TRABALHO DE CONCLUSÃO DE CURSO

USING TRANSFORMERS TO BLINDLY ESTIMATE AUDIO-VISUAL QUALITY FOR VIDEO CONTENTS

Guilherme Andrey Medeiros Ribeiro

Brasília, Setembro de 2022

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE CONCLUSÃO DE CURSO

USING TRANSFORMERS TO BLINDLY ESTIMATE AUDIO-VISUAL QUALITY FOR VIDEO CONTENTS

Guilherme Andrey Medeiros Ribeiro

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof.ª Dr.ª Mylène Christine Queiroz de Farias, ENE/UnB
Orientadora _____

MSc. André Henrique Macedo da Costa, ENE/UnB
Examinador interno _____

Dr. Helard Alberto Becerra Martinez, School of Computer Science, University College Dublin
Examinador externo _____

Dedicatória

A Deus, que me deu a vida e a minha razão de viver.

Guilherme Andrey Medeiros Ribeiro

Agradecimentos

Acima de qualquer coisa, agradeço em primeiro lugar a Deus, por todas as coisas que fez por mim. Foi Ele que me deu a vida, todas as minhas capacidades e habilidades, o meu sustento diário, minha razão de viver, e a minha salvação. A Ele seja dada toda a honra, toda a glória e todo o louvor.

Agradeço à minha família, pelo companheirismo, amizade, amor e suporte dado por todos esses anos. Agradeço especialmente aos meus pais, Aureny e Jairo, e meu irmão Lucas, que são a minha base, meu suporte, que sempre estiveram comigo e me sustentaram em tudo que faço. Agradeço aos meus pais pela criação que me deram, me ensinando os caminhos de Deus, e sempre se esforçando tanto para que eu tivesse uma boa qualidade de vida, sendo exemplos para mim na integridade, esforço, amor, carinho, fé, e tantas outras coisas. E agradeço ao meu avô, Arlindo, o melhor eletricista de Passo Fundo, que me incentivou muito a seguir nesse caminho.

Agradeço à minha namorada, Amanda. Esses últimos meses ao seu lado têm sido os melhores da minha vida. Obrigado pelo seu companheirismo, amor, carinho, paciência, compreensão, e por me apoiar em tudo que eu faço. Por me desafiar todos os dias a ser melhor, não só com palavras, mas pelo exemplo, sendo uma mulher trabalhadora, comprometida, inteligente, forte, altruísta e, acima de tudo, temente a Deus.

Agradeço aos meus amigos, por sempre estarem ao meu lado. Obrigado Alessandro, por nossa amizade duradoura, a mais longa e impactante que tenho. Obrigado aos mais recentes, mas muito especiais, Júlio, Kamila, Letícia, Maicon, Pedro, Sady, Victória, Talita, Janiffer, Ariel, Juliana, Gabriel e Sabrina. Ter amigos como vocês é o que me sustenta e ajuda a renovar minha fé todos os dias. E obrigado aos amigos que a UnB me deu, Isabel, Klebão, Maria e Mateus. Vocês foram ótimas companhias nesses anos de graduação, e que com certeza levarei para a vida.

E por fim, agradeço às pessoas que me ajudaram no decorrer desse curso. Obrigado Pedro e Osmar, por me introduzir e mentorear na área de IA, e por sempre estarem dispostos a me ajudar quando tenho qualquer dúvida ou dificuldade. E por fim, mas muito importante, agradeço à professora Mylène, por ter sido uma excelente orientadora e pela sua disponibilidade em me ajudar sempre que eu precisei, mesmo com sua agenda sempre lotada. A senhora foi de longe a melhor professora que eu tive nessa universidade.

Guilherme Andrey Medeiros Ribeiro

RESUMO

Nos últimos anos, a quantidade de conteúdo multimídia disponível na internet cresceu abruptamente devido aos avanços em tecnologias de transmissão e compressão, e as plataformas de *streaming* são responsáveis por boa parte desse volume. A transmissão de dados por redes com ou sem fio causa distorções no sinal sendo transmitido. Por causa disso, a confiabilidade na qualidade dos dados transmitidos é cada vez mais importante para provedores de serviços multimídia, dado que isso é um importante fator no processo de decisão do usuário final sobre qual serviço contratar. Assim, a necessidade de ferramentas de monitoramento de qualidade em tempo real eficientes é crescente na indústria, levando à demanda de desenvolvimento de métodos de avaliação de qualidade sem referência para vídeos.

Atualmente, poucos métodos propostos usam características de áudio e visuais concorrentemente para prever a qualidade de vídeo, apesar de áudio e vídeo terem relações importantes que juntas moldam a qualidade de experiência do usuário final quando consumindo conteúdos áudio-visuais. Esse trabalho propõe o uso da arquitetura *Transformer* para atacar o problema de avaliação de qualidade sem referência. O método proposto usa um conjunto de 115 características audio-visuais extraídas de 800 vídeos obtidos de uma base de dados pública para prever qualidade. O modelo obteve resultados ruins quando comparado com outros métodos apresentados na literatura, indicando que usar características tabulares em uma arquitetura de *Transformer* não é uma abordagem para o problema. Uma discussão sobre o uso de *Transformers* para esse tipo de tarefa e sobre o porquê de ter tido uma performance ruim é apresentado durante o trabalho, e possíveis alternativas para o problema de avaliação de qualidade áudio-visual combinada sem referência também são discutidas.

Palavras-chave: Sem-referência, Qualidade Áudio-visual, Transformers, Avaliação de Qualidade, Qualidade de Vídeo.

ABSTRACT

In the latest years, the amount of multimedia content available on the Internet has increased abruptly with advances in transmission and compression technologies, and streaming platforms are responsible for great part of that volume of information. Transmitting data through wired or wireless networks causes distortion to the transmitted signal. Thus, the reliability in the quality of the transmitted data is essential for multimedia service providers, since this is an important factor in attracting customers for the provided service. Because of that, the need for efficient real-time quality monitoring tools is rapidly growing in the industry, leading to the demand of developing precise no-reference quality assessment methods for video contents.

Currently, few methods using both audio and visual features of a video signal concurrently were proposed, notwithstanding that audio and video have important relationships that shape the Quality of Experience a user has when consuming video contents. This work proposes the use of the Transformer architecture to tackle the problem of no-reference audiovisual quality assessment. The proposed method uses a set of 115 audiovisual features from 800 videos obtained from a public database to predict quality. The model achieved poor results when compared to other metrics present in the literature, indicating that using tabular features in a Transformer architecture is not a good approach for this task. A discussion on the use of Transformers for this task and why they perform poorly on this kind of task is presented throughout this work, and possible alternatives for the combined audiovisual quality assessment problems are discussed.

Index terms: No-reference, Audiovisual Quality, Transformers, Quality Assessment, Video Quality

Contents

List of figures

List of tables

List of symbols

Greek letters

α	Linear combination coefficient/attention weight
σ	Standard deviation/Sigmoid
θ	Partial derivative
∇	Gradient

List of abbreviations

Quality Assessment

QoE	Quality of Experience
FR	Full-reference
RR	Reduced-reference
NR	No-reference
AV	Audiovisual
AVQ	Audiovisual quality
AVQA	Audiovisual quality assessment
QA	Quality assessment
QoS	Quality of service
MOS	Mean Opinion Score
PSNR	Peak-to-signal noise ratio
SSIM	Structural Similarity Index
VIF	Visual Information Fidelity
SNR	Signal-to-noise ration
AQA	Audio quality assessment
IQA	Image quality assessment
VQA	Video quality assessment
ACR	Absolute Category Rating
MQS	Mean Quality Score
NAViDAd	No-reference Audiovisual Quality metric based on a Deep Autoencoder

Machine Learning

ML	Machine Learning
DNN	Deep Neural Network
FC	Fully-connected
FFN	Feedforward network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network

BiRNN	Bidirectional Recurrent Neural Network
LSTM	Long Short-Term Memory
ViT	Visual Transformer
NMT	Neural Machine Translation
MSE	Mean-squared error
RMSE	Root-mean-squared error
PCC	Pearson Correlation Coefficient
SCC	Spearman Correlation Coefficient
ReLU	Rectified Linear Unit
GELU	Gaussian Error Linear Unit
Adam	Adaptive Moment Estimation
LN	Layer normalization
BN	Batch normalization
CS	Classification for sample prediction
CV	Classification for video prediction
RS	Regression for sample prediction
RV	Regression for video prediction

Others

SOTA	State-of-the-art
NSS	Natural Scene Statistics
NLP	Natural Language Processing
HVS	Human Visual System
HAS	Human Auditory System
ITU	International Telecommunication Union
HRC	Hypothetical reference circuit
ACF	Autocorrelation function

Chapter 1

Introduction

1.1 Contextualization

In recent years, the volume of multimedia content flow on the Internet has increased considerably with advances in transmission and compression technologies. Today, streaming (e.g., YouTube, Netflix, HBO Max) and virtual conference applications (e.g., Microsoft Teams, Skype, Zoom) are responsible for most of volume information traffic on the Internet [16].

The success of multimedia applications depends on the reliability of the service and the quality of experience (QoE) of the user, which takes into consideration characteristics of the Human Visual System (HVS) and the Human Auditory System (HAS). Data transmission through wired or wireless networks causes signal distortions, such as compression artifacts, package losses, bit errors, background noise, among others [17]. The most accurate way of determining the quality of a transmitted signal is through subjective experiments, where human subjects evaluate the quality of various video stimuli, scoring them in a pre-determined scale. However, for real time monitoring, it is impractical to perform subjective experiments in order to evaluate quality. Thus, the development of fast and accurate monitoring tools that objectively quantify the quality of a distributed signal in real time is of great interest to multimedia service providers.

In order to approach this issue, many objective quality metrics were developed in order to quantify the quality of a signal using a computational algorithm, aiming to attain similar scores to those obtained through subjective experiments. Those metrics can be classified into three categories, depending on the amount of information available for prediction, such as Full Reference (FR), Reduced Reference (RR), and No Reference (NR). In FR metrics, both reference and distorted signals are available. In RR metrics, only a set of features from the reference signal is used. In the NR metrics, only the distorted signal is available. Figure 1.1 illustrates the distinction between the three categories. Since FR and RR metrics need the reference signal to estimate quality, they cannot be used in streaming and monitoring scenarios. Thus, NR metrics are of great interest to the industry.

