PRE-TRAINED CNN AND BI-DIRECTIONAL LSTM FOR NO-REFERENCE VIDEO QUALITY ASSESSMENT

Doreen Duoduaah

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science in Electrical and
Computer Engineering
Concordia University
Montreal, Quebec, Canada

June 2022

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Doreen Duoduaah**

Entitled: **Pre-trained CNN and Bi-directional LSTM for Video Quality Assessment**

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

Master of Applied Science in Electrical and Computer Engineering

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

Signed by the final examining committee:

Dr. William E. Lynch ——————————— Internal Examiner (Chair)

Dr. Abdessamad Ben Hamza ——————————— External Examiner

Dr. Maria A. Amer ——————————— Supervisor

Approved by ———————————————————————

Dr Jun Cai, Graduate Program Director

July 08, 2022 ———————————————

Dr. Mourad Debbabi, Dean

Gina Cody School of Engineering and Computer Science

Abstract

Pre-trained CNN and Bi-directional LSTM for No-Reference Video Quality Assessment

Doreen Duoduaah

A challenge in objective no-reference video quality assessment (VQA) research is incorporating memory effects and long-term dependencies observed in subjective VQA studies. To address this challenge, we propose to use a stack of six bi-directional Long-Short Term Memory (LSTM) layers of different units to model temporal characteristics of video sequences. We feed this bi-directional LSTM network with spatial features extracted from video frames using pre-trained convolution neural network (CNN); we assess three pre-trained CNN, MobileNet, ResNet-50 and Inception-ResNet-V2, as feature extractors and select ResNet-50 since it showed the best performance. In this thesis, we assess the stability of our VQA method and conduct an ablation study to highlight the importance of the bi-directional LSTM layers. Furthermore, we compare the performance of the proposed method with state-of-the-art VQA methods on three publicly available datasets, KoNVid-1K, LIVE-Qualcomm, and CVD2014; these experiments, using same set of parameters, demonstrate that our method outperforms these VQA methods by a significant margin in terms of Spearman's Rank-Order Correlation Coefficient (SROCC), Pearson's Linear Correlation Coefficient (PLCC), and Root Mean Square Error (RMSE).

Acknowledgments

To begin with, my profound gratitude goes to my supervisor, Dr. Maria A. Amer for her unwavering support, attention and patience throughout my studies and her perseverance and her exceptional attention to details skills in making this possible.

I would also like to extend my appreciation to my senior colleague and friend, Mark, for his support and advice. Not forgetting my biggest support system, my dad and my friend, Millicent. I am highly indebted to you all and I appreciate your contributions towards making this a reality.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and problem statement

Rapid advances in digital media has led to a proliferation of video signals obtained through entertainment, consumer electronics, security, and communication. Video signals perceived by human end users may be subjected to different distortions, which may have been introduced during different stages such as capturing (e.g., out-of-focus, sharpness), compression (e.g., compression artifacts), transmission (i.e., transmission channel induced distortions), or storage (e.g., noise). The demand to meet end-user expectations while optimizing existing video services requires monitoring video quality at all stages.

Video quality assessment (VQA) can be broadly grouped into subjective and objective methods. Subjective VQA is the approach where human reviewers rate the quality of videos in experiments conducted under standard conditions [75]. The reviewers are trained observers, that provide quality ratings in the form of mean opinion scores (MOS) within a specific range of values, with the lowest value indicating the worst quality. Although this method is reliable, it is labour-intensive and thus, not often used, especially in real-time applications. It is worth mentioning that these quality scores provided by the human reviewers are also affected by human factors such as personality, age, gender, and culture [61].

Objective VQA involves developing mathematical models and estimators to predict the visual quality of a video signal. Objective VQA methods are relatively faster, cost-efficient, and more practical than subjective VQA. The ultimate goal of research in this area is to achieve objective VQA scores that highly correlate with subjective scores (MOS). Depending on the availability of the original and pristine video signal, objective VQA can be categorized as full-reference (FR), reduced-reference (RR), and no-reference (NR).

Full reference requires complete information on the original video signal for quality assessment. Classical methods such as Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural SIMilarity index (SSIM) [80] are good examples of FR in image quality assessment (IQA). The methods ST-MAD [78] and MOVIE index [62] are examples of FR VQA. Reduced-reference VQA methods require partial information from the original videos, for example, the RR methods in [55] and [70] are designed for coding and transmission distortions.

Although FR and RR have been proven to perform well, information on the original pristine video is unavailable in practical situations. As a result, NR methods that do not require prior information on original pristine videos are more attractive and practical, and have been adopted in recent applications [22, 16, 86]. NR methods vary from classical mathematical approaches such as [89, 12, 6, 7], TLVQM [31], V-CORNIA [81], and V-BLIINDS [58] to deep learning methods such as [77, 79], and DeepSTQ [88].

Recent advances in deep learning have led to the wide-use of deep convolutional neural networks (CNN) in standard image processing tasks such as image classifications and object detection. CNNs perform remarkably better than hand-crafted methods for visual feature extraction by learning complex features in these tasks [13, 68, 17]. However, building such powerful deep models requires a large amount of data for training. In comparison with image datasets such as ImageNet [11], MS-COCO [42], and CIFAR [32], the VQA community cannot boast of such large datasets.

Moreover, a study conducted in [36] revealed recent challenges in objective VQA including the lack of large datasets for modelling, the challenge of effectively incorporating long-term dependencies and memory effects of the human observer, and the need to provide a general-purpose VQA, that is, assessment not specialized for specific types of distortions.

1.2 Summary of contributions

The objectives of this thesis are 1) to develop a fast (e.g., using pre-trained CNN) and accurate (e.g., using all video frames) algorithm for a no-reference, general-purpose video quality assessment despite the relatively small data (videos) available to the VQA community, 2) to efficiently model the temporal features (e.g., memory effects, long-term dependencies) in videos (e.g., using LSTM). As a result, the contributions of this thesis are 1) a novel VQA method consisting of a pre-trained CNN to extract spatial features, a stack of six bi-directional LSTM layers of different units to model temporal dependencies, and two fully connected layers to learn one video quality score from the resulting spatio-temporal features; 2) comparison of three pre-trained CNN, MobileNet, ResNet-50, and Inception-ResNet-V2, to extract discriminative features for each video frame for VQA; 3) discussion of how the use of a CNN pre-trained on the large ImageNet dataset can reduce the need for a large amount of video sequences, unavailable in the VQA community, for training in order to build a robust CNN model and reduce computing resources and training time; and 4) stability and ablation studies of the proposed VQA.

1.3 Thesis Outline

In Chapter 2, we provide a summary of the concept of artificial neural networks, their types and their training. A review of related works follows in Chapter 3. Chapter 4 describes the details of the proposed method. Experiments and results on three publicly available datasets are discussed in Chapters 5 and 6, respectively. The final chapter, Chapter 7, summarizes the findings from the experiments and presents future work which concludes the thesis.

Chapter 2

Background Material to Deep Learning

2.1 Introduction

Machine learning (ML) is defined as the field of study that gives computers the ability to learn without being explicitly programmed [60]. ML can be understood as a branch of artificial intelligence that enables the development of algorithms used by computing systems to automatically learn and improve experience from data. ML applications includes spam detection, facial recognition, self-driving cars, fraud detection, robotics and many others. The algorithms used in these applications varies from statistical modelling techniques such as linear and logistic regression to more complex algorithms such as decision trees, support vector machines and artificial neural networks. Artificial neural networks mimic the processing abilities of the human brain and it is considered in most image and video processing applications.

2.2 Artificial neural networks

Artificial neural network (ANN) was developed inspired by the information processing and distributed communication nodes of the network of biological neurons in the human brain. The first architecture of ANN was introduced in [44] with the basic building block being the artificial neuron. An artificial neuron has one or more binary inputs and an output which is activated when a certain number of its inputs are active, similar to how biological neurons generates an output signal when an accumulated input signal exceeds a certain threshold. An artificial neuron can be used to compute simple logical operations (for example, AND, OR, and NOT) when one or both of its inputs are activated (i.e., on/off).

The perceptron, which is a slight variation of the artificial neuron, consists of multiple number-value inputs instead of binary (on/off) values as shown in Figure 2.1. Each input has a weight (w) associated with the connection to the neuron. A threshold (or activation) function $h(s)$ is applied to the weighted sum of its input. A perceptron can acts as a binary classifier. If the output of $h(s)$ exceeds a threshold, the function outputs the positive class, else it outputs the negative class. The output y of the perceptron is thus computed as:

$$y = h(s) = h(x_1 w_1 + x_2 w_2 + \cdot + x_n w_n + b), \tag{2.1}$$

where x_i are the inputs (also called input vector), w_i are the weights (weight vector) associated with the inputs, and b is the bias. An example of an activation function is the *unit step* function defined as:

$$step(z) = \begin{cases} 0 & \text{if z} < 0 \\ 1 & \text{if z} \geq 0 \end{cases} \tag{2.2}$$

In ANN, a perceptron is an artificial neuron using the step function as the activation function. A perceptron is also called "single layer perceptron". For a dataset of training samples and their labels (or ground truths), during training, the weights w_i are first initialized and the perceptron is fed with data and their corresponding labels. For each predicted output, the error made by the network is calculated considering the predicted value and the original label. The connection weights w_i for each neuron are then updated using these errors.

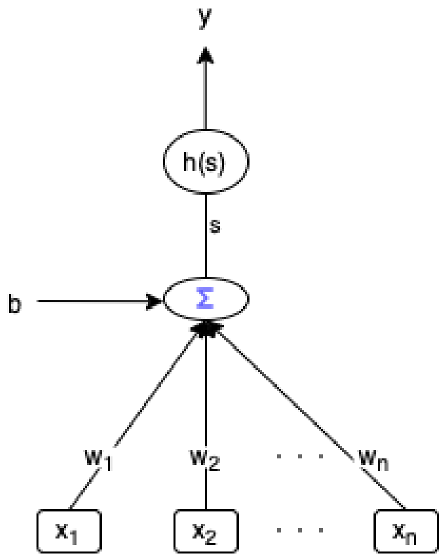

Figure 2.1: A single neuron perceptron.

The decision boundary of the perceptron is linear; consider two inputs x_1, x_2 as a point on a plane; then the perceptron decides this point belongs to which region (class) on the plane; and such classes separated by a line, are called linearly separable. Hence, this makes the perceptron suitable for linearly separable data only. To solve complex non-linear tasks, we use multi-layer perceptron. A multi-layer perceptron is a stack of single layer perceptrons. When all the neurons of a layer are connected to all the neurons of the previous layer, the layer is termed as a **fully connected or dense layer**. A fully-connected multi-layer perceptron is shown in Figure 2.2; there is the input layer, the output layer, and the layers between the output and input known as the hidden layers. An ANN with more than two hidden layers is considered a deep network.

