across all distortion types. This proposal achieved similar results as DeepIQA [29], while outperforming (at its best) all non-deep learning methods.

4.5 Output type

The quality of an image both by the type of distortion and the amount of it. People tend to forgive certain types of imperfections more than others, in some cases they can be even found artistic or desirable, for example grain or noise in stylized photographs. Therefore, it is desirable to determine the type of distortion, as it may aid the evaluation algorithm.

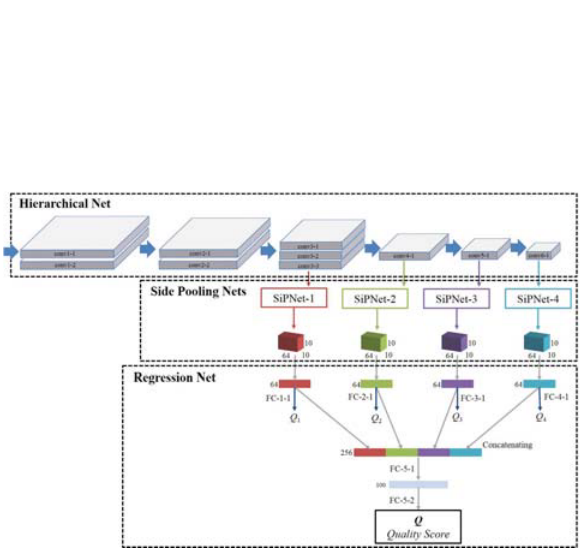

Figure 4.4: Architecture of [34].

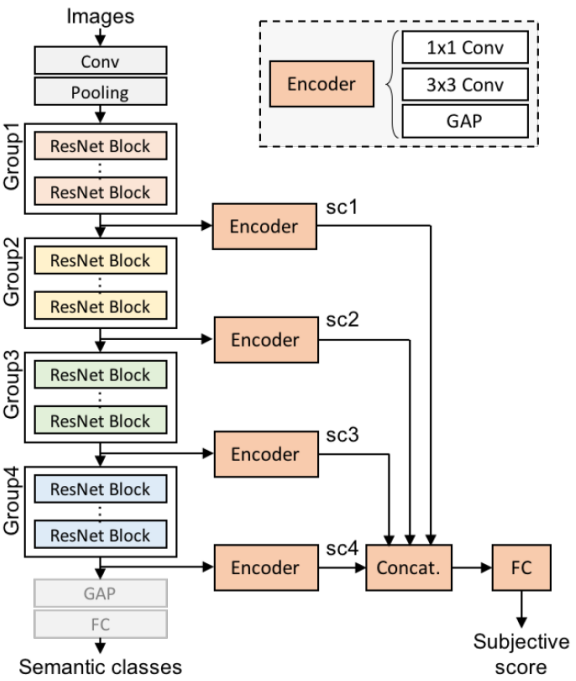

Figure 4.5: Overall structure of the proposed model in [14].

Performance of selected IQA algorithms on image databases						
	LIVE		CSIQ		TID2013	
Name	SROCC	PLCC	SROCC	PLCC	SROCC	PLCC
DIIVINE [21]	0,925	0,923	0,784	0,836	0,654	0,549
BRISQUE [20]	0,939	0,942	0,750	0,829	0,573	0,651
BLIINDS-II [30]	0,919	0,920	0,970	0,534	0,536	0,628
MEON [17]	0,951	0,953	0,839	0,850	0,811	0,828
DIQA [11]	0,975	0,977	0,884	0,915	0,825	0,850
CaHFI [34]	0,965	0,964	0,903	0,914	0,862	0,878
NSSADNN [3]	0,986	0,984	0,893	0,927	0,844	0,910
DeepBIQ [29]	0,970	0,979	0,957	0,968	-	-

Table 4.1: Performance of selected IQA algorithms on popular image databases.

4.5.1 Quality Score

Most of the networks proposed in papers on IQA produce as output quality score, which has corresponding scale to mean opinion score. The accuracy of the network is measured solely on this score, generally measured by Spearman's rank order correlation coefficient (SROCC) and Pearson's linear correlation coefficient (PLCC). You can find the SROCC and PLCC values of the proposed networks mentioned in this paper in Table 4.1.

4.5.2 Distortion classification

As a first step in distortion-specific classification and possible later reconstruction, two deep convolutional neural network architectures are proposed in [5]. They differ in number of trainable parameters, with the later one having twice the number of filters in each one of three convolutional layers and 32-times more cells in penultimate fully connected layer. The they both four output cells indicating presence of the most common distortions amongst the image databases: Gaussian blur, additive white noise JPEG and JP2K distortion. The paper states that accuracy of the distortion type determination was 92.12% for the first mentioned architecture and 94.14% for the second one, with both outperforming the benchmark method, Support Vector Machines by 10% or more.