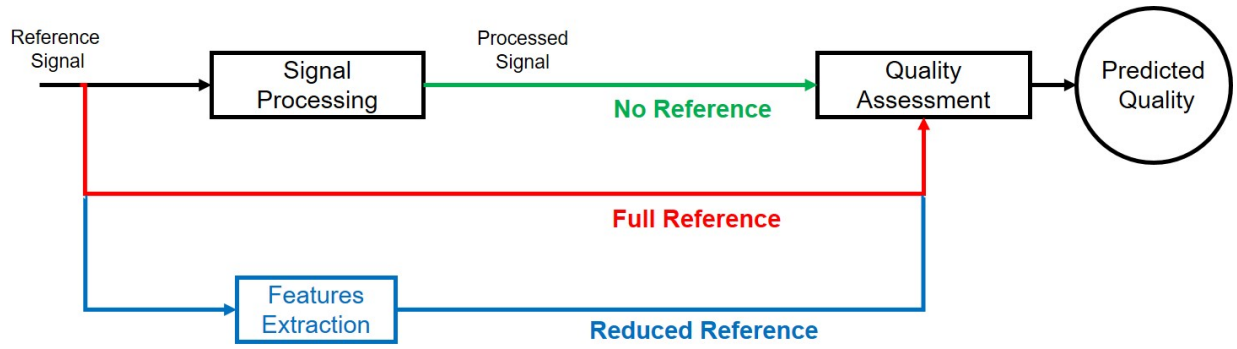


Figure 1.1: Block diagram illustrating the procedure for calculating FR, RR and NR metrics. **Green:** NR metrics procedure, where the distorted signal is the sole information available. **Blue:** RR metrics procedure, where besides the distorted signal, a set of features from the reference signal is available. **Red:** FR metrics procedure, where both the reference and distorted signals are available. Source: [1]

1.2 Problem description

The development of efficient NR metrics is still an active area of research. Because NR metrics only have access to the distorted signal, it is more difficult for them to correctly measure its quality compared to the FR and NR metrics. Earlier work on NR metrics focused on specific distortion types to measure quality, such as blockiness and contrast, for video, and noise, for audio [18][19][20]. Recently, researchers have focused on more general metrics that are capable of identifying multiple types of distortion, both for audio and video. Because of that, the use of machine learning (ML) algorithms to predict audio and video quality has increased in recent years, given their ability to learn complex patterns in data and their agnostic characteristic to a specific distortion type [21][22]. For the joint audiovisual quality assessment (AVQA) problem, most of the proposed methods evaluate audio and visual quality separately, combining them with a linear model to obtain an audiovisual quality (AVQ) metric. However, audio and video have essential relationships that affect QoE when combined, and few studies have tried to address both simultaneously.

This work uses a Transformer-based architecture for designing AVQ metrics, which uses audio and visual features extracted concurrently. The Transformer is a neural network architecture, designed for sequential data, that relies exclusively on attention mechanisms to extract information from the input signal in order to perform multiple tasks. It is the state-of-the-art (SOTA) method for multiple Natural Language Processing (NLP) and Computer Vision tasks, including Language Modeling [23], Image Recognition [24] and Video Object Segmentation [25]. In light of its recent success in various tasks, we believe that Transformers should be able to shape the underlying relationships between audio and visual signals in multimedia contents, through their self-attention layers. This model was inspired by the work of Martinez *et al.* [1], where an autoencoder-based model is used to combine features in the design of an AVQ metric. To the best of our knowledge, no model has been proposed for an audiovisual metric (AV) using the Transformer architecture.

1.3 Objectives

This work aims to present a novel algorithm, based on the Transformer architecture, capable of predicting the quality of a distorted video signal in a NR scenario. The model has to learn the intrinsic ties between audio and visual signals in order to learn patterns that shape the QoE of the end-user.

1.4 Manuscript organization

Chapter 2 presents a theoretical background of objective AVQA methods and previous works, while Chapter 3 discusses the background of ML, Neural Networks, and Transformers. In Chapter 4, the methodology of the proposed solution is explained. The experimental results are presented in Chapter 5, followed by the conclusions in Chapter 6.

Chapter 2

Objective Audiovisual Quality Assessment Methods

With the rapid advances in the transmission, generation, and applications of multimedia products, quality assessment (QA) of distributed signals has become an essential part of any multimedia service provider, since video and audio signals suffer distortions during coding, transmission and decoding. Some common distortions are coding, packet losses and frame freezing, for visual signals, and noise, clipping, echo and chop, for audio signals. Today, providing a good Quality of Service (QoS), which refers to the performance of multimedia applications and networks[26], to the user is not enough due to the need for studies in the area of QoE, which consider the Human Visual System (HVS) and the Human Auditory System (HAS), providing a more reliable assessment of the subjective quality of the service provided experienced by the consumer [1].

The subjective quality perceived by users is commonly quantified through experiments with human subjects, where a group of people judge the quality of individual or multiple test sequences. A widely used metric to determine the subjective quality of a stimuli is the Mean Opinion Score (MOS). The International Telecommunications Union (ITU) defined the opinion score as the 'value on a predefined scale that a subject assigns to his opinion of the performance of a system' [27]. Usually, the MOS is defined as the average quality value given by all subjects for each signal on the experiment. The MOS is commonly used as the ground truth for training objective quality metrics.

Many experimental methodologies for subjective experiments have been proposed and standardized, considering the variety, length, and types of distortion of the stimuli presented to each participant. A recent methodology proposed by Pinson [28], called immersive methodology, has the objective of putting the participant in a more natural scenario, presenting him with multiple hypothetical reference circuit (HRC) combinations. In this work, the HRC is a specific test condition of the source signal, that is, a combination of a video distortion and an audio distortion. In the immersive methodology, participants are asked to rate the general audiovisual quality of the video content, even if the HRC presents only one type of distortion (audio-only or video-only). Each participant is presented with each source only once, in order to reduce fatigue and avoid

memorization of the stimuli. In the immersive methodology, the MOS is calculated as the mean of the overall audiovisual quality values given by the participants for each content.

In recent years, many objective metrics for audio (AQA), image (IQA), and video QA (VQA) aim to be highly correlated with subjective quality scores provided by users, since subjective experiments are expensive, time-consuming, and difficult to automate for constant quality control. Most studies on NR quality metrics address audio or video signals separately. However, few addressed both simultaneously, despite the fact that audio and video signals jointly affect the QoE experienced by the final user [29]. This comes in part by the absence of relevant audiovisual public databases, since most public available databases comprise of audio-only or video-only content [26]. Some of the available public audiovisual databases are the UnB-Audiovisual Database[30] and LIVE-NFLX-II [31] databases.

This chapter presents a description of some quality metrics models, including FR and NR metrics, audio and video quality metrics, and combinations of audio and video quality metrics, which is the objective of the method proposed in this work.

2.1 Video Quality Metrics

2.1.1 Full-Reference Video Quality Metrics

FR metrics are the most widely explored QA methods, usually presenting good results when predicting subjective video quality. The Mean Square Error (MSE) and the related Peak-to-signal Noise Ratio (PSNR) are the most used metrics in the field because of their simplicity and convenience. However, they are criticized for having some underlying assumptions that are not sustained in visual signals [2], such as images and videos, and not taking into account aspects of HVS. Figure 2.1 shows how MSE fails to accurately measure the quality of an image, since different images with clearly different qualities result in the same MSE value.

To address this issue, the researchers proposed many other metrics that incorporate the characteristics of the HVS. Examples include the popular Structural Similarity Index (SSIM) [32] and some variations of it [33] [34], which take into account the luminance, contrast, and structure of the original and distorted signals. Other commonly used metrics are the Visual Information Fidelity (VIF), which uses Natural Scene Statistics (NSS), distortion models, and HVS properties to predict quality, and the Video Multimethod Assessment Fusion (VMAF), which fuses various quality image quality metrics and features using a Support Vector Machine (SMV) in order to predict quality.

2.1.2 No-reference Video Quality Metrics

Because NR metrics only have access to distorted video, early work on NR QA focuses on specific types of distortion, based on distortion features to make a prediction. Some examples of single-artifact-based algorithms include the work of Mittal *et al.* [35] and Farias and Mitra [19]. However,

Figure 2.1: Comparison of MSE and SSIM for an image with various distortion types and levels. Note that the MSE values for images (b)-(g) are very close, although the images have very different qualities and characteristics. SSIM, on the other hand, varies correctly with the subjective quality of the images. Other example are images (h)-(l), which visually seem similar, almost identical, but have very different MSE values, whilst the SSIM values stay close to each other. Source: [2]

work in NR QA indicates that perhaps the visual perception of quality is more likely to be related to nonlinear dependencies presented in natural scenes that are disturbed by noise. Thus, the use of NSS becomes a more general approach [36].

Many of the NR quality metrics used for video are in fact image quality metrics adapted to videos. Some of those available in the literature are distortion-specific, such as the work of Sheikh *et al.* [37], or capable of assessing quality for multiple types of distortion, such as the DIVIINE [38], C-DIVIINE [39] and BRISQUE[35] algorithms. An example of a pure VQA metric is the VIIDEO[40], that uses intrinsic statistical regularities present in videos to measure the distortion caused by irregularities.

In recent years, researchers have proposed many ML-based methods for NR QA, because of their ability to learn nonlinear relationships in data and their ability to extract meaningful visual and temporal features. Usually, these methods are based on features obtained by other objective

quality metrics algorithms, followed by some classification algorithm that learns the complex relationships between the features and the predicted quality score. An example is the work by Moorthy *et al.* [38], which first extracts NSS features and then uses regression to predict quality scores. However, with advances in studies of Deep Neural Networks (DNN), and specifically Convolutional Neural Networks (CNN), many methods today work in the spatial domain without using hand-crafted features [21, 22, 41].

2.2 Audio Quality Metrics

AQA metrics can be divided into two categories, intrusive and nonintrusive, equivalent to FR and NR in IQA. Intrusive methods use both the original and distorted audio signals, or features of those signals, in order to measure quality. Most algorithms proposed in AQA are intrusive and present better results than non-intrusive methods. Some important intrusive AQA metrics are the VISQOL[42], which calculates the similarity between the spectrograms of the original and distorted signals to predict quality, and the STOI[43], which uses a DTF-based time-frequency decomposition of the signal in order to compute intelligibility and predict quality.