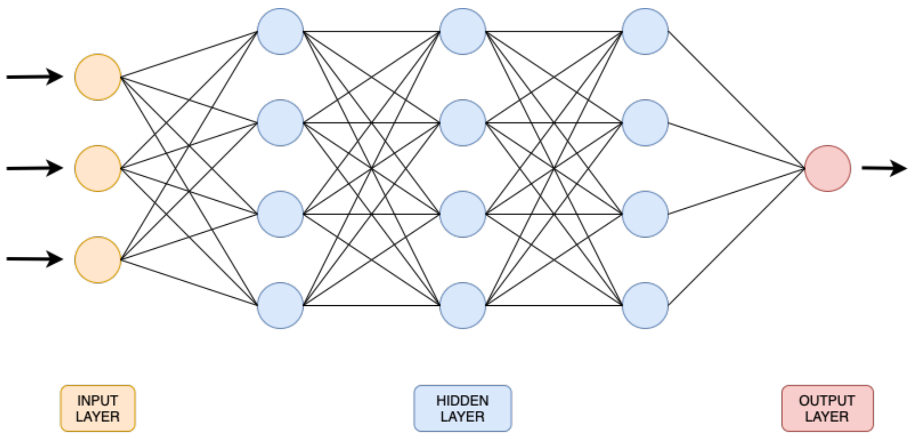

Figure 2.2: Architecture of a multi-layer perceptron (feed-forward network) with three inputs, three fully connected hidden layers each of 4 neurons and one output neuron.

2.3 ANN training (or learning)

Learning involves the adjustment of the weights in an ANN to produce the desired results. An optimization algorithm carries out the learning process. Stochatic gradient descent (SGD) based optimization algorithms are commonly used in training ANN. These include Adaptive Moment Estimation (Adam) optimizer [30], Adagrad [15], or Nesterov accelerated gradient (NAG) [47]. A cost (or loss) function is defined to evaluate the error between the predicted output and the ground truth. The loss function $E(\cdot)$ depends on the network architecture, activation function, and the type of error chosen for the training. Common loss functions are Mean Squared Error between estimated real-valued output (in regression tasks) and ground-truth and Cross-entropy of binary output (in classification tasks) using the probability by which the network detects the existence of a class. The aim of the optimization algorithm such as SGD is to minimize the loss function of a predictive model with regard to a training dataset. For each forward pass through the ANN, the loss function is computed using the predicted outputs and the ground truths. Back-propagation, a differentiation algorithm introduced in [57] is then used to compute the gradients of the parameters (i.e., weights and biases) in the network. This gives the contribution of each parameter to the prediction error. The optimization algorithm then adjusts the model's parameters using the computed gradients. The parameters are repeatedly adjusted using back-propagation and the optimization algorithm (such as SGD) during training of the ANN until the prediction error measured by the loss function is considerably low. Using the gradient descent, each of the model weights at each ANN layer is updated at each iteration i (that is, a complete forward and backward pass of the ANN) as in:

$$w_i = w_{i-1} + \Delta w_{i-1} = w_{i-1} - \alpha \frac{\partial E}{\partial w_{i-1}}. \tag{2.3}$$

Subtracting the gradient of the loss function from the weights helps update the weights so that the loss decreases in each iteration with the learning rate α. For each iteration, the learning rate which is a tunable (or optimizable) parameter (called hyper-parameter) determines the step size at which the optimization algorithms "moves" to minimize the loss function. Typical values of the learning rate range from 0.000001 - 0.1.

During training, a dataset can be divided into one or more batches. A batch may include one sample, a set of samples (called mini-batch, typically 32, 64, and 128), or all training samples. The batch size is the number of training samples to use before the model's internal parameters are updated. A complete pass of the ANN on a batch size is termed as an iteration. The number of epochs is the number of times the optimization algorithm sees the complete dataset. One epoch means that each sample in the training dataset was used once to update the internal model parameters. Typical values for epochs are 10, 100, 500, or higher. For example, if there are 1000 training samples and the batch size is 500, then it will take 2 iterations to complete 1 epoch. When the number of epochs is more than necessary, the ANN learns patterns that are specific to the training dataset which makes the network incapable to perform well on a new similar dataset (i.e., the network is overfitting). Therefore, to minimize overfitting, a good approach is to train the ANN for an optimal number of epochs.

The activation function used in a single layer perceptron is the unit step function, which is not differentiable. This means that, there is no gradient to work with when using the back propagation algorithm. Hence, this is replaced in a multi-layer perceptron with non-linear and differentiable activation functions such as logistic (sigmoid) [23], tanh [28], and ReLU [46] to enable the back-propagation algorithm to work effectively.

2.4 Types of ANN

The task, for which an ANN is designed to solve, is a dominant factor in the architecture design of the network. The two main tasks are classification and regression. Classification networks output a label which is chosen among a limited set of possible outcomes (classes). Object detection is an example, where a limited number of object classes exists; therefore, an object is assumed to belong to one of these classes. Classification problems with two possible classes (or labels) are called binary classification tasks. Regression networks predict a real quantity (usually a continuous value). An example of the regression task is image or video quality assessment where the input is an image or a video and the objective is to estimate its quality value. The quality is a continuous value, i.e., it is not chosen among a limited set of possible outcomes and can take any value in the valid range.

There are different types of ANNs. These include feed-forward neural network (for example, multi-layer perceptron in Figure 2.2), modular neural networks, convolutional neural network, radial-basis function neural network, and recurrent neural networks. Commonly used ones are the feed-forward, convolutional, and recurrent neural networks. A description of these is as follows.

2.4.1 Feed-forward neural networks

In a feed-forward network (see Figure 2.2), perceptrons are arranged in layers, where the first layer receives inputs and the last layer produces outputs. The middle layers are called hidden layers since they have no connection with the "external world". The more hidden layers the deeper the network is. Each perceptron in one layer is connected forward, that is, to every perceptron on the next layer. This means that data is only "fed forward" from one layer to the next. A perceptron has no connection with another layer in the same layer, this means that perceptrons are independent of each other in the hidden layer. By definition, a single perceptron is a linear binary classifier (that is, classifies input points into two classes that are linearly separable). A feed-forward network can classify a point into regions that are not linearly separable. By varying the number of perceptrons in the hidden layer, the number of layers, and the number of input and output perceptrons, a feed-forward network can classify points into an arbitrary number of classes. In supervised learning, pairs of inputs and their labels are fed into a feed-forward network for many cycles, so that the network 'learns' the relationship between these inputs and labels (called training samples).

2.4.2 Convolutional neural networks

Convolution neural networks (CNN) are designed for 2-D data (for example, images) and are usually used in image processing applications such as object detection and instant segmentation. CNNs are neural networks in which the hidden layers include several convolutional and pooling layers to extract deep features in addition to regular feed-forward fully connected layers for classification.

Convolutional layer

A convolutional layer performs the convolution operation on 2-D inputs (usually images) by applying a filter to extract the important information in the input. The features extracted are known as the feature map. The filter which is usually smaller in size than the image is systematically applied by sliding the filter starting from the top left corner of the image to the bottom left to produce the feature map. The feature map then contains the salient features of the image. A convolutional layer typically has multiple trainable filters where each filter outputs a single feature map. Typical sizes of filters are 3×3 and 5×5. The filter coefficients are optimized during training of the CNN to output application specific features of the input image.

Pooling layer

The pooling layer subsamples the feature maps from the convolutional layer using a defined window size. There are no trainable parameters in the pooling layer, hence, no learning takes place. There are different types of pooling. These include max-pooling, average-pooling and min-pooling. The common type of pooling used is the max-pooling. A simple 2×2 non-overlapping max pooling is shown in Figure 2.3.

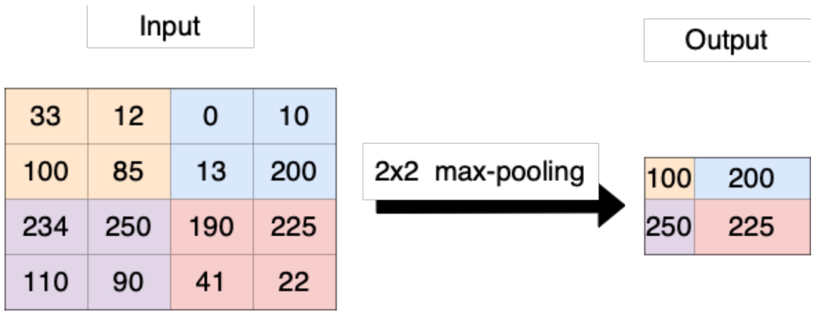

Figure 2.3: 2×2 max-pooling on an input.

Convolution and pooling layers in CNN are needed to reduce the computational load and memory requirement while successfully capturing the important spatial features of an image relevant for an application. Putting everything together, a CNN architecture includes a stack of convolutional layers interspersed with pooling layers and then fully connnected layers at the end. The architecture of a CNN can be simple as shown in Figure 2.4 or more complex, for example ResNet-50 [24] shown in Figure 2.5. A CNN learns from the input data by optimizing the weights in the filters for the convolutional layers and the weights for the neurons in the fully connected layers. The network uses a hierarchical learning approach where low features (such as edges, lines, and curves) are built on to form composite high level features such as objects and events (i.e., cars, eyes, face).

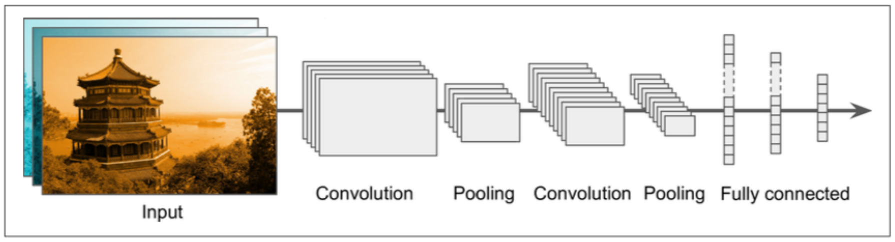

Figure 2.4: A simple CNN architecture (Figure taken from [18]).

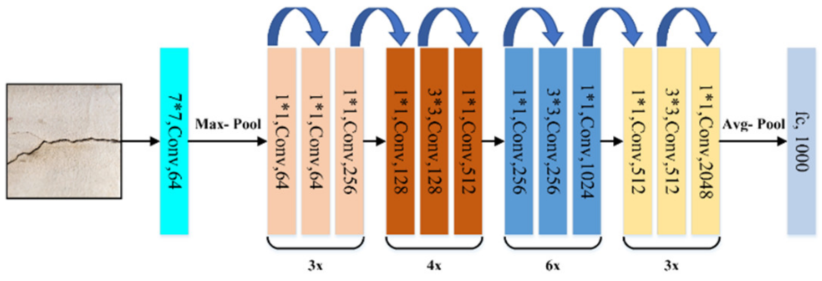

Figure 2.5: Architecture of ResNet-50 CNN (Figure taken from [3]).