Chapter 5

Convolutional Network with Cascaded Features

For the practical part of this thesis I chose to create a new deep convolutional neural network based on researched information and train it on the new and most extensive database Kadid 10k and partially on Kadid 700k [15]. The Convolutional Network with Cascaded Features for Blind Image Quality Assessment (CNCF) offers several aspects that add up to novelty of this approach. The first one is the branched architecture at the second half of the network to provide additional set of features. Another innovation is the combined use of ReLu and leaky ReLu for improved learning and minimized overfitting. Last but not least is the training and evaluating of the new databases Kadid 10k and 700k, which is the largest one proposed to this day.

5.1 Architecture

The input layer of the network takes 256x256 color images. Larger ones are not cut into patches but rather downscaled using MATLAB's DeepNetworkDesigner utility. The network stream network consists of 12 hidden layers with two additional in the secondary branch. The branches are later concatenated, which results in two fully connected layers and output classifier.

5.1.1 The Main Branch

Inspired by AlexNet [1], the first convolutional network uses rather big window of 6 by 6 pixels with a stride of 2 in both dimensions and 96 filters, followed by ReLu activation function and layer normalization to maintains the contribution of every feature. After

the convolution a max pooling takes place with 2 by 2 window and stride of 2, reducing dimensions by half. Then the second convolution with window of 4 by 4 and stride 1 takes place, with identical activation and normalization, followed by identical pooling.

5.1.2 Altered activation and split

Now we have 30 by 30 output with 128 filters which come through three identical convolution layers without pooling, each one having 3 by 3 window and producing 192 filters. These layers are however activated by leaky ReLu. Inspired by [2], the first layers is followed by leaky ReLu with $a = 0.02$, the second one with $a = 0.01$ and the third convolution layer by leaky ReLu with $a = 0.015$. These values were chosen to fall between the two most successful setting in the paper.

From the fifth convolution layer on, the stream is split. Main branch continues with max pooling of 3 by 3 window with stride of 2 and two convolution layers with 3 by 3 windows, each followed by standard ReLu. The side branch replaces the pooling and first convolution layer by single convolution layer with large window, to provide different set of features. From the last layers of both branches are concatenated together.

5.1.3 Concatenation and classifier

The two concatenated streams are followed by two fully connected layers. The first consists of 1024 cells and after ReLu activation the dropout layer takes the spot. As mentioned in [1],[31], the dropout layer help prevent overfitting of the neural network by randomly disabling some cell activations. Here the dropout is set to 0.5 which effectively leaves out random 50% of neural links for each iteration. The second fully connected layer contains 5 cells and is followed by Softmax function, leading into classifier. The network sorts images into five categories, corresponding with opinion score metrics. The full block scheme can be seen in Figure 5.1.

5.1.4 Image Pre-Processing

To introduce additional information about the distortion of the images, an alternative input will be tested: a gradient of the image, inspired by [25].

The original image is in the first instance converted to grayscale using function `rgb2gray`. To the product three different filters are applied: 2 custom kernels $h1, h2$ using `imfilter` function (see Figure 5.2) and `edge` function, using option 'Canny'. The first two filtered images are divided by 2 and since the `edge` function produces logical matrix, it is con-

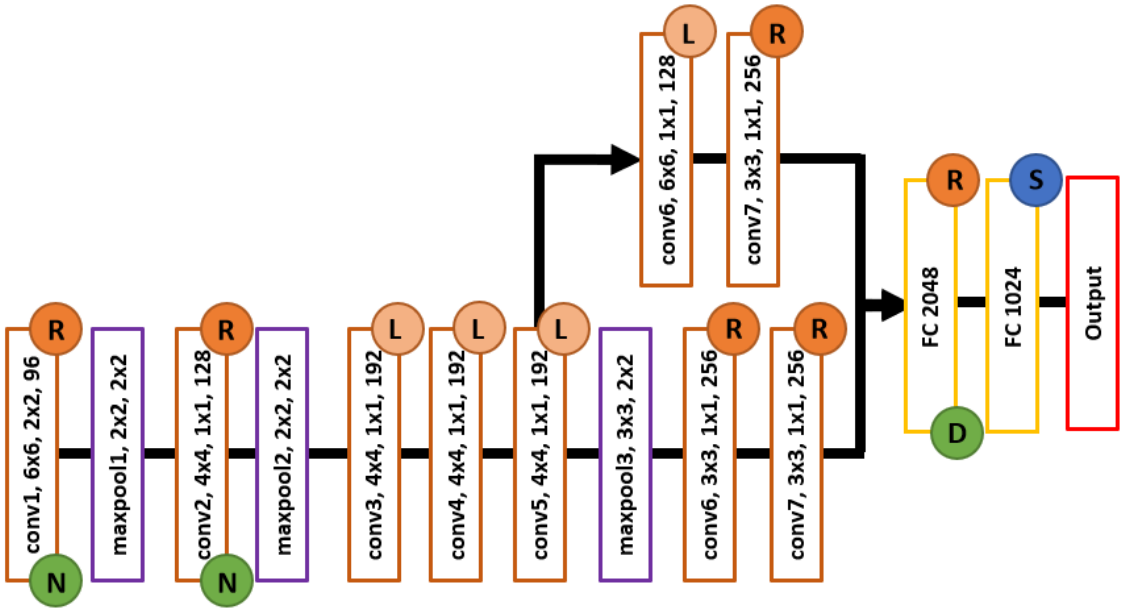

Figure 5.1: Architecture of the CNCF network. The activations in circles are: R = ReLu, L = leaky ReLu, D = dropout, S = Softmax