Similarly to IQA, classical metrics such as the MSE and related Signal-to-noise ratio (SNR) are unable to estimate the subjective quality as perceived by listeners in AQA, since they are direct measures of difference between signals, failing to consider important aspects of the HAS. Today, many algorithms proposed for intrusive and non-intrusive AQA use ML techniques combined with feature extraction methods, considering the perceptual quality related to HAS. An example of these methods is the NISQA[44], a non-intrusive AQA method that uses Convolutional Neural Networks and Attention Mechanisms to predict quality using the Mel-spectrogram of a distorted audio signal.

The study of nonintrusive methods for AQA still needs much work, though some promising results have been found in recent years. Many purely non-intrusive methods have been proposed in the last years, especially using DNN architectures, with a great capability of producing good results for different types of distortions, given that they are able to learn the complex nonlinear relationships in the input signal that shape the perceived quality. These approaches usually use features from the distorted signals as input to the model, as the Mel spectrogram for example [45][46]. Other promising approach that has recently been explored is the use of attention mechanisms to extract key features from the feature space [47]. Finally, wav2vec [48], a pre-training strategy for speech intelligibility, was proposed recently to help models to perform better when little data is available, by using an unsupervised training with a Convolutional Neural Network on the raw audio signal.

2.3 Combined Audiovisual Quality Metrics

Although many methods exist for addressing audio and video quality separately, there are few approaches to the problem of combined audiovisual quality assessment. In many studies, the

approach taken to obtain a single AVQ metric is to make a linear combination of separate video and audio metrics, using the following equation [49] [50]:

$$\text{Quality}_{av} = \alpha_1 + \alpha_2 \cdot \text{Quality}_a + \alpha_3 \cdot \text{Quality}_v + \alpha_4 \cdot \text{Quality}_a \cdot \text{Quality}_v. \tag{2.1}$$

where α_1, α_2, α_3, and α_4 are the linear combination coefficients, Quality_a is the audio quality score and Quality_v is the video quality score. Other combination strategies were proposed by Becerra *et al.* [49]. The first uses a weighted minkowski function, given by the following equation:

$$\text{Quality}_{av} = (\alpha_1 \cdot \text{Quality}_a^p + \alpha_2 \cdot \text{Quality}_v^p)^{\frac{1}{p}}. \tag{2.2}$$

and the second uses a power-based model, given by:

$$\text{Quality}_{av} = (\alpha_1 + \alpha_2 \cdot \text{Quality}_a^{p_1} \cdot \text{Quality}_v^{p_2}). \tag{2.3}$$

However, these models do not consider that subjective quality is not shaped by separately assessing video and audio quality, but through the critical relationships between both stimuli and their respective distortion combinations, which could result in a complex nonlinear correlation. A recent study by Min *et al.* [29] proposed new methods for combining audio and video for FR AV quality prediction, using different methods of feature extraction followed by a fusion strategy, such as a Support Vector Regressor.

2.3.1 No-reference Audiovisual Quality metric based on a Deep Autoencoder

Martinez *et al.* [3] proposed a deep autoencoder model, called No-reference Audiovisual Quality metric based on a Deep Autoencoder (NAViDAd), taking input of audio and video signals. The

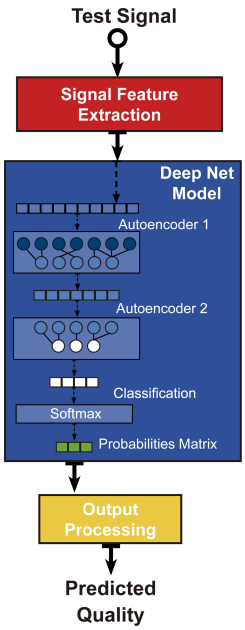


Figure 2.2: Block diagram of the NAViDAd model [3]. The input signal goes through two autoencoder models and a classification layer to generate the class probability matrix. The probabilities are then processed in order to generate the predicted quality. Source: [3].

method extracts independent audio and video features and uses the autoencoder module to generate new features that model the complex non-linear relationships between audio and video signals and predict quality. The Transformer model proposed in this work was inspired by the NAViDAd metric.

This metric uses two autoencoder structures and a classification layer in order to map the input data to the target quality scores. The training data consists of 800 video sequences presenting various types of distortion. The videos are submitted to a feature extraction procedure, which extracts 115 audiovisual features from every sample of the videos. After some pre-processing, the features are concatenated generating the global feature matrix, as well as the generation of the global target matrix by concatenating the quality values of all videos.

An illustration of the NAViDAd architecture is presented in Figure 2.2. The global feature matrix is passed through the first autoencoder model, which is trained to represent the input in a lower dimension. Then, the output of the encoder part of the autoencoder, which is low-dimensional, is used as input to a second autoencoder model, which lowers the dimension of the model even further. The low-dimensional features extracted from the output of the second autoencoder are used as input to the final classification layer, which is responsible for mapping the input to the predicted quality.

2.4 Performance of quality metrics

In the quality assessment field, it is common to use correlation coefficients to evaluate the performance of quality assessment methods, since they represent how much the predicted quality varies with the actual quality. The most commonly used metrics are the Pearson Correlation Coefficient (PCC) and Spearman Correlation Coefficient (SCC). The PCC is given by the equation below:

$$PCC(x, y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}, \tag{2.4}$$

where $\text{cov}(X, Y)$ is the covariance between the true and predicted labels vectors, σ_X is the standard deviation of the true labels vector and σ_Y is the standard deviation of the predicted labels vector. The SCC, which is based on the rank of the values in the vector, is given by the following equation:

$$SCC(x, y) = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n\left(n^2 - 1\right)}, \tag{2.5}$$

where d_i is the distance between the ranks of the true and predicted values and n is the size of the vector of labels.

Chapter 3

Machine Learning and Transformers

Machine Learning (ML) is the field of computer science consisting of developing algorithms capable of learning to perform a task from a given data without being explicitly programmed to solve it. Alpaydin [51] gives a good definition of ML: "Machine learning is programming computers to optimize a performance criterion using example data or experience." The ML model can perform the given task on unseen data by learning patterns in the data and optimizing internal parameters.

The Transformer is a neural network model that relies exclusively on the attention mechanism to extract meaningful information from the input data. Vaswani *et al.* [15] proposed an alternative to Recurrent Neural Networks (RNNs), which are computationally expensive, as they do not use parallel input simultaneously. The proposed solution used attention layers to identify critical parts of the input data so that all information could be fed to the network at once, making it computationally efficient while maintaining temporal relationships.

This chapter will begin by presenting some basic concepts of ML, such as supervised and unsupervised learning and some performance metrics. Then, we will explore important concepts of DNNs, including CNNs and RNNs. Finally, we will present the Attention Mechanism, altogether with the Transformers methods, which are the base of the proposed solution in this work.

3.1 Machine Learning Basics

ML algorithms are divided into supervised and unsupervised learning. Supervised learning algorithms make predictions on labeled data. That is, when both the input and the output of the model are known. The objective of supervised learning models is to, given the input data, predict the output data with minimum error, adapting the model parameters accordingly. This is the case for regression and classification algorithms. Unsupervised learning algorithms make predictions on unlabeled data. They are used in cases where there is no ground truth for the input data and the algorithm aims to learn its characteristics to perform some given task. This is the case for clustering algorithms. Figure 3.1 illustrates the differences between the supervised and unsupervised learning algorithms.

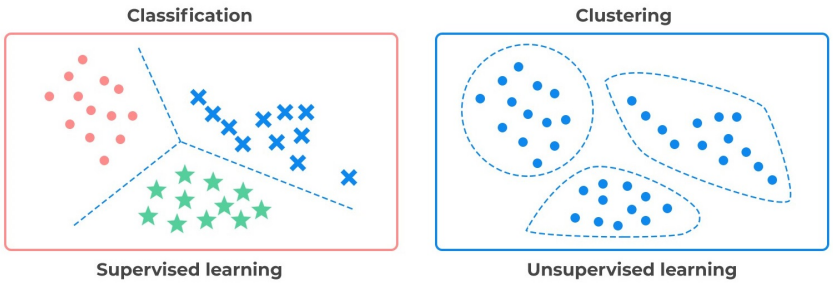


Figure 3.1: Illustration of the difference between supervised (left) and unsupervised learning (right) algorithms. The data on the left is colored, indicating the labels of each data point. The algorithm has the task of classifying each point to its respective label. On the right, the data is unlabeled, thus the task of the model is to aggregate similar points in groups, forming clusters. Source: [4]

The process of fitting ML algorithms to the data involves two steps: training and testing. As stated above, the objective of ML is to learn patterns from the data in order to make predictions on unseen data. Therefore, to correctly evaluate the quality of an ML algorithm, the model must be tested with data with which it was not fitted. The common practice is to split the data into a train set, on which the data is fitted, and a test set, on which the data is evaluated. The error in prediction in the training set is called *training error* and in the test set is called *test error*.

An important concept for training ML models is the observation of the bias-variance trade-off. *Bias* is the difference between the average prediction of a model and the value of ground truth. A high bias model is a model that cannot learn from training data, giving results that are far from the expected value. *Variance* is the variability of a model in relation to the training data. A model with high variance is perfectly fitted to the training data, following the trend of the values as closely as possible. However, it might not perform well on unseen data, which follows a slightly different pattern. Figure 3.2 illustrates the concepts of bias and variance.

When a model presents high bias in the training data, it is said that the model is *underfitting*. Underfitting indicates that a model was unable to learn from the training data. In most cases, models that underfit are not complex enough to explain the patterns in the data. On the other hand, when a model presents low bias in training data but high variance, the model is said to be *overfitting*. Overfitting indicates that the model fitted the training data too well, meaning that it was unable to generalize for unseen data, leading to poor performance on the test set. Figure 3.3 shows a visual example of overfitting and underfitting models.