2.4.3 Recurrent neural network

Recurrent neural networks (RNN) are a class of artificial neural networks specialized in processing sequential data such as time-series, speech, audio, text, DNA, and video signals. The elements of a sequence are not independent of each other and there exist an order in the sequence. RNNs are designed to process each element considering information from past elements, thus RNNs possess memory capabilities. The architecture of a simple recurrent neuron is similar to a feed-forward neuron except it has connections pointing backwards as shown in Figure 2.6. The connections enables the network to discover the temporal relations between the elements in the sequence. Figure 2.7 contains a single recurrent neuron (left) and the unrolled version (right). The unrolled neuron is basically a single neuron which has been repeated for each time step against the time axis. The part of the recurrent neuron that preserves some state across the time steps is called the memory cell. The memory cell's state at time step t, denoted h_t is a function of the inputs at that time step, x_t and the previous time step's state, h_{t-1}. Note that the h_i are the feedbacks. Here, x_0, x_1, x_2 are the time steps inputs and y_0, y_1, y_2 are outputs for the time steps.

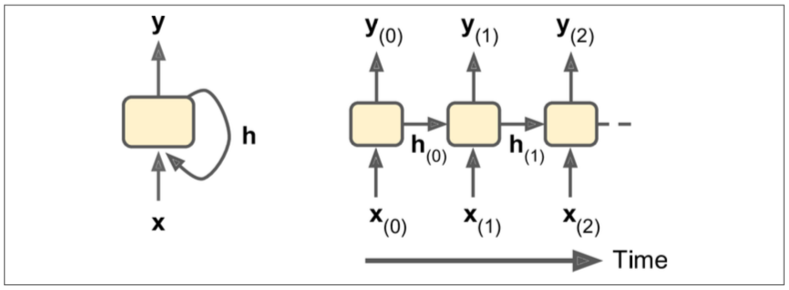

Figure 2.6: A recurrent neuron (left) and its unrolled through-time version (right) (Figure taken from [18]).

Similar to feed forward neural networks, multiple recurrent neurons can be stacked to form layers and then a network of layers as shown in Figure 2.7. The hidden layers in RNN receive their input from the input layer as well as the output from previous time step. The network has two sets of weights, one for the inputs and one for the output for previous time step. During back propagation, the RNN encounters the vanishing gradients (where the gradients exponentially goes to zero fast) and exploding gradients (where the gradients exponentially goes to infinity fast) problems [53]. In addition some information is lost at each time step due to the transformations the data go through. Hence in processing long sequences (typically more than 10 steps long), the RNN eventually ends up with little information of the first inputs. These challenges inspired the development of the long short-term memory (LSTM) which was initially proposed in [25] and improved in [59].

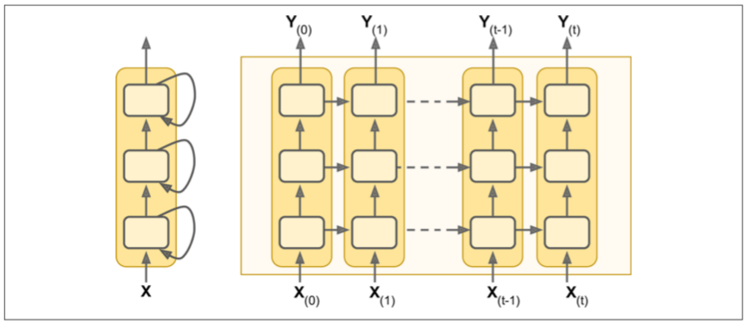

Figure 2.7: A deep RNN network (Figure taken from [18]).

2.4.4 Long short-term memory cells

The architecture of an LSTM network is like the regular RNN shown in Figure 2.7. But the LSTM has unique formulation that helps to prevent the vanishing and exploding gradients and other challenges during training. What makes this possible is the content of the memory cell of the LSTM. For an LSTM memory cell, it's state is split into:

- a long-term state c_t : memory state which traverses across the network for all the time steps,
- a short-term state h_t : the cell's output for the time step.

In addition, there exist the computational units:

- "input" gate: determines which information from the input to update the long-term state.
- "forget" gate: determines which information from the long-term state to discard.
- "output" gate: determines what to output using information from the input and the long-term state.

The gate units provide continuous analogues of "write", "reset" and "read" operations for the memory cell.

How the memory cell of an LSTM function? The long-term state c_t, traverses across all the time steps in the network and is updated using the forget and input gates at each time step. For an input at time step t, x_t, the input gate determines which information from the input x_t to add to c_t. The forget gate decides which memories in c_t to drop before it is forwarded to the next time step memory cell. The short-term state, h_t which is also the cell's output for the time step is obtained by filtering the long-term state using the output gate.

In essence, for each time step, there are two outputs: h_t and c_t. In the next time step, these outputs are labelled as h_{t-1} and c_{t-1}, respectively. The memory cell for this time step uses these outputs and it's input x_t to compute the corresponding outputs (h_t and c_t). This is repeated for all time steps and the cycle continues.

The long-term state c_t, short-term state h_t, input gate, forget gate, and output gate are all computations which have weight parameters. These weights are optimized during training using back propagation through time (BPTT) to extract the most potent temporal features at each time step from a sequence.

Also, the gate units and the consistent data flow called the constant error carrousel (CEC) keep each cell stable from the vanishing or exploding gradients problem [25]. Using the long-term state which passes through all the time steps, the LSTM is able to identify an important memory in the sequence (added by the input gate) and stores it in the long-term state for as long as it is needed. In summary, the two key advantages of LSTMs over regular RNN are:

1. Overcomes the vanishing and exploding gradient challenge in training RNN.

2. Has memory capabilities to handle the long-term temporal dependency in long sequences.

The capabilities of the LSTM has made it very successful at capturing temporal long-term relations in long sequential data and it is used in a range of sequence prediction tasks such as machine language translations, video tagging, speech recognition, and sentiment analysis. In this thesis, we show that LSTM can be perfectly used for video quality assessment.

Chapter 3

Related Works

This chapter provides a review of existing literature in NR VQA. A typical NR VQA method begins with features extracted from video frames. These features are then regressed via Support Vector Regressors (SVR), Random Forests or simply aggregated to obtain a single quality score for a video. The technique for feature extraction can be grouped into traditional methods and deep learning methods. We provide an overview of both approaches in the following.

3.1 Traditional methods

Traditional VQA methods use hand-crafted features obtained by assessing characteristics of videos such as gradients [43], natural video statistics [58], motion characteristics, saliency [6], and colorfulness [1]. V-CORNIA [81] uses local descriptors obtained through the intensity characteristics of video frames. These descriptors per frame are regressed via an SVR to obtain the frame-level quality scores. Similarly, V-BLIINDS [58] extracts features in the discrete cosine transforms domain and uses them to train an SVR which outputs the video quality score. A perceptual video quality metric, VMAF [41], was developed based on Visual Information Fidelity (VIF) [65] and the visual information loss method described in [39]. In TLVQM [31], low complexity features related to motion characteristics are computed for every second frame of a video. Based on the features computed, certain frames of the video are selected to compute high complexity features which are related to spatial artifacts. These complexity features are pooled to obtain a feature sequence vector for each video. The acquired feature vectors are regressed via SVR or Random Forest Regression to obtain the video quality score.

Some VQA methods are distortion specific. For example, [89] proposed a Laplacian method to measure the amount of compression distortions present in videos to determine video quality. The method in [12, 7], and [6] solely focus on H.264 compressed videos, while [82] considers transmission-specific distortions. These methods depend on information on video compression type and transmission channels which may not be available in all video streams or signals.

Many researchers have proposed different pooling techniques on frame-level quality scores to obtain one quality score for the whole video. For example, simple average pooling has been used in [41]; V-CORNIA [81] uses a hysteresis pooling method to predict video quality after obtaining the frame-wise quality scores. Some of the temporal pooling techniques in no-reference VQA have been discussed in [76]. These include statistics such as harmonic mean, geometric mean, arithmetic mean, percentile, Minkowski summation, and VQPooling proposed in [52]. Minkowski summation pooling method gives the strongest influence to the most distorted frame and it is shown in [56] to produce results with good correlation to subjective scores as well as percentile pooling, hysteresis, and VQPooling [52]. The experiments conducted in [56] revealed that the performance of a specific pooling technique is dependent on the video content.

The hand-crafted features and the pooling strategies for the methods described above have been designed based on limited knowledge of the human visual system (HVS) and video distortions, thus, reducing these methods' potency.

3.2 Deep-learning based methods

The remarkable capability of deep CNNs [13, 68, 17] to extract both low and high level features has led to significant advances in image processing tasks such as object detection [85, 87], image segmentation [50, 51, 45], image recognition [24, 69], and image classification [34, 29]. What is even more interesting is that these features can be shared across different image processing tasks.

In VQA, progress has been achieved also using CNN such as in [10] and [2]; the method [2] uses CNN in addition to handcrafted features and the method in [10] uses InceptionV3 [71] model for feature extraction.

Although CNN extracted features have proven more effective than hand-crafted features, these CNN models which are deep and complex need to be trained on a large dataset to obtain high performance and avoid overfitting. Unfortunately, compared to IQA, the VQA community does not currently have such large datasets, hence, the need for transfer learning. Transfer learning is a deep learning technique where pre-trained models typically trained on large, standard, benchmark datasets for a task are reused in other models for similar tasks. Common pre-trained models used are ResNet-50 [24], VGG-16 [69], AlexNet [33], and InceptionV3 [73] which are all trained on the ImageNet dataset [11]. In particular, VQA methods [77, 88, 79] use transfer learning and have demonstrated good performance on VQA datasets. These transfer learning based methods leverage knowledge from existing CNNs by using the weights obtained from training the CNN on large datasets as initializers for tuning their model parameters or by using the pre-trained CNN as feature extractors without fine-tuning. This leads to lower training time and low generalization error. In [77], Inception-V3 [73] and Inception-ResNet-V2 [72] are fine-tuned to obtain frame-level feature vectors. They carried out experiments using min, average, median and max-pooling of the frame-level feature vectors to obtain a video-level feature vector. The video-level feature vectors are trained using an SVR model for video quality prediction. Inception-V3 [73] demonstrated superior performance over Inception-ResNet-V2 [72]. The method in [79] obtains first, spatial frame level features using fine-tuned AlexNet [33] and then computes the motion information of video frames using the algorithm presented in [38]. The combined spatial and motion features are regressed using a radial basis function (RBF) kernel and a linear kernel based regression models to predict video quality. The method DeepSTQ [88] uses ResNet-50 [24] to extract features from generated video frames and frame difference maps without fine-tuning or training from scratch. The authors suggest that the frame difference maps represent the motion information and hence the temporal characteristics of videos. As a result, they regress the spatial-temporal features obtained using an SVR to predict the perceptual video quality.

3.3 Most related work and our contributions

In this section, we review VQA methods, [83, 40, 84, 35], most related (that is, use CNN for spatial features extraction and RNN/LSTM for temporal feature modelling) to our approach and discuss significance of our contributions.