1	0	-1
2	0	-2
1	0	-1

h1

1	2	1
0	0	0
-1	-2	-1

h2

Figure 5.2: Kernels used to create gradient of the image.

verted to `uint8` format and the former values of zero are multiplied by 64 to achieve a quarter of maximum pixel intensity. These three altered images are combined to form a full color image using `cat` function; the image filtered with $h1$ kernel make up red channel, $h2$ produces image for green channel and `edge` function takes place of the blue channel. Resulting images can be seen in Figure 5.4.

The gradient is especially sensitive to higher frequency information [25]. In Figure 5.5 we can clearly see the decrease when blur distortion is present. Oppositely, when image is distorted by contrast decrements or additional noise, the amount of detected edges and sharp transitions increases. This alternative input may increase the networks ability to classify images. More examples of the gradient changes due to distortion in Figure 5.6.

Images go as follows: Top left: pristine image, top right: heavily distorted with blur, bottom left: gradient of pristine image, bottom right: gradient of distorted image.

Figure 5.3: Images from Kadid 700k database fed into stream with no pre-processing

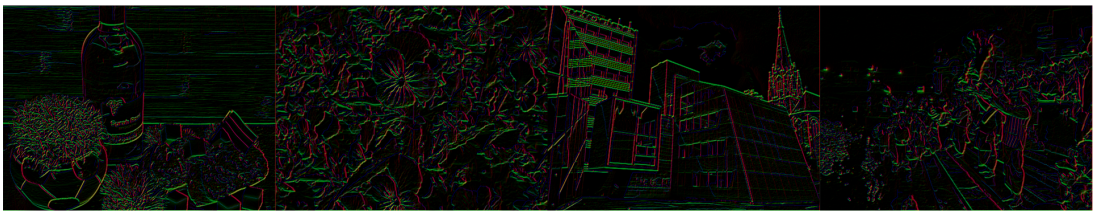

Figure 5.4: Gradient of images from Kadid 700k database for the gradient stream of network

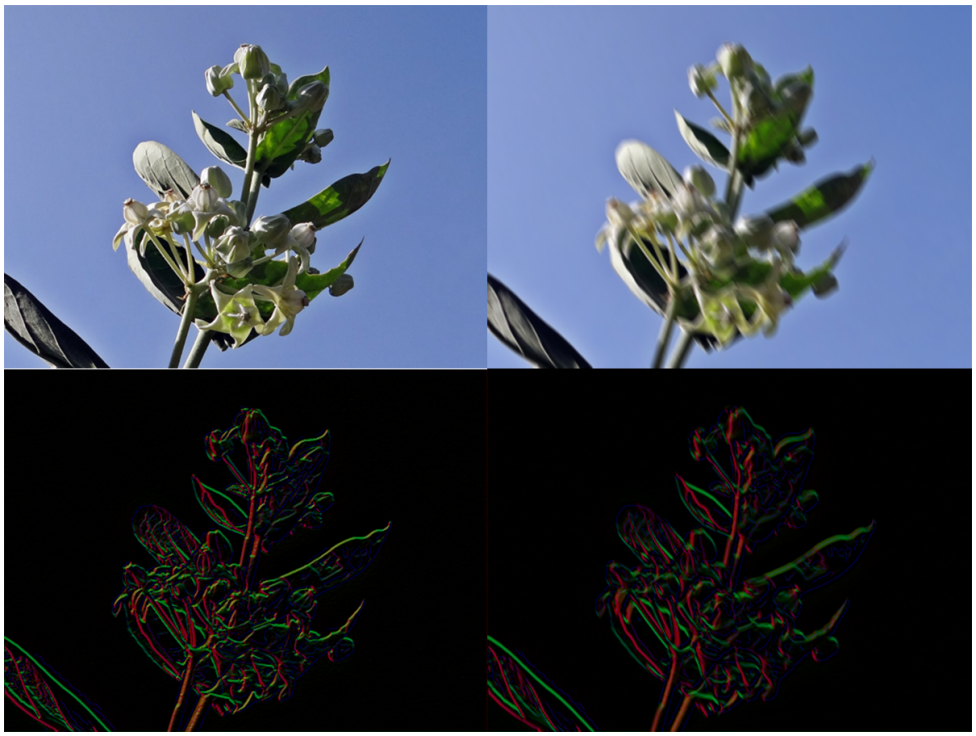

Figure 5.5: Image and image gradient affected by blur.