3.2 Performance Metrics

To learn, a ML algorithm needs a performance metric to optimize the task at hand. Performance metrics can be classified into two different groups: classification and regression metrics. Classification problems aim to predict a noncontinuous target vector, calculating the probability of each

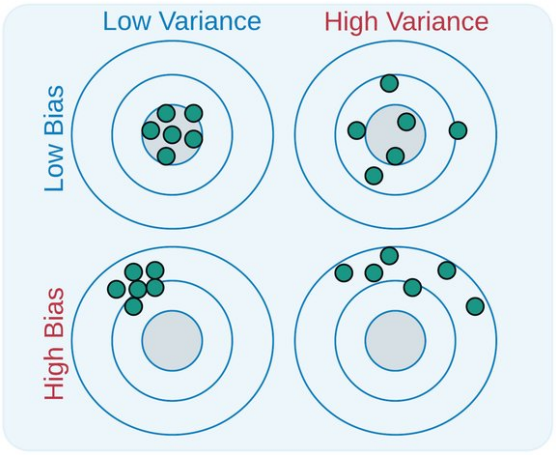


Figure 3.2: Bias and variance illustration using a bullseye diagram. Bias relates to how close to the center the points are, whereas variance relates to the dispersion of points around the center. Source: [5].

observation belonging to each target class and predicting it as belonging to the class with the highest probability. Classification metrics quantify how much the predictions were equal to the ground truth labels. On the other hand, regression problems predict a continuous target vector by minimizing the error between the predicted and ground-truth labels, which is quantified by a regression metric.

This work will focus on regression metrics due to the nature of the problem at hand. The most used regression metrics are the Mean-Squared Error (MSE) and Root-Mean-Squared Error (RMSE), given by the equations below:

$$MSE = \sum_{i=1}^{D}(x_i - y_i)^2 \tag{3.1}$$

and

$$RMSE = \sqrt{\sum_{i=1}^{D}(x_i - y_i)^2}, \tag{3.2}$$

where D is the size of the label vector and x_i and y_i are the true and predicted individual values.

3.3 Neural Network

Neural networks are ML algorithms that attempt to mimic the behavior of human neurons by applying nonlinearities to the underlying matrix operations. Samarasinghe defines neural networks as being "a collection of interconnected neurons that incrementally learn from their environment (data) to capture essential linear and non-linear trends in complex data" [52].

A *neuron*, also called *perceptron*, is the unit of processing in a neural network. The neuron is responsible for applying a matrix operation on the input data and passing the result through a

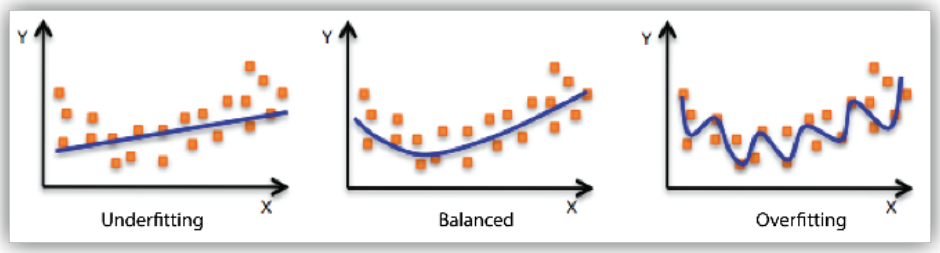


Figure 3.3: Examples of overfitting, underfitting and balanced fitting on a regression model. The trend of the data is similar to a quadratic function. Fitting a linear function to the model is too simplistic and does not represent the trend of the data points, thus being a case of underfitting. However, by fitting a complex function of high order in order that touches every point in the data fails to generalize, thus configuring overfitting. The ideal fit is a line that follow the trend of the data, but without attending perfectly to every data point. Source: [6]

nonlinearity, called a *activation function*. A collection of neurons connected in parallel is called a *layer* of a neural network. When every data point in the input is connected, i.e. goes through the matrix operation followed by the activation function, to every neuron in a layer, the layer is said to be *fully connected* (FC).

A deep neural network is made up of *input layer*, composed of input data, followed by one or more FC layers, called *hidden layers*, followed by the last layer, *output layer*, which maps the model output to the desired target value. A deep neural network is called a feedforward network (FFN) when the connections between neurons do not form a cycle, i.e. information flows in a single direction. Figure 3.4 shows a graphical example of a FC FFN.

The learning process follows two steps: forward and backward propagation. Forward propagation is the operation of passing the input data through the neural network to obtain the predicted output. Backward propagation is the process of updating the model's weights by propagating the prediction error through the network. Next, we will discuss the forward and backward propagation methods, and we will present and discuss the essential activation functions and optimization algorithms used in the learning process. Then, we will discuss two crucial neural network models used in the VQA.

3.3.1 Forward propagation

The forward propagation equation of a single perceptron, given the input vector X, the weight vector W, the bias vector b, and the output vector Y is given by the equation below:

$$Y = X \cdot W + b. \tag{3.3}$$

The output of the layer passes as input to a non-linear activation function. The most common nonlinear activation functions in neural network applications are the Rectified Linear Unit (ReLU), the sigmoid, and the softmax functions. However, for most SOTA applications of the transformer

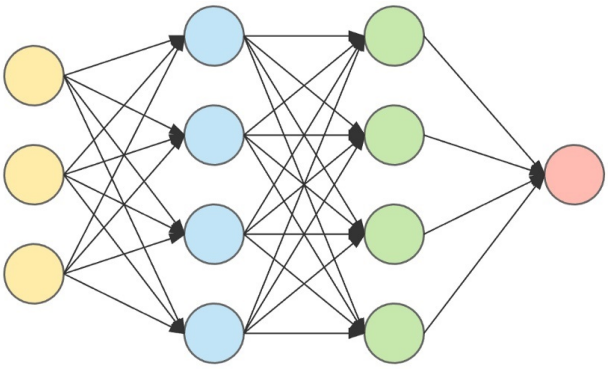


Figure 3.4: Illustration of a fully-connected neural network model composed of layers of perceptrons. The model includes an input layer (yellow), two hidden layers (blue and green) and an output layer (red). Source: [7]

architecture, ReLU activation is replaced by the Gaussian Error Linear Unit (GELU) [53] activation function [54, 55, 56].

The ReLU function is a nonlinear function that maps values between 0 and z, where z is each value in the vector, thus eliminating negative values from the input. It is efficient since not all neurons are activated. Thus, it requires less memory and provides a better generalization of the network [57], which is the reason why it is widely used in hidden layers of the models. It is given by the equation below:

$$ReLU(z) = max(0, z). \tag{3.4}$$

The GELU is a modification of the ReLU function, by combining characteristics of the ReLU, *dropout* [9] and *zoneout* [58]. ReLU deterministically multiplies the input value by 0 or 1, while dropout and zoneout stochastically multiply the input value by 0 and 1, respectively. Merging the three functionalities, the input value in the GELU activation function are multiplied by 0 or 1, with the zero-one value being stochastically determined by the cumulative distribution function of the Gaussian distribution, but also dependent on the input. By taking the expected value of that transformation, in order to obtain a deterministic value, the GELU function is given by the equation below:

$$\begin{aligned} \text{GELU}(x) &= xP(X \leq x) = x\Phi(x) \\ &\approx 0.5x\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right) \end{aligned} \tag{3.5}$$

Figure 3.5 shows a plot of the GELU and ReLU activation functions.

Sigmoid is the most used activation function and maps the values between 0 and 1 with an S-shaped curve. It is normally used as the last activation function in binary classification problems. It produces a value between 0 and 1 that can be interpreted as the probability that the observation belongs to the positive class. It is given by the equation below:

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3.6}$$

Figure 3.6 shows a plot of the sigmoid activation function.

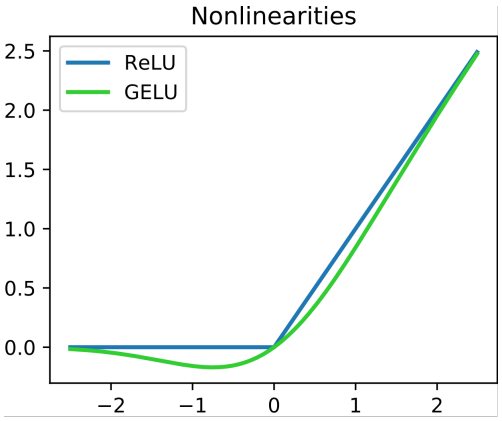


Figure 3.5: Comparison between ReLU and GELU activation functions. Source: [8]

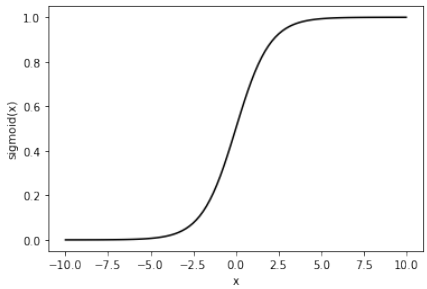


Figure 3.6: Illustration of the sigmoid activation function.

The softmax function combines multiple sigmoids, which is why it is used as the activation function for the output layer of multi-class classification problems. It is given by the following equation:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots K, \tag{3.7}$$

where K is the number of classes in the problem.

3.3.2 Backward propagation

After every step of forward propagation, the predicted output is compared with the ground-truth values. If the predicted output differs from the true values, a loss function is used to measure the error. Backward propagation uses that loss function to update the weights of the layers with the aim of minimizing that loss.

There are many loss functions that are used for different applications. The MSE, given by equation (3.1), is widely used in regression cases, for example. For classification, the most common loss used is the cross-entropy loss, given by the equation below:

$$\text{Loss} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}), \tag{3.8}$$

where M is the number of classes, $y_{o,c}$ is 1 if o, classified as of class c, was correctly classified, 0 otherwise, and $p_{o,c}$ is the predicted probability that observation o belongs to class c.

The loss is compensated by propagating it through the layers. In order to do that, we calculate the contribution each layer had in the overall loss. Since the layers are connected in series, the way to calculate that error is by differentiating it using the chain rule. In this way, the error is given by:

$$J\left(\theta_{t,k}\right) = \frac{\partial E}{\partial \theta_{t,k}}, \tag{3.9}$$

where θ are the network weights, k is the index of each parameter, and t is the training time step.

The last step in backward propagation is to apply the gradient descent. Gradient descent is the process of minimizing a function by updating its parameters in the opposite direction to its gradient [59]. Therefore, the gradient descent of a function $J(\theta_{t,k})$ is given by:

$$g_{t,k} = \nabla J\left(\theta_{t,k}\right), \tag{3.10}$$

where η is the learning rate parameter, which determines the size of the steps taken to reach the local or global minimum. A widely used gradient descent algorithm is the Adaptive Moment Estimation (Adam) optimizer [60], which uses an exponentially decaying average of past gradients and past squared gradients to smooth the process of updating the learning rate of the models, which determines the size of the steps taken to reach the local or global minimum.