One of the challenges in VQA research is how to effectively model the long-term dependencies and memory effects between video frames in the temporal domain. Recurrent Neural Networks (RNN) can handle sequences of any length and capture long-term dependencies. RNNs are used to model time-series data where at every time step t, the network considers input x_t and the output from the previous time step. Thus, the network processes new information considering weighted information from outputs of previous inputs. Because of that, RNNs are said to have some form of memory, and the part of the RNN which preserves this information across previous time steps is called the memory cell [18]. However, in processing long sequences, deep RNNs suffer the standard vanishing gradients or exploding gradients problem and have a short memory [5]. The Long Short-Term Memory (LSTM) has a long-term memory and is designed to overcome the challenges of using RNNs for long sequences [20]. LSTMs can discover long-range temporal relationships and have yielded good performances in text classification [67], speech recognition [21], and handwriting recognition [20]. LSTMs are generally used in long sequences and time-series applications mainly because they perform better and converge faster during training than deep RNNs [18].

In the related work VSFA [35], the feature maps from 'res5c' layer of ResNet-50 [24] are pooled using the global average and standard deviation pooling strategy to obtain spatial feature vectors for each video frame. To model the long-term dependencies in videos and to predict the video quality score, the authors then employ a 32 gated recurrent units (GRU) layer; GRU is similar to LSTM but without the output gate and thus, has fewer parameters. Therefore, the LSTM performs better than GRUs for long and high complexity sequences [8]. In addition, the authors adopt and modify the temporal pooling approach proposed in [63] which defines a memory element for each video frame to obtain the overall video quality score. The authors train the entire network of their method in an end-to-end manner. The related method in [83] fine-tunes VGG16 [69] pre-trained CNN for feature extraction; it then employs gated recurrent units (GRU) to model the temporal characteristics. The related methods in [40] and [84] extract spatio-temporal features using 3D-CNN. However, as discussed in [14], 3D-CNNs are limited in capturing discriminative features in the temporal domain. Consequently, in addition to 3D-CNN, one 32-unit LSTM layer is used in [84] to extract temporal features. [40] did not specify the number of LSTM layers used. A peculiar characteristic common to the methods [83],[40], [84], and [35] is that, more attention is given to the spatial feature extraction stage where most of their method's complexities lie. Specifically, they use CNN with multiple convolution and pooling layers then a single layer of RNN units to regress the features learned to a video quality score with less emphasis on the long-term dependencies in video frames. In addition, CNN training on video data is computationally intensive. As a result, these methods perform a partial selection of video frames during training: [83] samples the video frames at 1 frames/second (fps) and [84] splits each video into blocks and assume each block has the same quality as the complete video. These techniques do not guarantee that the frames selected are keyframes and thus, well represent the spatial and temporal characteristics of the full video.

Different than these most related VQA work, in the proposed method, we use a pre-trained CNN as a spatial feature extractor for *all* video frames without any fine-tuning. Using all frames eliminates the need to sample (key) video frames during training, which is not a trivial task. On the other side, in using pre-trained CNN, we minimize the need for a large amount of data (i.e., video signals) which are unavailable in the VQA community to build a robust model. In addition, there is a significant reduction in the complexity during training of our spatio-temporal deep model. That is, CNN is designed mainly for images, using CNN for video applications requires a T-fold (where T is the number of frames per video) of the regular training time, which is highly resource-intensive and time-consuming. Not re-training and fine-tuning the CNN can significantly cut down the computing resources and time needed. For this, we assessed MobileNet [27], ResNet-50 [24], and InceptionResNetV2 [72] as the spatial feature extractor and propose to use ResNet-50 [24] because it showed the best performance.

While related work [83, 40, 84, 35] pay more attention to spatial feature extraction than temporal features modelling, we aim to balance both. Given the spatial features extracted, we thus employ bi-directional LSTM units, which capture relations in sequences in both the forward and backward directions to model the memory effects and long-term dependencies between video frames. We propose bi-directional

LSTM network consisting of four 128-unit and two 64-unit bi-directional LSTM layers with simple concatenation as the merging strategy for the forward and backward processes.

Chapter 4

Proposed Method

The propose NR VQA consisting of two modules is depicted in Figure 4.1. Details of the non-trainable and trainable modules are described in the following sections. In summary, we use existing pre-trained CNN as a feature extractor for VQA, where we keep all layers preceding the top-most layer (that is, the one for classification) and apply max-pooling to the last convolutional layer. We feed this slightly modified pre-trained CNN with individual video frames and obtain an n-dimensional vector representing the spatial features (for example, $n = 2048$ for ResNet-50 [24]). We then feed this sequence of feature vectors to a bi-directional LSTM network in order to model the temporal characteristics between video frames in both the forward and backward direction. The architecture of the bi-directional LSTM model was obtained by using keras tuner [49] to tune the number of layers and units over a range of values. We found using a stack of four 128-unit and two 64-unit bi-directional LSTM layers to give the best results. The combined spatial and temporal features obtained are pooled via a 32-unit and then a 1-unit fully-connected layers to output a single video quality score.

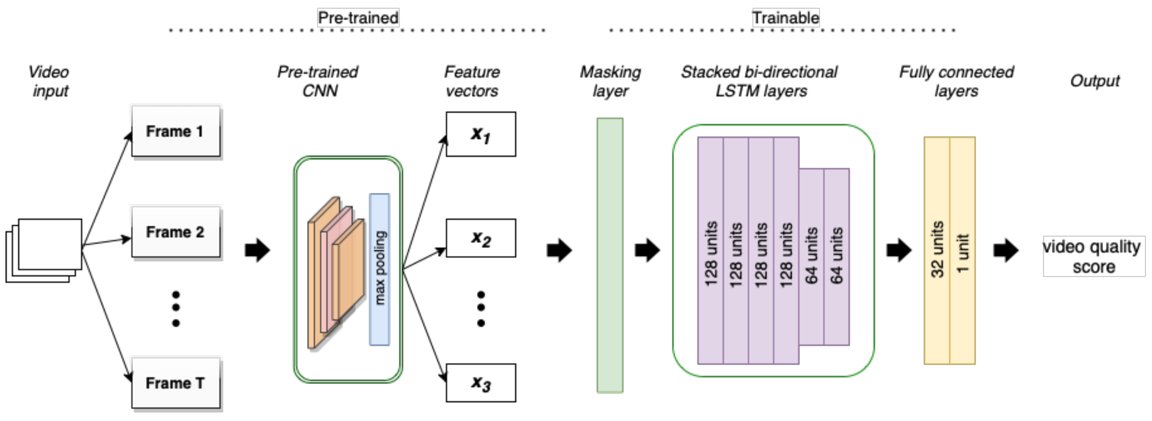

Figure 4.1: The framework of the proposed VQA method. (See Figure 4.2 for a block diagram of an LSTM unit).

4.1 Pre-trained module

Many deep CNN models have been made available along with their pre-trained weights for feature extraction, fine-tuning or prediction. We use the pre-trained network merely for spatial feature extraction from video frames without fine-tuning. Based on results from the experiments we conducted (see Section 5.2), we select ResNet-50 [24] as the CNN model for feature extraction. It is a network with 50 layers grouped into 5 stages. All, with the exception of the first stage, consist of blocks of convolutional layers. The network ends with a global average pooling and a 1000-unit fully connected layer as the top-most layer for prediction. We remove this 1000-unit fully connected layer and apply max-pooling to the last convolutional layer instead of the average pooling. We chose max-pooling because it has the best performance in [79, 10]. With this slight modification, the CNN results in a feature vector that we use as an input to the LSTM network. For a video V with T frames, i.e.,

$$V = \{I_1, I_2, I_3, \cdot\cdot\cdot, I_T\}, \tag{4.1}$$

we pre-process each video frame I_t according to the specifications provided by the authors of ResNet-50 [24]: the frame is converted from RGB to BGR, each colour channel is zero-centred with respect to the ImageNet dataset [11], and resized to $224 \times 224 \times 3$ without scaling. We then feed the pre-processed frame to the modified CNN model and the output is an $n = 2048$ feature vector, x_t, i.e.,

$$x_t = modified\_ResNet(I_t). \tag{4.2}$$

That is, we obtain a sequence $\{x_t\}$ of T feature vectors per video.

4.2 Trainable module

The trainable module consists of a stack of six bi-directional LSTM layers and two fully connected layers (see Figure 4.1). Before the LSTM network, we apply zero-padding and then masking: Due to the varying length of videos in VQA datasets, we zero-pad the shorter spatial feature vector sequence to match the size of the longest feature vector sequence (representing that video with the maximum number of video frames). This facilitate batch training of the LSTM network. Each padded sequence is then passed through a masking layer in order to skip the padded part of the sequence during batch training and testing of the LSTM network.

An LSTM layer has a memory cell for each time step, that is, for a T-frame video, there are T memory cells. Each memory cell in turn has structurally identical units. For example, the first bi-directional LSTM layer in the proposed method has 128 units for each memory cell. A unit has a long-term state c_t, a short-term state h_t, a hidden state g_t, and "input" i_t, "forget" f_t, and "output" o_t gate as shown in Figure 4.2. Note that lower-case indexed variables are vectors and upper-case indexed variables are matrices. At time step t in an LSTM unit, the "input" i_t, "forget" f_t, and "output" o_t gates, and the hidden state g_t, are computed as weighted addition of the input data x_t and the previous short-term state h_{t-1}, ($h_0 = 0$) as follows,

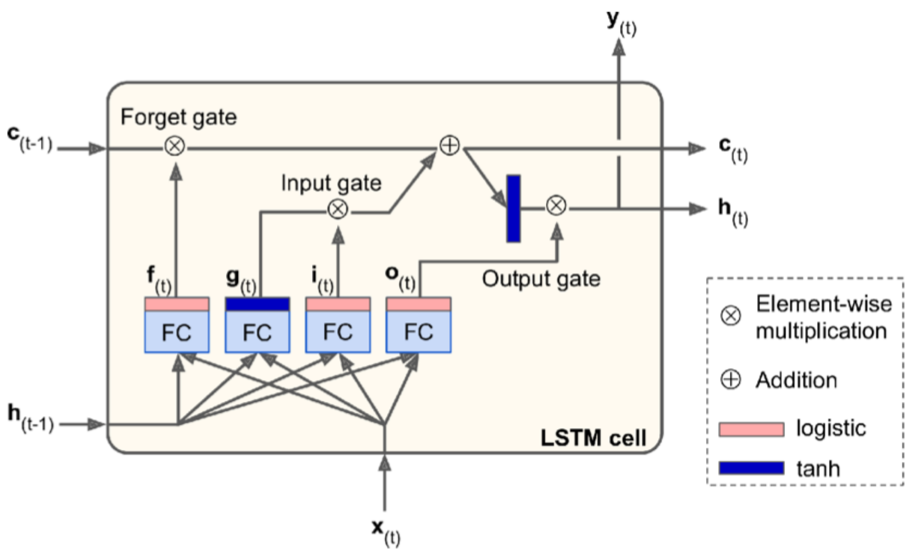

Figure 4.2: The contents of the unit of a single direction LSTM memory cell. FC refers to fully connected layer. (Figure taken from [18]).