Figure 5.6: Image and image gradient affected by JPEG compression.

Chapter 6

Experiments

6.1 Used datasets

The unpacked Kadid 700k database after download contains pristine 140 thousand images and MATLAB code to generate the full database, picking one of 25 types of distortion randomly for each image and applying it in 5 levels, which authors fine-tuned to correspond with five levels of the opinion score [15].

For the initial training and parameter settings, I limited the algorithm used in Kadid to just four most common distortions: Gaussian blur, JPEG and JP2K compression and color noise. I processed 10 000 pristine images with 4 distortion types per 5 levels, creating 200 000 images for training and validation.

6.2 Training hyperparameters

When training neural network, the settings adjusting the process are similar for all network architectures. They are called training hyper parameters and their adjusting is crucial to proper functioning of the network.

- **Learning rate** defines how quickly the network updates it's weights and biases. The recommended value in most literature is between 10^{-2} and 10^{-6}. As mentioned before, rate set too high may result in non-functioning network, too low may result in very slow learning or not reaching accuracy that high. This parameter is challenging to determine, it depends heavily on size of your dataset, depth of the network and others circumstances. It is usually required to determine experimentally, which can be time consuming. One of the best practices is also to lower learning rate every

few epochs.

- **Batch size** is the number of images that are processed by the network at the same time. Increasing this number leads to quicker training as the CPU or GPU can process multiple items at the same time. But too large batch size can decrease the network's ability to generalize, i. e. recognize and classify images that were not used during training and validation. Commonly recommended batch sizes are 32, 64, 128, rarely higher; the number should be 2^n.
- Number of **epochs** determine how many times we process the entire training dataset of images. Good practice is to shuffle the images each epoch. Training usually requires number of epochs, although running too many can decrease the ability to generalize, especially with smaller datasets.
- **Weight decay** limits the weight updates, when the incremental values would be too big. Unlike learning rate, this affects the bigger increments more and vice versa. When deciding how big should be set, two parameters play role: the dataset size and number of trainable weights in the network. The larger your dataset is, the smaller the decay and oppositely the more weights you have, the higher decay.
- Choosing **optimizer**, the final parameter is nowadays question of two options: the Stochastic gradient descent and Adam optimizer. Adam has steeper learning curve, therefore requires lower learning curve.

6.2.1 Initial network testing

To find ideal parameters of the network, I chose images distorted by Gaussian blur, 5000 of each level to train. From the experiments, the optimal training hyperparameters for further testing are:

- learning rate: $4 \cdot 10^{-5}$
- batch size: 32
- number of epochs: 30
- weight decay: 0.9
- optimizer: Adam with default values

The obtained accuracy after 27 epochs was 85,04%. The experimental training was also done using the image gradients, but the resulting accuracy was less than 0,5% better, for which reason I chose to not use them for the main training. For initial training I

changed the values of learning rate and batch size, results can be seen in Table 6.1. Notice the achieved 20% with too high learning rate. The graph of the best performing initial training can be seen in Figure 6.1.

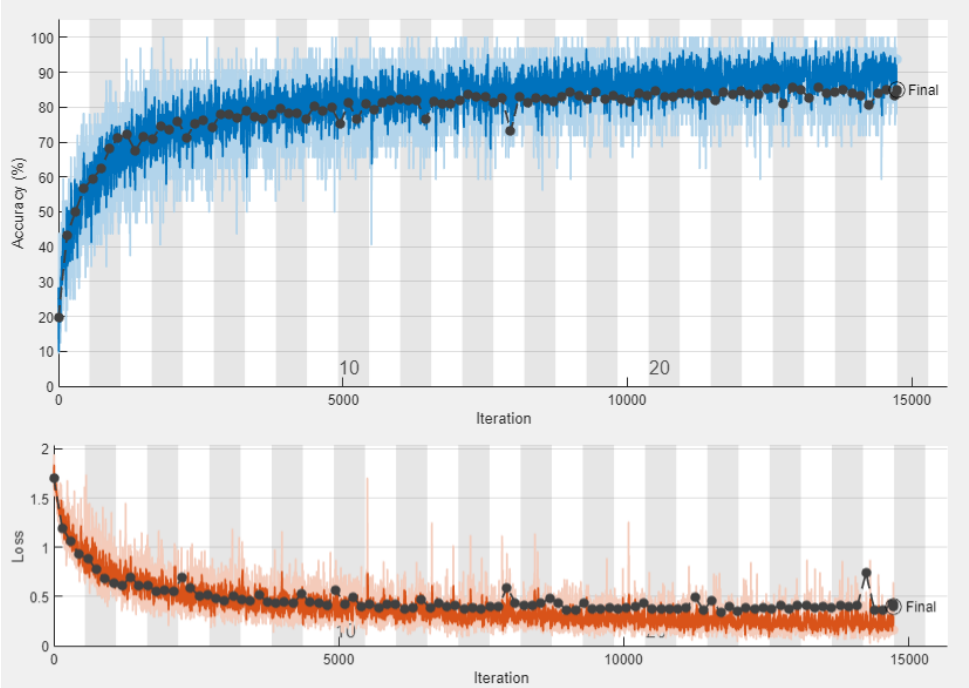

Figure 6.1: Graph of the training progress on 25 000 images distorted by Gaussian blur.