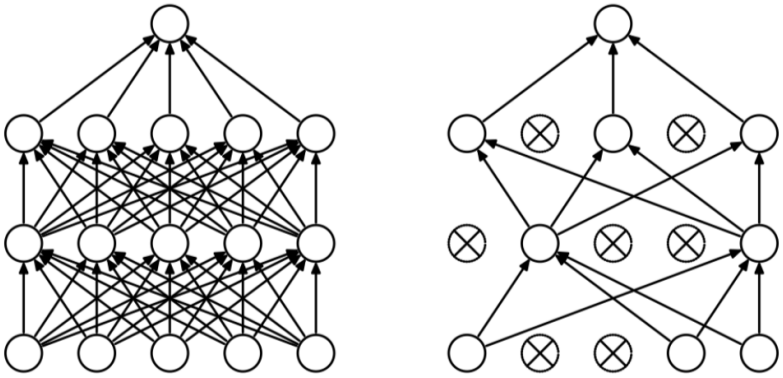
3.3.3 Regularization

A recurring problem in ML is training a model capable of performing well on unseen data rather than just on the data it was trained on. Many tools have been designed to reduce the test error, even if it means harming the performance of the model on the training data. These tools are known as regularization. Goodfellow [61] defines regularization as 'any modification we make to a learning algorithm that is intended to reduce its generalization error, but not its training error'.

Some regularization techniques focus on constraining or even eliminating the learned parameters of the model, preventing it from focusing on specific characteristics of the training data and rather learning more general patterns, improving generalization, and consequently test performance [61]. Other algorithms apply regularization by modifying model parameters using general statistics of the model, reducing its variance but maintaining their underlying characteristics so as not to increase bias considerably. The most common regularization methods applied to Transformers are the *dropout* [9] and *layer normalization* [62] techniques [15][63][64].

The dropout method consists of randomly dropping neurons into a neural network by setting their values to 0. Basically, at each training iteration a neuron is kept with a probability of p, independent of the other neurons. The concept originates from the idea that the best way of regularizing a model is by averaging the predictions of every possible combination of model parameters. However, since this is computationally expensive, by randomly dropping values in each training iteration, a large collection of different models with shared weights are trained with little computational cost [9]. Figure 3.7 illustrates the dropout procedure in a FC FFN.

Layer normalization (LN) is the process of normalizing the weights of a neural network using the mean and variance statistics of the summed inputs to the neurons of a hidden layer [62]. The



(a) Standard Neural Net (b) After applying dropout.

Figure 3.7: Illustration of the dropout procedure in a FC FFN. On the left, a standard FC FFN, without any dropped units. On the right, the resultant net after randomly dropping units. Crossed units are the dropped neurons. Source: [9]

method was proposed as an alternative to the previously established batch normalization (BN) technique [65], which was proposed by normalizing the weights with the mean and variance of each mini-batch of training, reducing the training time and improving the generalization of the model. However, since BN is dependent on the input batches, for sequential data, where the input to the model is not always of same size, applying normalization based on the layer parameters rather than the batch parameters proved to be more efficient.

3.4 Convolutional Neural Network

CNN is a type of neural network that is widely used in image processing, which contains *convolutional layers* as part of the network. A convolutional layer is a special layer that applies the convolution operation between an image patch and the network filters. Figure 3.8 illustrates the CNN architecture.

Convolutions can extract specific features from an image, depending on the constitution of the filter weights. For example, using a Laplacian filter, we can extract the edges of an image, depending on the orientation of the filter parameters [66]. Since weights in a CNN layer are learned through backward propagation, the network can optimize the convolutional layer to focus on the most important features for the specific task at hand.

Convolutional layers are commonly followed by a *pooling* layer. The pooling operation summarizes the input by the statistical characteristics of the input. Thus, a pooling layer will reduce the input layer to a summary statistic. The most widely used pooling strategy is the *max pooling* operation, which substitutes the input vector for its maximum value. Figure 3.9 shows an example of the pooling operation.

Pooling has two important benefits for CNN performance: computational cost and generalization. First, by reducing the output of the convolutional layer, we reduce the number of learnable

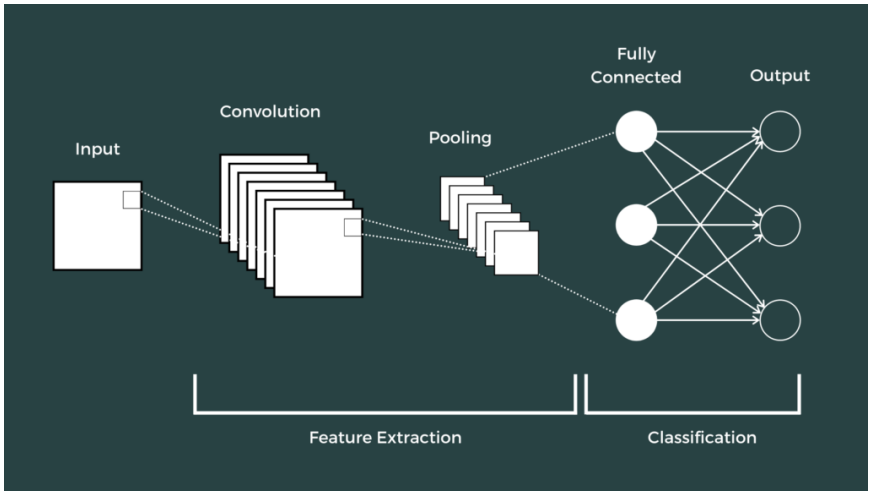


Figure 3.8: Structure of a basic CNN. The convolutional layer has multiple filters, each one applying the convolution operation on the input image. The output of the convolutional layer is then passed through the pooling layer, which will summarize the input with its statistical characteristics. The output of the pooling layer is passed through a FC layer to map its values to the desired target. Source: [10]

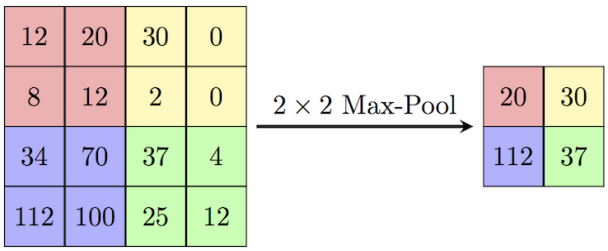


Figure 3.9: Example of the max pooling operation. The value of every $2x2$ region is replaced by the maximum value in the region. Source: [11]

parameters in the model, hence reducing the memory and time needed for training the model. Second, and most importantly, by substituting the input vector with its statistical summary, we make the output less sensitive to minor variations in the input, thus better generalizing the model performance. This is essential in any ML algorithm since the main goal is to make it reproducible to unseen data.

3.5 Recurrent Neural Network

RNN is a type of neural network that processes sequential data, such as time series, textual data, audio samples, and video frames [61]. In RNNs, the input to each time step is combined with the output of the last time step, called *hidden states*, for prediction, that is, keeping the temporal information throughout the weights of the network. This type of model is called an *autoregressive* model, which consumes previously generated output as input to predict future outputs. The equations for a time step are the following:

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \tag{3.11}$$

$$\boldsymbol{h}^{(t)} = \tanh\left(\boldsymbol{a}^{(t)}\right), \tag{3.12}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \tag{3.13}$$

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{o}^{(t)}\right), \tag{3.14}$$

where $x^{(t)}$ is the input of time t, $h^{(t)}$ is the hidden state of time t, $\hat{y}^{(t)}$ is the output of time t, c and b are bias vectors and W, U, and V are weight vectors.

There are three types of RNN configurations: one-to-many, many-to-one, and many-to-many. In one to many, the input is a single vector, and each time step outputs a predicted value, which serves as input to the next time step. This configuration is found on text, music and video generation tasks, for example. Figure 3.10 illustrates a one-to-many RNN configuration.

In many-to-one, the input is sequential data and the output is a single value. On each time step, a data point is inputted, and a hidden state is passed to the next time step. The model only outputs a value in the last time step. This configuration is present in text or video classification tasks, for example. The latter is the focus of this work since the task is to classify sequential audiovisual data into a single quality value. Figure 3.11 illustrates a many-to-one RNN configuration.

In many-to-many, both input and output are composed of sequential vectors. It has two possible configurations: (1) the output has the same size as the input, so that every time step outputs a predicted value and passes a hidden state to the next time step. This configuration is used in named entity recognition tasks, for example; or (2) different sizes, in which case values start to be outputted only after the last data value is inputted to the model. This configuration is used, for example, in machine translation tasks. Figures 3.12 and 3.13 illustrate a many-to-many RNN configuration for cases where input and output lengths are equal and different, respectively.