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \\ g_t &= tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g), \end{aligned} \tag{4.3}$$

where $\sigma(.)$ is the logistic function, W_{xi}, W_{xf}, W_{xo} and W_{xg} are the weights in connection with the input data x_t, W_{hi}, W_{hf}, W_{ho} and W_{hg} are the weights in connection with the short-term state h_{t-1} from previous memory cell, and b_i, b_f, b_o, b_g are the biases. The i_t, f_t, o_t and g_t are vectors, each of size equal to the number of units in the LSTM layer (for example, in a 128-units LSTM, the size is 128). The logistic function is defined as,

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{4.4}$$

and the tanh function is defined as,

$$tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1}. \tag{4.5}$$

The three gates and the hidden state are then used to compute the long-term states c_t and the short-term states h_t. The long-term state for the forward process c_{tf}, and backward process c_{tb} (marked $c_{(t)}$ in Figure 4.2 for a single direction) are computed as in,

$$\begin{aligned} c_{tf} &= (f_t \otimes c_{t-1}) + (i_t \otimes g_t), \\ c_{tb} &= (f_t \otimes c_{t+1}) + (i_t \otimes g_t), \end{aligned} \tag{4.6}$$

where $\otimes$ is element-wise multiplication. The corresponding short-term states h_{tf} and h_{tb} (marked $h_{(t)}$ in Figure 4.2 for a single direction) which are also the unit's output y_{tf} and y_{tb} are then computed as,

$$\begin{aligned} y_{tf} &= h_{tf} = o_t \otimes tanh(c_{tf}), \\ y_{tb} &= h_{tb} = o_t \otimes tanh((c_{tb}), \end{aligned} \tag{4.7}$$

where $\otimes$ is element-wise multiplication. The outputs from both processes are then merged (concatenated) to obtain the final output y_t of the bi-directional LSTM layer, i.e.,

$$y_t = y_{tf} \oplus y_{tb}, \tag{4.8}$$

where $\oplus$ is concatenation operation.

In summary, for each time step, the units in the memory cell outputs a long-term state $c_{(t)}$ and a short-term state $h_{(t)}$ as shown in Figure 4.2. The long-term state contains information from $t = 0$ to the current time step. As the long-term state traverses through the network, at every time step, portions of the information are dropped and updated using the forget and input gate, respectively. The short-term state which is the final output at each time step is computed using the long-term state. This gives the feature descriptors of an event at time step t based on information from previous events, i.e., the long-term state $c_{(t-1)}$ and the previous short-term state $h_{(t-1)}$, and its input x_t. For more details on the structure and computations in an LSTM cell see [25].

In the following, using the optimal number of layers and units obtained after tuning the model (see Section 5.3), we describe the steps as a video sequence passes through the proposed method. For a video with T frames, we feed ResNet-50 [24] with its video frames and obtain a sequence of extracted spatial feature vectors. The length of the sequence is T and each time step in this sequence is a vector of size n (for ResNet-50 [24], $n = 2048$). We then feed the T sequence of spatial feature vectors to the bi-directional LSTM network. (As explained earlier, padding and masking are necessary when considering multiple videos with different lengths at a time). We use four 128-unit and two 64-unit bi-directional LSTM layers. The first bi-directional LSTM layer has 128 units and takes the T sequence of spatial feature vectors. For each time step, we compute y_{tf} and y_{tb} of equation 4.7 using the n spatial feature vector. This results in a spatio-temporal feature vector of size 128 for each of the forward and backward process. We then concatenate the two spatio-temporal feature vectors y_{tf}, y_{tb} as in equation 4.8 to obtain a T sequence of spatio-temporal feature vectors, where each time step is a vector of size 256. The second, third and fourth LSTM layers, in a similar manner, outputs a T sequence of spatio-temporal feature vectors with a vector size of 256 for each time step. The fifth bi-directional LSTM layer has 64 units and hence, outputs a T sequence of spatio-temporal feature vectors with of size 128 for each time step. For the last 64-unit bi-directional LSTM layer, we consider only the spatio-temporal feature vector of the last time step which is also the final short-term state (h_T) in the network. This is usually done in LSTM networks to obtain a single vector as the final output instead of a sequence. As a result, we obtain a spatio-temporal feature vector of size 64 for each of the forward and backward direction, which we then concatenate to obtain a spatio-temporal feature vector of size 128. We next feed this final 128-size spatio-temporal feature vector of the input video to a 32-unit feed-forward fully connected layer with ReLu activation, which enables the model to learn complex relations within the features. The output is a 32-size feature vector, which we finally connect to a 1-unit fully connected layer that outputs the video quality score.

Uni-directional LSTM preserves information from the past only. We use bi-directional LSTM because we aim to capture temporal features from both the past and future; also, experiments show that viewers assign the same quality score to a video if played forward or backward in time [64].

The input to the bi-directional LSTM network is a sequence of T spatial feature vectors, where each vector has a size n, and the output is a single vector of size equal to the number of units in the last LSTM layer, that is, 64. This vector contains the spatio-temporal features of the sequence.

Chapter 5

Implementation Details

In this chapter, first, we describe the performance measures and datasets selected for our experiments, then, we discuss the feature extraction model selection process and the bi-directional LSTM model training details.

5.1 Performance measures and datasets

For VQA datasets, videos are first subjectively assessed by human viewers and mean opinion scores (MOS) are collected. VQA methods developed using these datasets aim to predict video quality scores correlated with these MOS. The widely used performance measures are Spearman's Rank-Order Correlation Coefficient (SROCC) [66], Pearson's Linear Correlation Coefficient (PLCC) [4], and Root Mean Square Error (RMSE) used in [31]. PLCC and SROCC indices reflect the correlation between the objective predicted scores and the MOS, while RMSE shows the error between these scores. It is expected that a good method should have high SROCC and PLCC values and a small RMSE value.

For testing, we used videos with corresponding subjective quality scores in three datasets: Konstanz Natural Video Database (KoNViD-1k)[26], Mobile In-Capture

Video Quality Database (LIVE-Qualcomm) [19], and Camera Video Database (CVD2014) [48].

KoNViD-1k [26] dataset has 1200 content diverse real-world video sequences with authentic distortions sampled from Yahoo Flickr Creative Commons 100 Million (YFCC100m) dataset [74]. The authors do not state the exact video distortion types. The videos are encoded at three different rates: 24, 25, and 30 fps and with 12 different varying resolutions. The most common resolution among the videos is 1280 x 720 pixels. The MOS range from 1 to 5, and its distribution is depicted in Figure 5.1.

The LIVE-Qualcomm [19] dataset comprises 208 videos obtained using eight mobile devices such as Samsung galaxy, Apple iPhone, HTC, and Nokia lumia. The videos in the dataset were created to model six common distortions introduced during the capturing process. These include artifacts, color, focus, exposure, sharpness, and stabilization. The videos have a uniform resolution of 1920 x 1080 progressive (1080p). The MOS range from 16.6 to 73.64 and its distribution is shown in Figure 5.1.

CVD2014 [48] contains 234 videos obtained using 78 different capturing devices. Like LIVE-Qualcomm [19], the dataset consists of videos with complex distortions related to the video capturing process. There are two distinct video resolutions, 640 x 480 (57 videos) and 1280 x 720 (177 videos). The MOS have a wide range of -6.5 to 93.38 and their distribution is shown in Figure 5.1.

Unlike standard deep learning applications such as object detection and image classification datasets, VQA datasets are small in size and their authors do not explicitly specify the training and test sets for experiments. Also, the authors of CVD2014 [48] and KoNViD-1k [26] do not group the videos per distortion type. Consequently, video distortions in one dataset may be entirely different from those in other datasets. Therefore, for solid evaluation, during the experiments, we randomly split each dataset into a train and test set ratio of 0.8 : 0.2. We train our proposed model using the training set and evaluate the model performance on the test set.

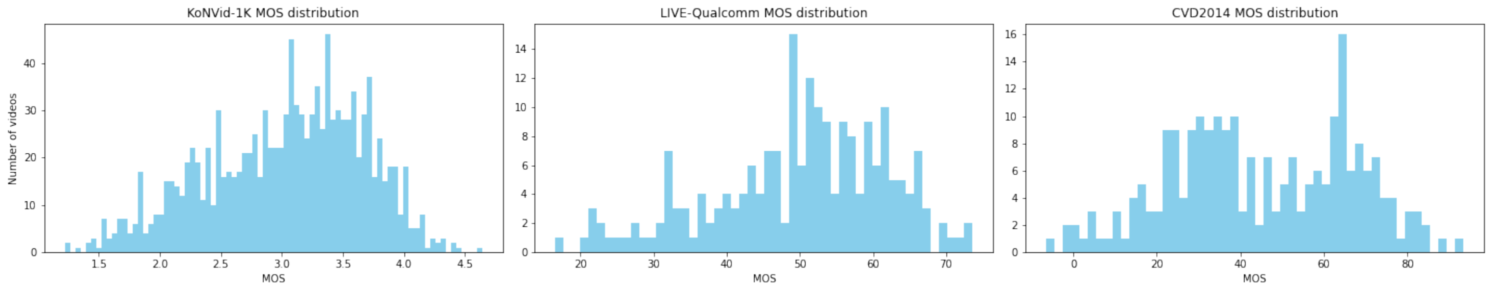

Figure 5.1: MOS distribution of KoNVid-1K, LIVE-Qualcomm and CVD2014 datasets.

5.2 Selection of the feature extractor

For feature extraction of video frames, taking inspiration from [35], [88], and [77], we use the pre-trained CNN ResNet-50 [24], Inception-ResNet-V2 [72], and MobileNet [27] trained on ImageNet [11] dataset for image classification.

MobileNet [27] has 28 layers with an input size of $224 \times 224 \times 3$. The model uses a standard convolution operation to combine filtered features from convolution kernels to form new representations. This model achieves a top-5 accuracy of 89.5% on the ImageNet [11] dataset. The model ouputs a 1024 size feature vector after applying max-pooling to the last convolution layer.

Inception-ResNet-V2[72] is a deeper network and has 164 layers with an input size of $299 \times 299 \times 3$. Considering the benefits of residual connections in deep learning, this method combines the inception architecture with residual connections, making training faster. The model achieves a top-5 accuracy of 95.3% on the ImageNet [11] dataset and outputs a feature vector of size 1536 after applying max pooling to the last convolution block.

ResNet-50 [24] has 50 convolution and pooling layers, which form the baseline architecture. Residual connections are added after each 3-layer block (residual block), where a 3-layer block consists of 1×1, 3×3, and 1×1 convolutions. The residual addition leads to easier and faster optimization with high accuracy. The input size is $224 \times 224 \times 3$, and it has a top-5 accuracy of 92.1% on ImageNet [11] dataset. Applying max-pooling to the conv_5 block which is the last convolution block outputs a 2048 size feature vector.

We implement our method in python using keras [9] deep learning framework. We load the weights from training on the ImageNet [11] dataset for each CNN model. We apply max-pooling to the last convolutional layer in the networks and take the resulting output as the frame spatial feature vector. We then train the bi-directional LSTM network in a 5-fold cross-validation for each set of the three pre-trained CNN extracted features. Then, we compare the performance of the pre-trained networks by considering the mean RMSE between the predicted scores and subjective MOS.