6.2.2 Main network training

After fine-tuning of the hyperparameters, I divided the Kadid distorted images into 5 groups according to set level of distortion, including all four types, each group containing 8 000 images and began training the network. As an experiment I also ran the training with the same settings but only 250 images per category.

All training and validation of the CNCF network were conducted on following hardware: AMD Ryzen 5 2600 six-core 3.6GHz, 16GB RAM, Nvidia RTX 2060 6GB. When compared, GPU provided significantly higher speed than CPU.

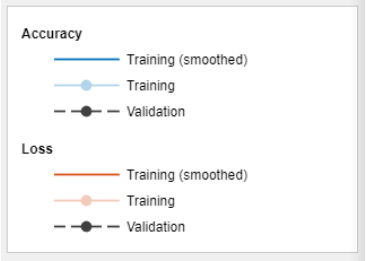

Figure 6.2: The legend of the graphs of training progress

Parameter settings for CNCF		
Batch size	Learning rate	Accuracy
32	0,0001	79.8%
64	0,0001	77,39%
32	0,001	20%
32	0,0005	76,75%
32	0,00005	**85,04%**

Table 6.1: Experimental settings of learning hyperparameters.

6.3 Results

6.3.1 Full reference validation

To evaluate the results of the neural network, we need to verify the opinion score of the generated set. As with the full Kadid 700k generated with unaltered code, the dataset generated for my purposes does not have assigned opinion score and must be evaluated by metrics. Since we have the pristine images, we can use full reference metrics as they are outperforming the blind ones. The SSIM index and VIF index [32] are the right tool for the job, the first one is available as a standard function in MATLAB and the second one is publicly available. The VIF index requires grayscale image so `rgb2gray` function was used. The script for group evaluation of images can be found in the Appendix B repository.

The full reference evaluation results can be seen in Table 6.2. The evaluation processed five distortion classes independently, SSIM values are averaged over 250 samples and 2000 samples, VIF values over 2000 samples.

`Full Reference Kadid Dataset Evaluation`					
Method	Excellent	Good	Fair	Poor	Bad
SSIM (small sample)	0,966072	0,943522	0,904775	0,827636	0,703504
SSIM (large sample)	0,970115	0,940862	0,897821	0,813220	0,722375
VIF	0,847774	0,725568	0,533829	0,308312	0,152306

Table 6.2: Evaluation of the generated dataset using FR methods.

6.3.2 CNCF training and validation

The main training of the CNCF network was done in MATLAB's DeepNetworkDesigner with 30 epochs containing 28 110 iterations. Frequency of validation was every 500 itera-

tions and the whole training process took 7 hours and 31 minutes. The training set of 40 000 images was split 3:1 for training:validation. The reached accuracy was 70,37%. The process of learning is detailed in the Figure 6.3.