Variations of the RNN architecture have been proposed in the literature. A widely used one is the Bidirectional Recurrent Neural Network (BiRNN) [67], a model that reads inputs in both

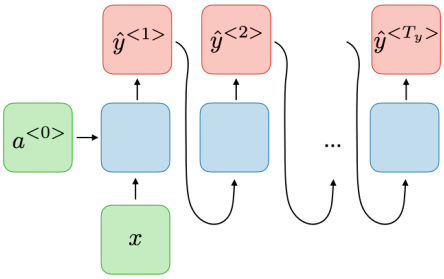


Figure 3.10: Schematics of a one-to-many RNN. The data serves as input to the first recurrent unit. Each time step outputs a predicted value, which serves as input to the following time step. Source: [12]

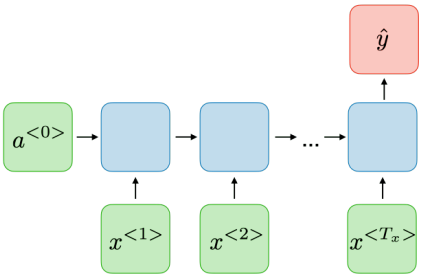


Figure 3.11: Schematics of a many-to-one RNN. The data values serves as input on each time step, and a hidden state is passed to the next time step. The model only outputs a value on the last time step. Source: [12]

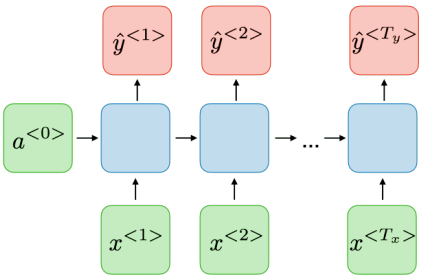


Figure 3.12: Schematics of a many-to-many RNN. The data values serves as input on each time step, and a hidden state is passed to the next time step. Input and output have same length, so each time step outputs a prediction value. Source: [12]

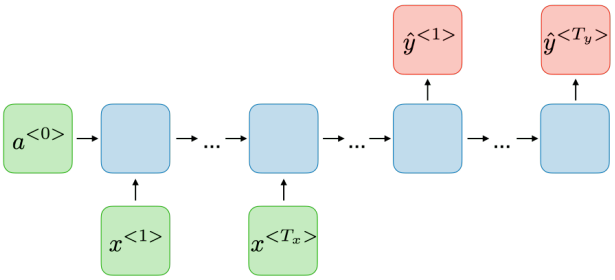


Figure 3.13: Schematics of a many-to-many RNN. The data values serves as input on each time step, and a hidden state is passed to the next time step. Input and output have different lengths, so values start to be outputted after the last input data is passed. Source: [12]

directions to retain information from both the past and the future. Another extensively used is the Long Short-Term Memory (LSTM) network [68], which addresses the problem of vanishing gradient on long sequences. It uses gates to control the information passed between different time steps using a set of new weight vectors to avoid the problem of vanishing gradients. This is done through a new *cell state* output and a *forget gate*, which will contain information on the long-term dependencies between the input vectors.

3.6 Transformers

The Transformer is a network architecture proposed by Vaswani *et al.* [15] that relies solely on the attention mechanism to solve the encoder-decoder task. It was proposed as an alternative to previous models that connected the encoder and decoder parts through an attention model, although still using recurrent or convolutional layers as part of their architecture [14, 69]. The attention mechanism is explained below, followed by a detailed explanation of the Transformer architecture.

3.6.1 Attention mechanism

Many applications with sequential data include the *encoder-decoder* architecture, for example, neural machine translation (NMT), image caption generation, video description generation, and end-to-end neural speech recognition [70]. In this architecture, a set of input vectors is encoded into a single vector containing their information, and then the decoder uses it to generate the desired output sequence.

Several studies proposed RNN-based models for the encoder-decoder architecture. An RNN encodes the input sequence into a single vector, and a second RNN decodes it into the output vector. However, RNNs have two problems for this type of application: first, it is computationally expensive since the data are passed sequentially, keeping it from being parallelizable; and most importantly, for long sequences, it is challenging to capture the temporal relations into a single fixed-size vector.

Because of that, as an alternative to using RNNs in the decoder part, Bahdanau *et al.* [14] uses an *attention mechanism* instead for the Neural Machine Translation application. The idea is to produce, for each word generated in translation, a set of attention weights for each word in the input sentence, representing the importance of each word in the translation of the current output vector. Instead of encoding the input vector into a single fixed-size vector, it produces a sequence of vectors and chooses some of them to produce the output. Figure 3.14 illustrates the mechanism of attention for NMT.

The decoder is an RNN that uses the context vectors of the encoder and the past values of the decoder as input to each time-step computation. The context vector is obtained by a weighted

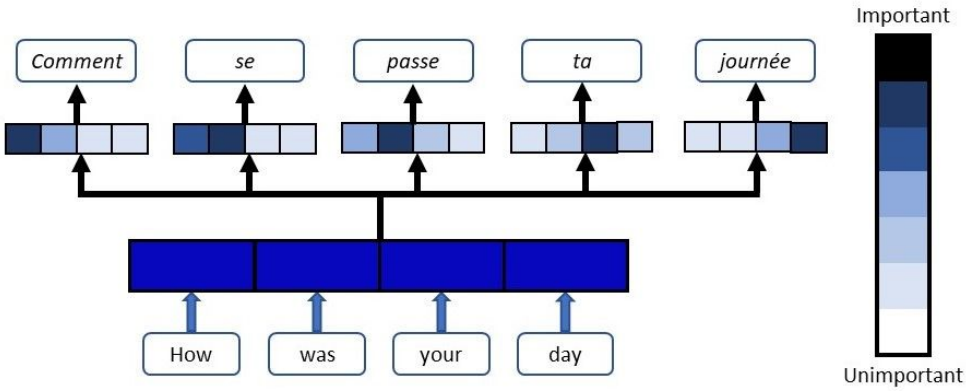


Figure 3.14: Illustration of attention mechanism in an English-French translation application. The colors indicate the attention weight of each input word, representing how much that word influences the translation of each output word. Source: [13]

sum of the hidden states outputted by the encoder and is given by:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j, \tag{3.15}$$

where T_x is the length of the input sentence and α_{ij} is the weight of attention of the input at position i in relation to the output at position j, and is given by:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \tag{3.16}$$

where $e_{ij} = a(s_{i-1}, h_j)$ is the *alignment model*, which determines the level of match between the inputs around the position j and the output at the position i. Figure 3.15 shows the building blocks of the proposed attention-based NMT algorithm.

The attention function a differs in different applications. For example, the attention function presented at [14], commonly referred to as *additive attention*, is a neural network of forward propagation given by:

$$a\left(s_{i-1}, h_j\right) = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j\right), \tag{3.17}$$

where v_a, W_a, and U_a are weight matrices.

Other frequently used attention functions, referenced as *dot-product attention*, proposed by Luong *et al.* [71], are given by:

$$a\left(s_{i-1}, h_j\right) = s_{i-1}^T W h_j, \tag{3.18}$$

where W is a weight matrix. The dot-product attention is used in the construction of the Transformer architecture, explained in the next section.

3.6.2 Transformer architecture

The Transformer architecture is shown in Figure 3.16. The proposed method applies *self-attention* to the input and output sequences. Self-attention is an attention mechanism that computes a representation of a sequence by relating different positions in it. The encoder is made up

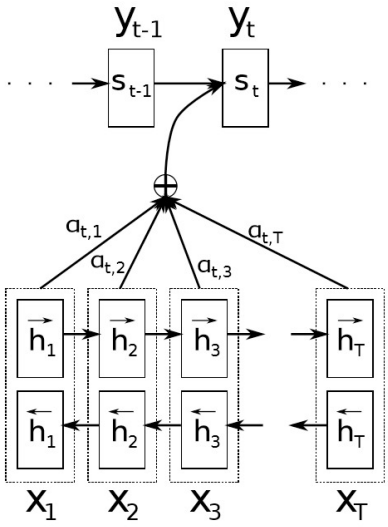


Figure 3.15: The proposed attention-based model, computing the output y_t in time step t given input vector $(x_1, x_2, ..., x_T)$. The encoder is a bidirectional RNN, which generates a set of attention weights used in the decoder to predict the words in the translated sentence. Source: [14]

of a *Multi-Head Scaled Dot-Product Attention*, followed by an FFN, followed by an added residual connection and layer normalization. The decoder has the same architecture, with the addition of a *Masked Multi-Head Scaled Dot-Product Attention*, where the values after the current position are masked so that the output is dependent only on the previously known values in a sequence. This configures the Transformer model as an autoregressive model.

The attention function used is a Multi-Head Scaled Dot-Product Attention, shown in Figure 3.17. The model concatenates six different attention functions applied to the input data to make the model learn different representations from the input sequence simultaneously. A single scaled dot-product attention function is given by:

$$\text{Attention}(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3.19}$$

where Q, K are queries and keys of dimension d_K, and V are values of dimension d_V.

Another important feature present in the transformer architecture is *positional encoding*. Since the model does not use recurrence or convolution, passing the inputs simultaneously, positional encoding injects information about the order of the elements in the sequence. The positional encoding functions used are sine and cosine functions with different frequencies:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\text{model}}}\right), \end{aligned} \tag{3.20}$$

where *pos* is the position of the element and i is the dimension.

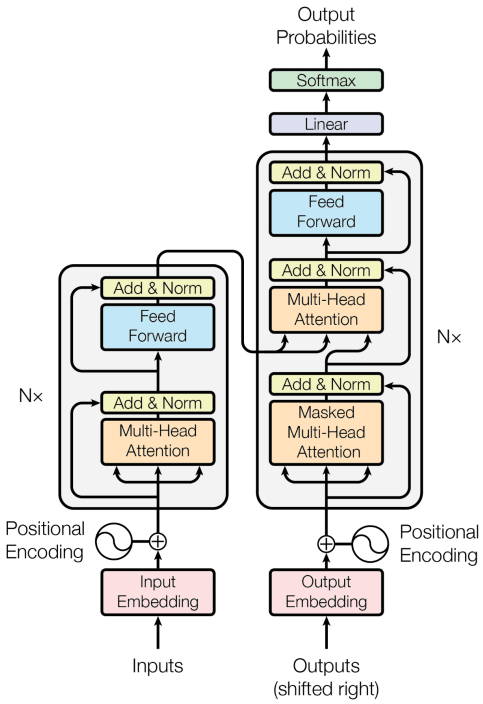


Figure 3.16: The Transformer architecture. Left-side corresponds to the encoder and right-side to the decoder. The input sequence is fed to the encoder, and the desired output sequence is shifted right and fed to the decoder. The output of the decoder if fed to a linear plus softmax layers to generate the next-token probabilities. Source: [15]

3.6.3 Transformer in Computer Vision

Given the extensive use of transformers in NLP tasks, given their ability to compute temporal relationships with considerably less computational complexity and more parallelization, Computer Vision tasks lately have adopted the Transformer for various tasks, such as object detection [72], image generation [73], and image recognition [24].

Different approaches were taken in using transformers for image-related tasks. Many models rely on CNN to extract features from images, which are then processed by the Transformer [74], while others, called Visual Transformers (ViT), use image patches directly in the Transformer [24]. Transformer has also been an active field of research for video processing tasks, such as video classification [75] [76], video retrieval [77], and future action anticipation [78].

Transformers were also used for IQA tasks, both for scenarios FR [79] and NR [64, 80]. However, for VQA, the Transformer architecture has yet to be exploited. Recent work by You *et al.* [81] used a Long Short-term Convolutional Transformer (LSCT) for the prediction of the NR video quality.