The MOS of LIVE-Qualcomm [19] and CVD2014 [48] were scaled to the same range as KoNVID-1K [26] to enable a uniform comparison. A plot of the mean RMSE against each pre-trained CNN is shown in Figure 5.2. It is seen that ResNet-50 [24]

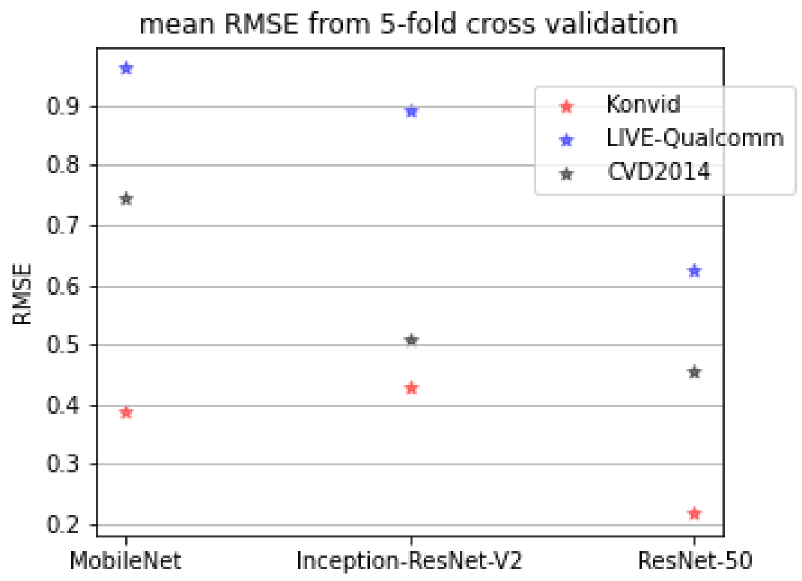

Figure 5.2: Performance comparison of pre-trained networks: MobileNet, ResNet-50 and Inception-ResNet-V2.

has the lowest average RMSE across the datasets, confirming its best performance among the three pre-trained networks. MobileNet has a lower mean RMSE than Inception-ResNet-V2 on KoNVid-1K [26] dataset. However, for LIVE-Qualcomm [19] and CVD2014 [48] datasets, Inception-ResNet-V2[72] performs better than MobileNet [27]. It is also observed that, the LIVE-Qualcomm [19] dataset produces relatively high RMSE values for all three pre-trained networks. Based on the results in Figure 5.2, ResNet-50 [24] was chosen as the preferred CNN model for our method. All analyses made in the following sections are with reference to the ResNet-50 [24] as our feature extractor.

5.3 Model training and hyper-parameter tuning

In our proposed method, the primary model for optimization is the network of the bi-directional LSTM and the two fully-connected layers (see Figure 4.1). Using the CNN extracted spatial features, we train the network to output a single video quality score with a high correlation to the MOS.

We do not temporally sample the video frames but we use all frames of a video. The frames are resized to $224 \times 224 \times 3$. The number of frames of a video determines the length of the sequence input to the LSTM network. Most videos in the Konvid-1k [26] and LIVE-Qualcomm [19] datasets have 240 frames; consequently, the sequence length selected for these two datasets was 240. Feature vectors of videos with less than 240 frames were zero-padded. For CVD2014 [48], majority of the videos had 500 frames, hence, the chosen sequence length was 500.

The parameters of our network architecture are the number of layers and the number of units per layer. The hyper-parameters of our network are: batch-size, learning rate, kernel regularizers, and merge mode (see equation 4.8). Our objective is to find one set of optimal values of these parameters for all three datasets. For this, we merged the training sets of KoNVID-1K [26] and CVD2014 [48] (since both have videos of same length) into one training set for parameter tuning. For the merge mode in equation 4.8, we experimented with four modes: average, sum, multiplication, and simple concatenation; we selected simple concatenation because it yielded the best results in terms of the mean squared error between the predicted video quality scores and the MOS. The rest of the parameters we tuned using the hyperband [37] search method in keras tuner [49]. The hyperband [37] method is a sophisticated version of random search approach where resource allocation is optimized during tuning. That is, the method runs a model for several combination of the parameters for a few epochs. Based on the results from these few epochs, the method selects the best candidates for the best parameter combination model and train them further. This strategy is repeated until an overall best model is attained. We provided the hyperband [37] search method with these range of values: number of LSTM layers: $1-8$; number of fully-connected layers: 2, number of units: 32 - 2048; batch size: 16, 32, and 64; learning rate: $0.0000001-0.1$ (we used the Adam [30] optimizer); and L_2 kernel regularizers: $0.0-0.1$ (we used L_2 loss function). Subsequently, we obtained the optimal architecture parameter values: number of LSTM layers = 6 (see also Figure 4.1), number of units 128, 64, 32, and the optimal hyper-parameters values: batch-size = 32, kernel regularizers = 0.1, learning rate = 0.00015.

The number of epochs, a parameter of the network optimizer (in our case Adam [30]) needs also to be set. Recall that VQA datasets are not divided by their authors into training, validation, and test sets. For us to observe the effect of number of epochs on the learning of our model on a dataset, we split its training set into a training subset and a validation subset in the ratio of 0.8 : 0.2. Then, we trained our model on KoNVid-1K [26] and CVD2014 [48] for 60 epochs and on LIVE-Qualcomm [19] for 100 epochs. Presented in Figure 5.3 are the learning curves from the model training.

These curves show the progress of the model performance in terms of the training

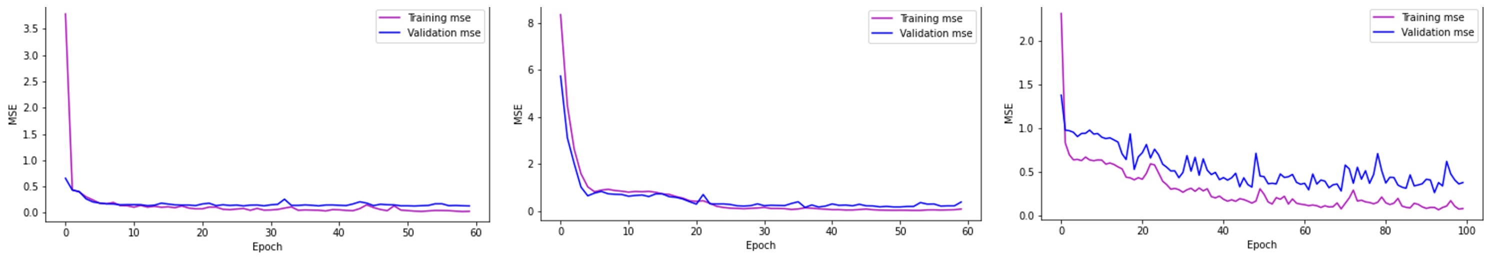

Figure 5.3: Mean squared error versus the number of epochs on the training and validation set of: left KoNVid-1K , middle CVD2014, and right LIVE-Qualcomm.

and validation error (that is, MSE between predicted quality and MOS) as we increase the number of epochs. From the graphs, it is seen that, the training and validation error decreases for a certain number of epochs until there is no significant changes in the error, indicating no overfitting, that is, no bias of our model to a particular dataset. There are more fluctuations in the training and validation error for LIVE-Qualcomm [19] than KoNVid-1K [26] and CVD2014 [48]. In fact, initial training of the model on LIVE-Qualcomm [19] using 5-fold cross-validation on the training set yielded poor performance irrespective of how many times the model parameters were tweaked. This was not unexpected since as stated in [35] and shown in [84], the videos in LIVE-Qualcomm [19] dataset pose challenges for both human viewers and subjective VQA models. Also, the LIVE-Qualcomm dataset [19] has video frames with high resolution 1920×1080 (compared to 1280×720 in KoNVid-1K dataset [26]); on the other side, CNN models downsample input images (for example, in ResNet-50 [24] to 224×224). Such strong downsampling might affect accuracy. Hence, similar to [84], our model weights obtained from training on KoNVid-1K [26] training set were used to train on the LIVE-Qualcomm [19] train set for 100 epochs.

After obtaining the optimal number of epochs using the training and validation subsets, we trained our model on the whole training set of each dataset. The training time for KoNVid-1K [26] training set with 960 videos for 60 epochs was 2 hrs, 5 mins and 40 seconds on a NVIDIA Quadro RTX 8000 GPU of 260W capacity. To estimate the testing time for a single video, we applied our model on a 240 frames video with resolution 540×960; predicting its video quality took 8 seconds, that is, 30 frame/second (FPS).

Chapter 6

Simulation Results and Analysis

6.1 Comparison with related work

We compare the performance of our model on the three datasets with three no-reference VQA methods, V-BLIINDS [58], TLVQM [31], and VSFA [35], whose implementation details and code are publicly available. The codes of the related works [40, 83, 84] are not publicly available, hence, we could not compare to them. Also, we note that each of these three related works carry out simulations under different setups: [83] samples videos in KoNVid-1K [26], CVD2014 [48] and LIVE-Qualcomm [19] at one frame per second and resizes the frames to 448×448. They run their method ten times and report the mean correlation coefficients between the predicted scores and the subjective scores. The method in [84] conducts experiments on KoNVid-1K [26] and LIVE-Qualcomm [19] datasets. They select 90% and 80% of videos in KoNVid-1K [26] and LIVE-Qualcomm [19] respectively as the training set, run the method one time, and report the model performance on the remaining videos in the datasets.

We train and evaluate V-BLIINDS [58], VSFA [35], and TLVQM [31] codes strictly following the requirements specified by the authors. In V-BLIINDS [58], Image Naturalness Quality Index, NIQE, Natural Scene Statistics, NSS, and motion characteristics of videos are computed in MatLab. Using SVR, the extracted features are trained to predict the video score. VSFA's [35] method is implemented in python using Pytorch [54]. Their architecture involves ResNet-50 [24], a fully connected layer and a GRU layer. The model was trained for 2000 epochs using a learning rate of 0.00001, and a batch size of 16. In TLVQM [31], low and high complexity features are computed in Matlab. Training and testing using the features computed are done in python using an RBF-kernel SVR with parameters, gamma = 0.1, epsilon=0.3 (again, as provided by the authors).

For comparison, in VQA literature, some related work train and test in a single run [84, 2], while others do multiple random splits and hence multiple runs [35, 83, 31, 79]. We decided to train and test using multiple random splits in 10 runs, making sure to use the same train/test set for each method. We report the mean of the PLCC, SROCC, and RMSE indices from the 10 runs in Table 6.1. The best results are in bold.

Table 6.1: Comparison over 10 runs of the proposed method with VBLIINDS [58], TLVQM [31] and VSFA [35].