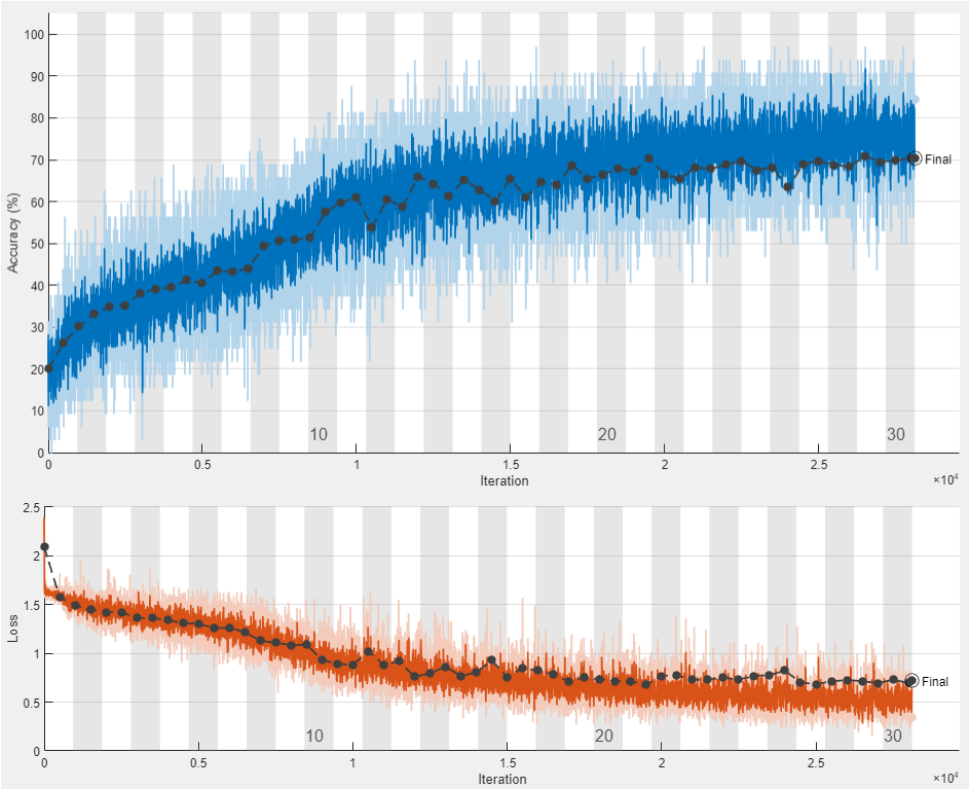

Figure 6.3: Training performance of the CNCF on 40 000 images with 4 different distortions (Gaussian blur, JPEG and JP2K compression and color noise) in 5 levels.

You can see that the validation curve traces the training curve quite well over the first 15 epochs. Then the network starts to get little overfitted, with validation curve being little less steep compared to the training curve. The dispersion of the training curve is rather high, which can be caused by distortions of lower level being harder to identify.

Experimental testing of the network with smaller amount of 250 images per category (randomly chosen from the full 40 000 dataset) unfortunately resulted in overfitting, as the training accuracy after 27 epochs reached around 90% but the validation accuracy stagnated around 20%, which for 5 outputs mean almost pure guessing. See the graph of training progress in Figure 6.4.

6.3.3 CNCF testing

After training the CNCF network, I tested it on other databases, namely LIVE, Kadid 10k (different images than Kadid 700k but the procedure of obtaining the distorted one is the same) and CSIQ. The `classify` function gives output corresponding to the settings

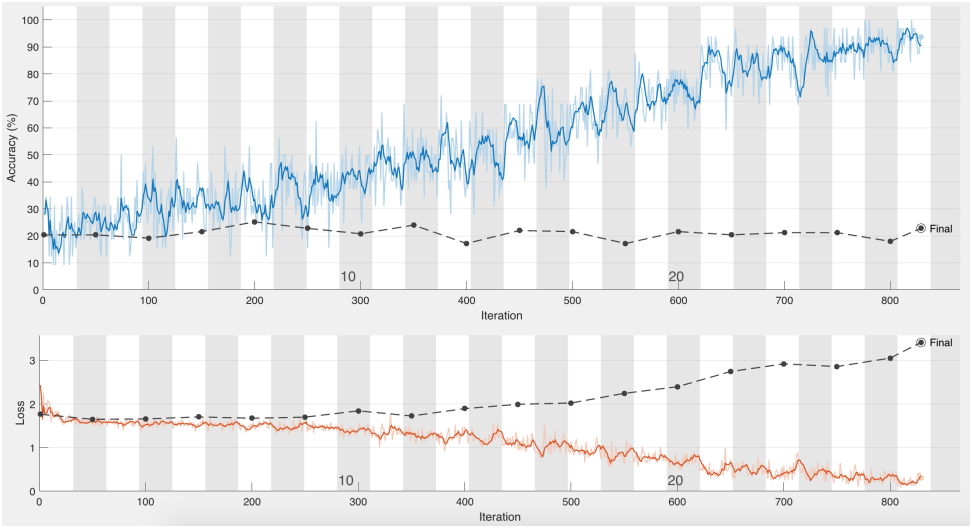

Figure 6.4: Graph of training and validating of the CNCF network on 1250 images of Kadid 700k dataset.

of the neural network output, in this case number from 1 to 5, with 1 being excellent quality and 5 bad quality. The accuracy of these classifications can be seen in Table 6.3.

Accuracy of CNCF on Image datasets	
Name and dist. type	Accuracy
CSIQ awgn	0,667%
CSIQ jpeg	0,5%
CSIQ jp2k	0,567%
CSIQ fnoise	0,7%
CSIQ blur	0,487%
Kadid 10k	0,707%

Table 6.3: Accuracy of trained CNCF on CSIQ and Kadid 10k databases.

The results correspond with the validation accuracy in best cases, having more difficult time mainly with blur determination and compression. This is understandable to the nature of the distortions, with noise being visible even in small amounts and disturbs the image subjectively more than compression or slight blur.