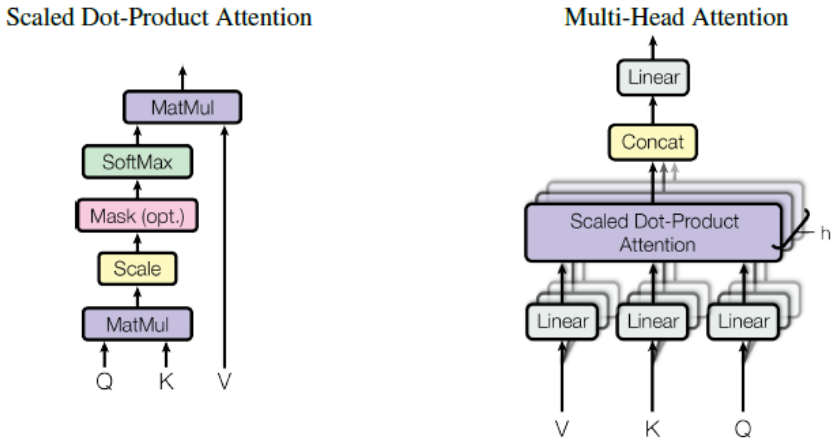


Figure 3.17: (left) Scaled dot-product attention. (right) Multi-Head Attention, with multiple attention heads running in parallel. Source: [15]

Chapter 4

Proposed Methodology

This chapter presents the methodology applied in this project. First, the the architecture of the model is detailed; second, the database used for training and testing is described; third, the feature extraction process is explained; fourth, the process of resampling the video features is disclosed; last, the training process is explained.

4.1 Model architecture

We used the Transformer architecture to predict the quality score of the signals under analysis. As stated before, the Transformer architecture has the capacity of extracting meaningful information from the input features, given the ability of the self-attention layer to identify important relationships between them in order to correctly shape the desired output from the network. In this work, we used only the encoder part of the Transformer followed by a classification layer, since the decoder is specific for sequence-to-sequence tasks.

The input AV features pass through an embedding layer, composed of a linear layer followed by a non-linear activation function, which maps the input vector to a different dimension in order to learn better representations of the input data. Then, a positional encoding, described by equation (3.20), is applied to the embeddings, to capture temporal information between the AV signal samples. The encoded embeddings are then passed through the Transformer encoder, illustrated to the left of the architecture shown in Figure 3.16. The output of the transformer encoder is processed to attend to the specification of each training configuration explored, explained in further detail in the next section. An illustration of the model architecture is shown in Figure 4.1.

4.2 Database

In order to effectively train a model to predict the AVQ of streaming multimedia content, the training data must contain different types of degradation present in streaming scenarios. Thus, the dataset used in this project is Martinez's UnB-Audiovisual Database [30], specifically Experiment

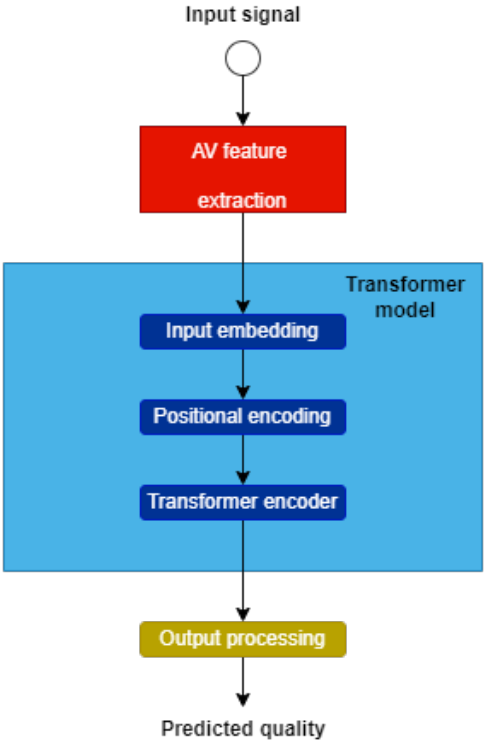


Figure 4.1: Transformer for AVQ prediction diagram. The AV features of the input signal are extracted and passed to the Transformer model, composed of the embedding layer, the positional encoding layer and the Transformer encoder model. Finally, the output values are processed to generate the predicted quality of the video.

3, described in [82], which presents both audio and visual distortions.

The database was constructed with a psychophysical experiment, where 40 human participants rated a set of 40 high-definition AV sequences. The experiment used an immersive methodology, proposed by Pinson at [28], which presents signals in a more natural setting to better capture the perceived quality of the AV sequences. The participants rated the video sequences on a five-point Absolute Category Rating (ACR), ranging from 1 to 5. The mean of the ratings of the participants who rated each signal, defined as the mean quality score (MQS), was assigned as the subjective quality score of the video.

The original data set, composed of 40 video sequences, was processed by adding 16 different combinations of audio and visual distortions, plus an additional 4 anchor testing conditions, resulting in 800 video sequences. The visual distortions are bitrate compression, packet loss, and frame-freezing. Bitrate compression refers to the losses applied resulting from compressing the video data. Packet loss is the phenomenon where a packet fails to reach the destination during transmission. Frame-freezing is experienced in live transmission scenarios, when the required data is not available for download, freezing the video until enough data is available.

The audio distortions were background noise, chop, clip, and echo. Background noise refers to

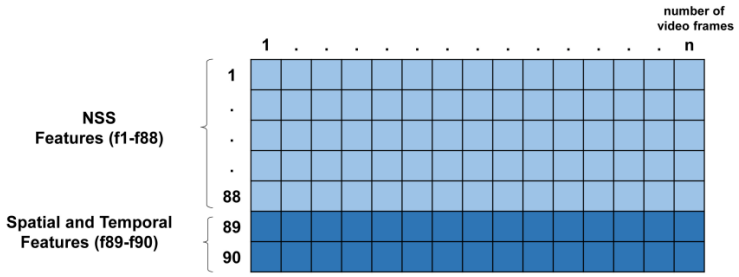


Figure 4.2: Visual Feature set, composed of 88 NSS features and 2 temporal features for each video frame. Source: [1]

any sound that is not the sound being studied, such as traffic noise, people talking and rain. Chop refers to audio signals with missing samples due to errors in transmission. Clip refers to when a transmitted signal's amplitude is higher than the maximum level permitted. Echo occurs mostly in voice calls when the transmitted audio is sent back to its origin, causing a feedback.

4.3 Feature extraction

The aim of this work is to train a model capable of predicting the quality of a video signal using audio and video signals simultaneously, with the aim of learning the important relationships between them that shape the perceived QoE. To efficiently combine audio and visual signals, the chosen approach was to use a set of extracted features for each signal type, then combining them to form the input audiovisual feature matrix. The features used in this work were obtained using the feature extraction technique presented by Martinez [1]. The visual and audio feature sets are briefly explained below, as well as the process of combining them to form the audiovisual feature set.

4.3.1 Visual features

Natural scenes have statistical properties that are altered in the presence of distortion. Thus, using NSS as input to a model allows it to identify those characteristics that indicate and quantify the distortion present in a signal. Besides, NSS are not specific to any distortion type, thus being a good choice for scenarios with different kinds of distortions.

The proposed model uses 88 NSS features, extracted using the feature extraction function from the Diviine algorithm implementation [39], resulting in a n-by-88 array for each video sequence, where n is the number of video frames. Furthermore, we use temporal features extracted using the algorithm presented by Ostaszewska and Kloda [83], resulting in a n-by-2 array. By merging both feature matrices together, we obtain the set of n-by-90 visual features for a single video sequence. Figure 4.2 illustrates the set of visual features.

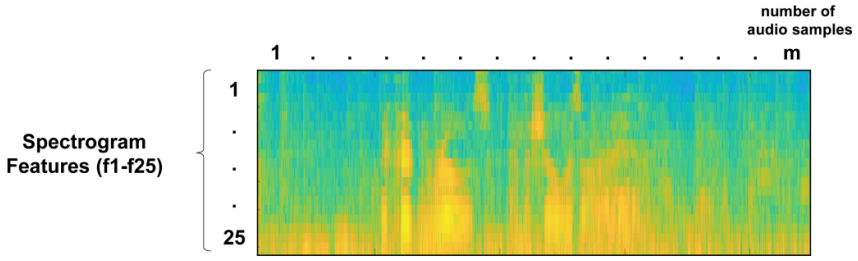


Figure 4.3: Audio Feature set, composed of 25 frequency bands of the spectrogram of the audio signal for each audio sample. Source: [1]

4.3.2 Audio features

In order to obtain a set of features capable of describing the distortions in the audio signal, a spectrogram representation is used as input to the proposed model. Spectrograms are widely used in the literature for audio classification [84] and speech quality assessment [42]. The spectrograms were extracted using the feature extraction function presented in the Visqol implementation [85], resulting in a m-by-25 array of features for each video sequence, where 25 is the number of frequency bands and m is the number of audio samples of the signal.

4.3.3 Audiovisual features

Since the objective of this model is to explore the relationship between audio and visual signals in the composition of the perceived quality of AV signals, both sets of features are merged, resulting in a larger set of AV features. However, to achieve this, a processing of both sets is necessary, as the number of audio samples and video frames of an AV signal is frequently different. The strategy used to match the sizes of both sets was to replicate the values in the smaller set to match the size of the larger set. Since there are fewer video frames than audio samples in a video sequence ($n < m$), the visual features are replicated to match the audio features, as illustrated in Figure 4.4. Finally, the processed visual (rows 1-90) and audio (rows 91-125) feature arrays are merged to form the final m-by-115 AV feature matrix. From this point forward, each element in the AV feature matrix will be referred to as AV sample.

4.4 Videos resampling

The sampling rate of the videos, after matching the audio and visual features with the procedure illustrated in Figure 4.4, is around 20 Hz. Given that we have a high sampling rate, we observed that the features extracted from each video sample had high auto-correlation. That is, the features of the AV samples close to each other had little variance compared to those of the AV samples around them. That could possibly hurt the ability of the model to learn the temporal relationship between the AV samples, since most of them are very close to each other.