	Konvid-1K			LIVE-Qualcomm			CVD2014		
Method	PLCC	SROCC	RMSE	PLCC	SROCC	RMSE	PLCC	SROCC	RMSE
V-BLIINDS	0.432	0.433	0.576	0.532	0.557	11.028	0.342	0.342	20.224
VSFA	0.723	0.720	0.454	0.722	0.723	**8.255**	0.718	0.711	14.100
TLVQM	0.731	0.732	0.441	0.772	0.760	8.389	0.714	0.691	14.869
OURS	**0.808**	**0.795**	**0.407**	**0.786**	**0.766**	8.339	**0.868**	**0.858**	**12.019**

The values in Table 6.1 shows that our method outperforms all compared methods in terms of the mean PLCC and SROCC. For the RMSE metric, VSFA [35] slightly outperforms our method on the LIVE-Qualcomm [19] dataset. VSFA [35] and TLVQM [31] have similar performance characteristics. However, TLVQM [31] has overall better mean values than VSFA [35]. Our method has a substantial increase in performance on all datasets over TLVQM [31]. Considering the mean PLCC, our method improved by 10.54% on KoNVid-1K [26], 1.8% on LIVE-Qualcomm, and 21.6% on CVD2014 [48] over TLVQM [31].

With regards to the comparison of our method's performance on the different datasets used, it is seen that LIVE-Qualcomm [19] proved challenging, evident by our method having its lowest performance on the dataset. This could be attributed to videos in LIVE-Qualcomm [19] having a higher resolution (1920×1080) than the others. The drastic resizing into 224×224 for spatial feature extraction by the pre-trained CNN may have hindered the model's ability to capture the complete spatial characteristics of the videos.

6.2 Stability study

To assess the stability of the proposed method, we conduct 100 runs on 100 random train/test splits of each dataset for VSFA [35], TLVQM [31] and our method making sure to use the same train/test split for each method. We did not include V-BLIINDS [58] in this experiment because it showed inferior performance as per Table 6.1. We evaluate the changes in each method's performance as the input data changes by reporting the mean and standard deviation (STD) of PLCC, SROCC and RMSE metrics. The results are shown in Table 6.2. It is seen that our method has the best mean values for all three metrics, but more importantly, has clearly, the lowest STD across all three datasets for all three metrics. On KoNVid-1K [26] dataset (the largest), the margin between the STD of our method and TLVQM [31] is between $0.006 - 0.02$, while that of VSFA [35] is between $0.03 - 0.058$. Although neural networks are known to be very stochastic in nature, it is seen that our method is more stable than TLVQM [31] and VSFA [35].

Table 6.2: 100 runs: Stability analysis of the proposed method, TLVQM [31] and VSFA [35].

Dataset	Method	PLCC (±STD)	SROCC (±STD)	RMSE (±STD)
KoNVid-1K	VSFA [35]	0.615 (±0.068)	0.576 (±0.083)	0.469 (±0.063)
	TLVQM [31]	0.715 (±0.032)	0.719 (±0.031)	0.404 (±0.053)
	OURS	**0.812 (±0.020)**	**0.803 (±0.025)**	**0.399 (±0.033)**
LIVE-Qualcomm	VSFA [35]	0.646 (±0.091)	0.669 (±0.077)	8.254 (±1.578)
	TLVQM [31]	0.717 (±0.081)	0.730 (±0.070)	8.204 (±1.576)
	OURS	**0.754 (±0.047)**	**0.735 (±0.053)**	**8.299 (±1.361)**
CVD2014	VSFA [35]	0.632 (±0.081)	0.596 (±0.094)	15.214 (±1.423)
	TLVQM [31]	0.706 (±0.067)	0.677 (±0.080)	15.138 (±1.413)
	OURS	**0.852 (±0.034)**	**0.857 (±0.048)**	**12.929 (±1.350)**

6.3 Ablation study

We conducted an ablation study to highlight the importance of the bi-directional LSTM module in the proposed method. We replace the bi-directional LSTM submodule by wrapping the mask layer and 32-unit fully connected layer in a keras time-distributed layer. The time-distributed layer allows a layer to apply the same weight to each step of a time series, similar to how LSTMs work except it has no memory cell. This process results in a 2-dimensional output. This 2-D output for each video sequence is flattened to obtain a 1-D linear vector which is fed into the last fully connected layers that outputs the video quality score. The time-distributed and flatten layers used in this part of the experiment are similar to convolution or pooling layers [9]. We run the new version of the proposed method ten times, and we report the results in Table 6.3. As expected, the bi-directional LSTM is central to the performance of our method. In Table 6.3, we see a huge difference between the correlation indices of the full proposed method and the LSTM-replaced version. Clearly, the powerful memory and long-term dependency modelling capabilities of LSTM units helps to predict video quality scores that correlates highly with subjective

Table 6.3: Comparison of the proposed method with LSTM and with LSTM-replaced.

Dataset		PLCC	SROCC	RMSE
KoNVid-1K	w	0.8082	0.7956	0.4070
	w/o	0.1783	0.1667	1.3271
LIVE-Qualcomm	w	0.7864	0.7657	8.3393
	w/o	0.0809	0.0837	49.100
CVD2014	w	0.8680	0.8587	12.020
	w/o	0.2629	0.2610	50.780

scores.

Additionally, we analyze the performance of the proposed method by removing one bi-directional LSTM layer at a time. That is, we run the proposed method using just the first bi-directional LSTM layer, then using the first 2, then the first 3 and so on up to using the 6 layers. We run these experiments 10 times using the previously obtained optimal hyper-parameter values (see Section 5.3). We report the changes in the mean PLCC, SROCC and RMSE values in Figure 6.1.

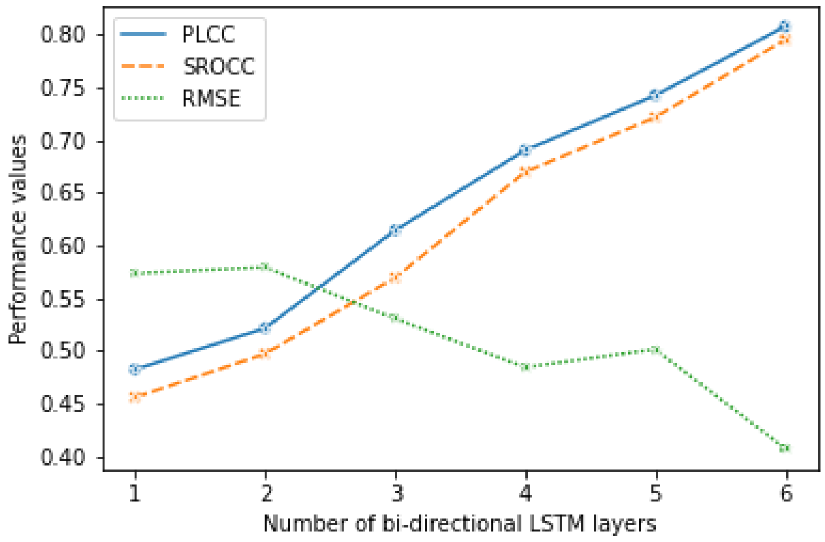

Figure 6.1: Performance comparison using different number of bi-directional LSTM layers.

From Figure 6.1, a gradual increase in the number of bi-directional LSTM layers improves the model's performance by an increase in the correlation indices and reduction in the squared error values. That is, the deeper we go, the better the model performance. In fact, in the beginning stages of the implementation of the proposed method, we found three 1024-unit bi-directional LSTM layers (a total of 3072 LSTM units) to be adequate during training. However, during the testing stage, the model performed poorly, indicating high level of overfitting. This situation influenced our decision to go deeper by increasing the number of layers while minimizing the model's width by reducing the number of units (We used a total of 640 LSTM units distributed across 6 layers).

Increasing the number of layers in a neural network allows the network to learn features at more different abstract levels. However, it must be noted that, going deeper increases computation time and more importantly, the risk of overfitting [71]. Our aim of tuning the architecture of the network is to achieve the optimal number of layers which is just right to approximate the underlying function in the data used and possess the ability to generalize well. And for that reason, we found six bi-directional LSTM layers to achieve satisfactory performance.

Chapter 7

Conclusions and Future work

7.1 Conclusions

In this thesis, we propose a general-purpose no-reference video quality assessment method using pre-trained CNN for spatial feature extraction and a stack of six bi-directional LSTM layers of different units, to model temporal dependencies in video signals. We assess three pre-trained CNNs with publicly available weights, MobileNet [27], Inception-ResNet-V2 [72], and ResNet-50 [24] to extract spatial features from video frames. We employ bi-directional LSTMs to model the temporal characteristics of videos given the extracted features per frame. The performance of the proposed method is compared with no-reference VQA methods, V-BLIINDS [58], VSFA [35], and TLVQM [31] on KoNVid-1K [26], LIVE-Qualcomm [19], and CVD2014 [48] datasets. Experimental results show that our method significantly outperforms related work in terms of the correlation coefficient between the predicted objective scores and the subjective MOS scores. Additionally, we conduct an ablation study which validates the importance of the bi-directional LSTM units and also, assess the high stability of the proposed method in comparison with other VQA methods.

7.2 Future work

Results from the experiments conducted revealed that, the LIVE-Qualcomm [19] dataset was a challenge to work with, and although the performance of our method on the dataset is acceptable, it could be improved. In future work, we would like to dig deeper into analyzing the model's performance on LIVE-Qualcomm [19] dataset and ways to improve it, for example, by detection and exclusion of "bad" samples.

In addition, we will explore the option of selecting keyframes for each video for feature extraction in order to decrease temporal redundancy, for example, by simply skipping every other frame or by non-uniform sampling.

Also, despite the remarkable ability of discriminative features extraction of pre-trained CNN models, inevitably it may generate non-discriminative features for some new classes beyond the dataset they were pre-trained on. In such cases, "memory selection" module can be useful to select those hard cases for the pre-trained model. The selected memory can be used to adjust the LSTM network, instead of updating the backbone model.

We will also look at the option of normalizing the MOS between 0 and 1 for all datasets during training in order to have a uniform range of predicted quality scores irrespective of the dataset the model is trained on.

Bibliography

[1] M. Agarla, L. Celona, and R. Schettini. "An efficient method for no-reference video quality assessment". In: *Journal of Imaging* 7.3 (2021), p. 55.

[2] S. Ahn and S. Lee. "Deep blind video quality assessment based on temporal human perception". In: *25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 619–623.

[3] L. Ali et al. "Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures". In: *Sensors* 21.5 (2021), p. 1688.

[4] J. Benesty et al. "Pearson correlation coefficient". In: *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[5] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[6] H. Boujut et al. "No-reference video quality assessment of H. 264 video streams based on semantic saliency maps". In: *IS&T/SPIE Electronic Imaging*. Vol. 8293. 2012, pp. 8293–28.

[7] T. Brandao and M.P. Queluz. "No-reference quality assessment of H. 264/AVC encoded video". In: *IEEE Transactions on Circuits and Systems for Video Technology* 20.11 (2010), pp. 1437–1447.

[8] R. Cahuantzi, X. Chen, and S. Güttel. "A comparison of LSTM and GRU networks for learning symbolic sequences". In: *arXiv preprint arXiv:2107.02248* (2021).

[9] F. Chollet et al. *Keras*. `https://keras.io`. 2015.