Chapter 7

Conclusion and future work

The research in image quality assessment has come a long way in the last decade. No reference solutions accomplished by the neural networks perform generally better than approaches based on other learning approaches, such as natural scene statistics. While the foundations of the most architectures are based on older models of neural network, mostly AlexNet [1] and VGG16 [37], the proposed networks usually shows superior performance due to branching streams of the network, preprocessing images, adding another image classification tools of using architectures trained on extensive image recognition datasets. The main challenge still seems to be lack of evaluated images to train on, which is solved by augmenting the datasets by turning the images into patches or rotating them. The newest proposed image database, Kadid 700k [15] is much larger than any other database, however it's images are not evaluated by human observers but rather by assigning the opinion score based on empirical testing of the distortion generator algorithm.

The proposed network CNCF is trained on part of the generated Kadid 700k set, using only 4 distortion types. It's architecture is inspired by AlexNet and the proposed approaches, branching the stream to learn additional features and classifying image into 5 classes. It's performance, however is not optimal, showing accuracy of 70,37%, due to many factors. Fine tuning of the hyperparameters consumes huge amounts of time and is usually more experimental rather than deterministic. Training of the network is extremely computationally- and time-demanding, requiring processing of many thousands of images in many epochs to reach satisfactory results. The network can now recognize quality of image when distorted by one of four types of distortion, but fails when other distortion is present. In the future work, the network can be trained on more types of image distortion to become more robust. Using different datasets or their combination for training may lead to increased accuracy of classification.

Bibliography

[1] I. Sutskever A. Krizhevsky and G. E. Hinton. "ImageNet Classification with Deep convolutional Neural Networks". In: *Communications of the ACM* (2010).

[2] T. Chen B. Xi N. Wang and M. Li. "Empirical Evaluation of Rectified Activations in Convolutional Network". In: *Cornell University* (2015).

[3] B. Bare B. Yan and W. Tan. "Naturalness-Aware Deep No-Reference Image Quality Assessment". In: *IEEE Transactions on Multimedia* (2019), pp. 2603–2615.

[4] A. Bovik. *Handbook of Image Video Processing*. Elsevier Academic Press, 2005.

[5] M. Buczkowski and Ryszard Stasiński. "Convolutional Neural Network-Based Image Distortion Classification". In: *IEEE on Image Processing* (2019).

[6] Matteo Carandini, David Heeger, and J Movshon. "Linearity and Normalization in Simple Cells of the Macaque Primary Visual Cortex". In: *J Neurosci* 17 (Oct. 2004). DOI: 10.1523/JNEUROSCI.17-21-08621.1997.

[7] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[8] Microsoft Corporation. *ML.NET - An open source and cross-platform machine learning framework*. URL: `https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet`.

[9] T. Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *MM '14: Proceedings of the 22nd ACM international conference on Multimedia* (2014), pp. 675–678.

[10] Robert Hirsch. *Exploring colour photography*. Laurence King, 2005.

[11] A.-D. Nguyen J. Kim and S. Lee. "Deep CNN-Based Blind Image Quality Predictor". In: *IEEE Transactions on neural Networks and Image Processing* (2019), pp. 11–24.

[12] S. Ren K. He X. Zhang and J. Sun. "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[13] L. Kang et al. "Convolutional Neural Networks for No-Reference Image Quality Assessment". In: (2014), pp. 1733–1740.

[14] J. Kim et al. “Multiple Level Feature-Based Universal Blind Image Quality Assessment Model”. In: (2018), pp. 291–295.

[15] Hanhe Lin, Vlad Hosu, and Dietmar Saupe. “KADID-10k: A Large-scale Artificially Distorted IQA Database”. In: *2019 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2019, pp. 1–3.

[16] Tsung-Jung Liu et al. “Visual quality assessment: recent developments, coding applications and future trends”. In: *APSIPA Transactions on Signal and Information Processing* 2 (2013), e4. DOI: 10.1017/ATSIP.2013.5.

[17] K. Ma et al. “End-to-End Blind Image Quality Assessment Using Deep Neural Networks”. In: *IEEE Transactions on Image Processing* 27.3 (2018), pp. 1202–1213.

[18] Kede Ma et al. “Waterloo Exploration Database: New Challenges for Image Quality Assessment Models”. In: *IEEE Transactions on Image Processing* 26.2 (Feb. 2017), pp. 1004–1016.

[19] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[20] A. Mittal, A. K. Moorthy, and A. C. Bovik. “No-Reference Image Quality Assessment in the Spatial Domain”. In: *IEEE Transactions on Image Processing* 21.12 (2012), pp. 4695–4708.

[21] A. K. Moorthy and A. C. Bovik. “Blind Image Quality Assessment: From Natural Scene Statistics to Perceptual Quality”. In: *IEEE Transactions on Image Processing* 20.12 (2011), pp. 3350–3364.

[22] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *ICML’10 Proceedings of the 27th International Conference on International Conference on Machine Learning* (2010).

[23] M. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[24] C. Pan. “Blind Image Quality Assessment via Vector Regression and Object Oriented Pooling”. In: *IEEE transactions on Multimedia* (2018), pp. 1140–1153.

[25] D. Gong Q. Yan and Y. Zhang. “Two-Stream Convolutional Networks for Blind Image Quality Assessment,” in: *IEEE Transactions on Image Processing* (2019), pp. 2200–2212.

[26] G. Qiu. “Direct Application of Convolutional Neural Network Features to Image Quality Assessment”. In: *IEEE on Image Processing* (2018).

[27] S. Winkler R. C. Streijl and D. S. Hands. “Mean Opinion Score (MOS) revisited: Methods and applications, limitations and alternatives”. In: *Multimedia Systems* 22 (2016), pp. 213–227.

[28] M. T. Abed S. Albawi and S. Al-zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)* (2017).

[29] R. Schettini S. Bianco L. Celona and P. Napoletano. "On the use of deep learning for blind image quality assessment". In: *Signal, Image and Video Processing* (2017).

[30] M. A. Saad, A. C. Bovik, and C. Charrier. "Blind Image Quality Assessment: A Natural Scene Statistics Approach in the DCT Domain". In: *IEEE Transactions on Image Processing* 21.8 (2012), pp. 3339–3352.

[31] R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research 15* (2014), pp. 1929–1958.

[32] H. R. Sheikh and A. C. Bovik. "Image information and visual quality". In: *IEEE Transactions on Image Processing* 15.2 (2006), pp. 430–444.

[33] H. R. Sheikh, M. F. Sabir, and A. C. Bovik. "A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms". In: *IEEE Transactions on Image Processing* 15.11 (2006), pp. 3440–3451.

[34] G. Shi. "End-to-End Image Quality Assessment with Cascaded Deep Features". In: *IEEE International Conference on Multimedia and Expo (IMCE)* (2019).

[35] E. P. Simoncelli. ""Mean squared error: love it or leave it? - a new look at signal fidelity measures". In: *IEEE Signal Processing Magazine* 26 (Jan. 2009), pp. 98–117.

[36] E. P. Simoncelli and W. T. Freeman. "The steerable pyramid: a flexible architecture for multi-scale derivative computation". In: 3 (1995), 444–447 vol.3.

[37] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-scale Image Recognition". In: *Computer Vision and Pattern Recognition* (2015).

[38] Christian Szegedy et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. arXiv: `1602.07261 [cs.CV]`.

[39] D. Weimer, Ariandy Benggolo, and Michael Freitag. "Context-aware Deep Convolutional Neural Networks for Industrial Inspection". In: Dec. 2015.

[40] Z. Guoqiang Y. Yuan C. Zhenwei and G. Yudong. "Color Image Quality Assessment with Multi Deep Convolutional Networks". In: *IEEE 4th International Conference on Signal and Image Processing* (2019).

[41] P. Ye et al. "Unsupervised feature learning framework for no-reference image quality assessment". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1098–1105.

[42] L. Lu Z. Wang and A. Bovik. "Video Quality assessment based on structural distortion measurement". In: *Signal Processing: Image Communication* 19 (2004), pp. 121–132.

[43] Eero P. Simoncelli Zhou Wang and A. C. Bovik. “Multi-Scale Structural Similarity For Image Quality Assessment”. In: *Proceedings of the 37th IEEE Asomilar Conference on Signals, Systems and Computers, Pacific Groove, CA* (Nov. 2003).

Appendix A

External links

ImageNet:	http://www.image-net.org/
CIFAR databases:	https://www.cs.toronto.edu/~kriz/cifar.html
MNIST database:	http://yann.lecun.com/exdb/mnist/
TID2013 database:	http://www.ponomarenko.info/tid2013.htm
LIVE database:	https://live.ece.utexas.edu/research/quality/subjective.htm
CSIQ database:	http://vision.eng.shizuoka.ac.jp/mod/page/view.php?id=23
KADID databases:	http://database.mmsp-kn.de/kadid-10k-database.html#
Waterloo Exploration database:	https://ece.uwaterloo.ca/∼k29ma/exploration/
LIVE Image Quality Assessment algorithms:	https://live.ece.utexas.edu/research/Quality/index_algorithms.htm

Appendix B

Structure of appendix archive

The Appendix B consists of two folders:

- **\matlabScripts** containing models of CNCF network in untrained version, pre-trained on Gaussian blur images and pre-trained on full set of 40 000 images from Kadid 700k. Script named *Evaluation* contains simple piece of code used to to classify images using the CNCF network.
- **\sampleImages** contains 12 sample images from the Kadid 700k database, with 4 different types of distortion in the highest degradation level.

Due to the size limitations the archive is split into 3 pieces, namely *Appendix B.zip*, *Appendix B.z01* and *Appendix B.z02*. The files are to be extracted from the .zip file.