[10] Z.L. Chu, T.J. Liu, and K.H. Liu. "No-Reference Video Quality Assessment by A Cascade Combination of Neural Networks and Regression Model". In: *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2020, pp. 4116–4121.

[11] J. Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *IEEE conference on computer vision and pattern recognition.* 2009, pp. 248–255.

[12] M. Dimitrievski and Z. Ivanovski. “No-reference quality assessment of highly compressed video sequences”. In: *IEEE 15th International Workshop on Multimedia Signal Processing (MMSP).* 2013, pp. 266–271.

[13] J. Donahue et al. “Decaf: A deep convolutional activation feature for generic visual recognition”. In: *International conference on machine learning.* PMLR. 2014, pp. 647–655.

[14] Q. Dou et al. “Automatic detection of cerebral microbleeds from MR images via 3D convolutional neural networks”. In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1182–1195.

[15] J. Duchi, E. Hazan, and Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011).

[16] V. Frants et al. “Blind visual quality assessment for smart cloud-based video storage”. In: *IEEE International Conference on Smart Cloud (SmartCloud).* 2018, pp. 171–174.

[17] L. Gatys, A. Ecker, and M. Bethge. “A Neural Algorithm of Artistic Style”. In: *Journal of Vision* 16.12 (2016), pp. 326–326.

[18] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* ” O’Reilly Media, Inc.”, 2019.

[19] D. Ghadiyaram et al. “In-capture mobile video distortions: A study of subjective behavior and objective algorithms”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.9 (2017), pp. 2061–2077.

[20] A. Graves et al. “A novel connectionist system for unconstrained handwriting recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 31.5 (2008), pp. 855–868.

[21] A. Graves and N. Jaitly. “Towards end-to-end speech recognition with recurrent neural networks”. In: *International conference on machine learning.* PMLR. 2014, pp. 1764–1772.

[22] Z. Guan et al. “A novel objective quality assessment method for video conferencing coding”. In: *China Communications* 16.4 (2019), pp. 89–104.

[23] J. Han and C. Moraga. "The influence of the sigmoid function parameters on the speed of backpropagation learning". In: *International workshop on artificial neural networks*. Springer. 1995, pp. 195–201.

[24] K. He et al. "Deep residual learning for image recognition". In: *IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[25] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[26] V. Hosu et al. "The Konstanz natural video database (KoNViD-1k)". In: *IEEE Ninth international conference on quality of multimedia experience (QoMEX)*. 2017.

[27] A.G. Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[28] B. Karlik and A. V. Olgac. "Performance analysis of various activation functions in generalized MLP architectures of neural networks". In: *International Journal of Artificial Intelligence and Expert Systems* 1.4 (2011), pp. 111–122.

[29] A. Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

[30] D.P Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (Poster)*. 2015.

[31] J. Korhonen. "Two-level approach for no-reference consumer video quality assessment". In: *IEEE Transactions on Image Processing* 28.12 (2019), pp. 5923–5938.

[32] A Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *Master's thesis, University of Tront* (2009).

[33] A. Krizhevsky, I. Sutskever, and G.E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[34] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[35] D. Li, T. Jiang, and M. Jiang. "Quality assessment of in-the-wild videos". In: *27th ACM International Conference on Multimedia*. 2019, pp. 2351–2359.

[36] D. Li, T. Jiang, and M. Jiang. "Recent advances and challenges in video quality assessment". In: *ZTE Communications* 17.1 (2019), pp. 3–11.

[37] L. Li et al. "Hyperband: A novel bandit-based approach to hyperparameter optimization". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816.

[38] R. Li, B. Zeng, and M.L. Liou. "A new three-step search algorithm for block motion estimation". In: *IEEE transactions on circuits and systems for video technology* 4.4 (1994), pp. 438–442.

[39] S. Li et al. "Image quality assessment by separately evaluating detail losses and additive impairments". In: *IEEE Transactions on Multimedia* 13.5 (2011), pp. 935–949.

[40] Y. Li et al. "Video quality assessment with deep architecture". In: *IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA).* 2021, pp. 268–271.

[41] Z. Li et al. "Toward a practical perceptual video quality metric". In: *The Netflix Tech Blog* 6.2 (2016).

[42] T. Lin et al. "Microsoft COCO: Common objects in context". In: *European conference on computer vision.* Springer. 2014, pp. 740–755.

[43] W. Lu et al. "A spatiotemporal model of video quality assessment via 3D gradient differencing". In: *Information Sciences* 478 (2019), pp. 141–151.

[44] W.S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[45] L. Mou, Y. Hua, and X. Zhu. "Relation matters: Relational context-aware fully convolutional network for semantic segmentation of high-resolution aerial images". In: *IEEE Transactions on Geoscience and Remote Sensing* 58.11 (2020), pp. 7557–7569.

[46] V. Nair and G.E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Icml.* 2010.

[47] Y. Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2)". In: *Doklady an ussr.* Vol. 269. 1983, pp. 543–547.

[48] M. Nuutinen et al. "CVD2014 — A database for evaluating no-reference video quality assessment algorithms". In: *IEEE Transactions on Image Processing* 25.7 (2016), pp. 3073–3086.

[49] T. O'Malley et al. *Keras Tuner*. `https://github.com/keras-team/keras-tuner`. 2019.

[50] I. Oksuz et al. "Deep learning-based detection and correction of cardiac MR motion artefacts during reconstruction for high-quality segmentation". In: *IEEE Transactions on Medical Imaging* 39.12 (2020), pp. 4001–4010.

[51] S. Pang et al. "Spineparsenet: spine parsing for volumetric MR image by a two-stage segmentation framework with semantic image representation". In: *IEEE Transactions on Medical Imaging* 40.1 (2020), pp. 262–273.

[52] J. Park et al. "Video quality pooling adaptive to perceptual distortion severity". In: *IEEE Transactions on Image Processing* 22.2 (2012), pp. 610–620.

[53] R. Pascanu, T. Mikolov, and Y. Bengio. "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*. PMLR. 2013, pp. 1310–1318.

[54] A. Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[55] M.H. Pinson and S. Wolf. "A new standardized method for objectively measuring video quality". In: *IEEE Transactions on broadcasting* 50.3 (2004), pp. 312–322.

[56] S. Rimac-Drlje, M. Vranjes, and D. Zagar. "Influence of temporal pooling method on the objective video quality evaluation". In: *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*. 2009, pp. 1–5.

[57] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[58] M.A. Saad, A.C. Bovik, and C. Charrier. "Blind prediction of natural video quality". In: *IEEE Transactions on Image Processing* 23.3 (2014), pp. 1352–1365.

[59] H. Sak, A. Senior, and F. Beaufays. "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition". In: *arXiv preprint arXiv:1402.1128* (2014).

[60] A.L. Samuel. “Some studies in machine learning using the game of checkers. II—Recent progress”. In: *IBM Journal of research and development* 11.6 (1967), pp. 601–617.

[61] M.J. Scott et al. “Do personality and culture influence perceived video quality and enjoyment?” In: *IEEE Transactions on Multimedia* 18.9 (2016), pp. 1796–1807.

[62] K. Seshadrinathan and A.C. Bovik. “Motion tuned spatio-temporal quality assessment of natural videos”. In: *IEEE transactions on image processing* 19.2 (2009), pp. 335–350.

[63] K. Seshadrinathan and A.C. Bovik. “Temporal hysteresis model of time varying subjective video quality”. In: *IEEE international conference on acoustics, speech and signal processing (ICASSP).* 2011, pp. 1153–1156.

[64] H.O. Shahreza, A. Amini, and H. Behroozi. “No-reference video quality assessment using recurrent neural networks”. In: *IEEE 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS).* 2019, pp. 1–5.

[65] H.R. Sheikh and A.C. Bovik. “Image information and visual quality”. In: *IEEE Transactions on image processing* 15.2 (2006), pp. 430–444.

[66] D.J. Sheskin. “Spearman’s rank-order correlation coefficient”. In: *Handbook of parametric and nonparametric statistical procedures* 1353 (2007).

[67] M. Shi, K. Wang, and C. Li. “A C-LSTM with word embedding model for news text classification”. In: *IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS).* 2019, pp. 253–257.

[68] K. Simonyan, A. Vedaldi, and A. Zisserman. “Deep inside convolutional networks: visualising image classification models and saliency maps”. In: *International Conference on Learning Representations.* 2014.

[69] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[70] R. Soundararajan and A.C. Bovik. “Video quality assessment by reduced reference spatio-temporal entropic differencing”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 23.4 (2012), pp. 684–694.

[71] C. Szegedy et al. “Going deeper with convolutions”. In: *IEEE conference on computer vision and pattern recognition.* 2015, pp. 1–9.

[72] C. Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[73] C. Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[74] B. Thomee et al. “The new data and new challenges in multimedia research”. In: *arXiv preprint arXiv:1503.01817* 1.8 (2015).

[75] T. Tominaga et al. “Performance comparisons of subjective quality assessment methods for mobile video”. In: *IEEE Second international workshop on quality of multimedia experience (QoMEX)*. 2010, pp. 82–87.

[76] Z. Tu et al. “A comparative evaluation of temporal pooling methods for blind video quality assessment”. In: *IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 141–145.

[77] D. Varga. “No-reference video quality assessment based on the temporal pooling of deep features”. In: *Neural Processing Letters* 50.3 (2019), pp. 2595–2608.

[78] P.V. Vu, C.T. Vu, and D.M. Chandler. “A spatiotemporal most-apparent-distortion model for video quality assessment”. In: *18th IEEE International Conference on Image Processing*. 2011, pp. 2505–2508.

[79] C. Wang, L. Su, and W. Zhang. “COME for no-reference video quality assessment”. In: *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. 2018, pp. 232–237.

[80] Z. Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[81] J. Xu et al. “No-reference video quality assessment via feature learning”. In: *IEEE international conference on image processing (ICIP)*. 2014, pp. 491–495.

[82] F. Yang et al. “No-reference quality assessment for networked video via primary analysis of bit stream”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 20.11 (2010), pp. 1544–1554.

[83] F. Yi et al. “Attention Based Network For No-Reference UGC Video Quality Assessment”. In: *IEEE International Conference on Image Processing (ICIP)*. 2021, pp. 1414–1418.

[84] J. You and J. Korhonen. “Deep neural networks for no-reference video quality assessment”. In: *IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 2349–2353.

[85] C. Zhang and J. Kim. “Object detection with location-aware deformable convolution and backward attention filtering”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9452–9461.

[86] Y. Zhang et al. “Blind video quality assessment with weakly supervised learning and resampling strategy”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.8 (2018), pp. 2244–2255.

[87] Y. Zhang, J. Lu, and J. Zhou. “Objects are different: Flexible monocular 3D object detection”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3289–3298.

[88] W. Zhou and Z. Chen. “Deep local and global spatiotemporal feature aggregation for blind video quality assessment”. In: *IEEE International Conference on Visual Communications and Image Processing (VCIP)*. 2020, pp. 338–341.

[89] K. Zhu et al. “A no-reference video quality assessment based on laplacian pyramids”. In: *IEEE International Conference on Image Processing*. 2013, pp. 49–53.