Because of that, we experimented with two configurations of the input signals. First, we

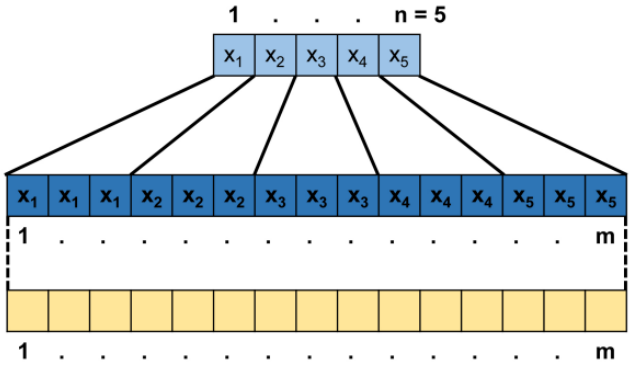


Figure 4.4: Illustration of the procedure to match audio and visual features arrays. The video frames are replicated in order to match the size of the audio samples [1]. Source: [1]

trained the model with the original data, with all features extracted from the video AV samples. Second, we resampled the videos by eliminating AV samples of the video, reducing their effective sampling rate. To estimate the rate at which we should remove the AV samples, we plotted the Autocorrelation Function (ACF), which calculates the correlation between a time series, which in this case is the video AV samples, with its own version lagged from 0 to p AV samples [86]. By doing that, we can identify how many AV samples in the video sequence are highly correlated with its neighbors, allowing us to determine the rate at which to remove AV samples.

The ACF plot for 4 random video sequences is presented in Figure 4.5. To analyze the graphs, we must observe the blue area, which consists of the confidence interval 95%. That is, anything inside the blue area is statistically close to zero, which means that there is no significant autocorrelation. Given that, we can observe the highest lag value outside the blue area, which indicates the number of AV samples that have a high autocorrelation with its neighborhood, indicating the sampling rate to use in eliminating AV samples. The results of our experiment indicate that the rate is 30 AV samples; thus, every 30 AV samples of the video, we eliminate 29 and keep just the last one.

4.5 Training

The model was trained using the Python programming language, with the Pytorch module for the architecture and training of the Transformer neural network. For training and evaluation, we use the K-fold cross-validation technique. In K-fold cross-validation, we divide the dataset into K equal-sized parts, and perform K iterations of training-evaluation. In each iteration, one part is used as the test set and the other $K-1$ parts are combined to form the training set [51]. The test score is the average of the test scores in the K folds. In this work, we use a 10-fold cross-validation, with each fold consisting of 80 video sequences. Using K-fold cross-validation is essential to ensure that test results are not influenced by the random factor of splitting the dataset into training and test sets, which could yield unreliable and different results given different splits of the dataset. The K-fold cross-validation method avoids that by allowing the model to learn and be tested with

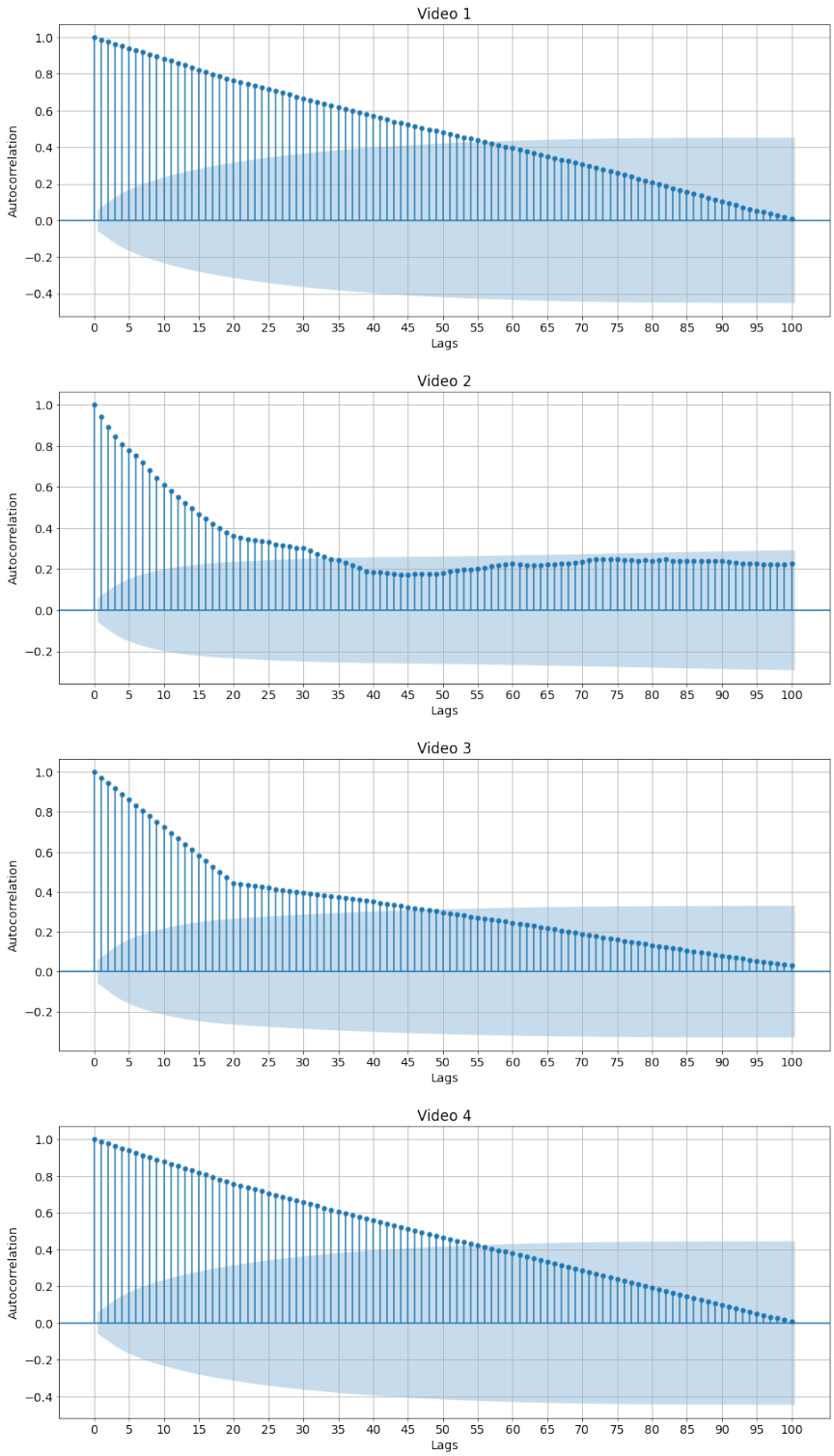


Figure 4.5: ACF plots of 4 random video sequences from the UnB-Audiovisual Database. The blue area indicate the 95% confidence interval, where the correlation values are close to 0. From the plots, we identify that for all videos, up to 30 AV samples, the correlation is high enough that we can eliminate those AV samples without significant loss of information.

multiple configurations.

Four configurations were explored in the model training process:

1. Classification for sample prediction (CS): The model is trained as a classification problem, assigning class probabilities to each AV sample, processing the output probabilities to make predictions, and averaging the predictions of all AV samples of the video.
2. Classification for video prediction (CV): The model is trained as a classification problem, predicting the quality of the whole video as a single target score.
3. Regression for sample prediction (RS): The model is trained as a regression problem, predicting the quality of each AV sample and averaging the predictions of all AV samples of the video.
4. Regression for video prediction (RV): The model is trained as a regression problem, predicting the quality of the whole video as a single target score.

In addition to the different configurations of the model architecture to predict AVQ, we explored the impact of the audio and visual parts of the features on the performance of the model. Therefore, three different feature configurations were combined with the training architecture configurations: (1) using only 25 audio features to train the model; (2) using only 90 visual features; and (3) using the full set of 115 AV features. Combining the 4 training configurations, 3 feature configurations and 2 sampling configurations, we trained a total of 24 setups. The 4 training configurations and their respective input and target arrays are further explained in the following.

4.5.1 Training input

Besides the AV features extracted from the input signal, a target matrix was created using the quality scores of the videos. Those quality scores correspond to the MQS obtained from the subjective experiments explained in Chapter 2, since those correspond to the closest values from the ground-truth we can get.

Since the quality score is a single continuous value ranging from 1-5 associated with each video, a pre-processing was needed to predict the quality of each AV sample individually in the sample prediction approaches (CS and RS). To achieve this, the quality score of each video was replicated m times and a random Gaussian noise of mean 0 and standard deviation 0.1, forming a m-by-1 continuous target array. This approach considers that when a signal has a certain overall quality, we can consider the quality of each individual AV sample of the video to have a value close to the overall quality of the entire video, with slight random variations in time.

However, in the CV and CS approaches, before replicating the quality value of the video to the AV samples, we applied an additional pre-processing step. We added Gaussian noise to the video target, as in the regression approach, but then mapped the resulting value to four different categorical classes, corresponding to the four quality groups from the ACR scale, ranging from 1-4. For example, a target score of 1.52 is assigned to class 1, since 1.52 is in the range <1,2>.

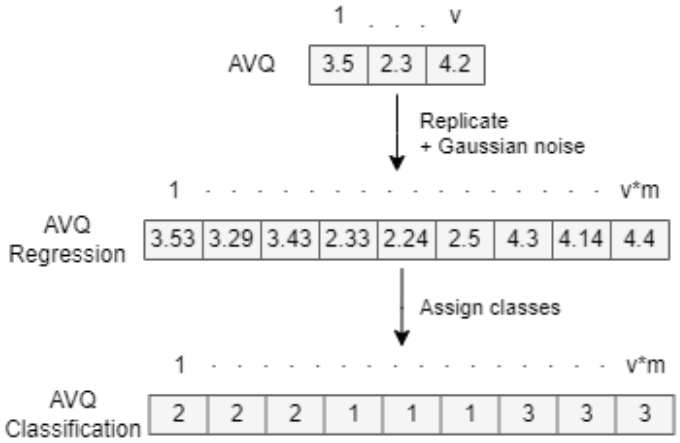


Figure 4.6: Illustration of the input pre-processing stage for the RS (top) and CS (bottom) approaches. The AVQ value of the video is replicated for every AV sample, with the addition of Gaussian noise. For CS, the values are assigned to categorical classes.

The class value is replicated for every AV sample of the input signal, forming a m-by-1 categorical target array. The array is finally subtracted by 1, transforming the classes into the range <0,3>, since in Pytorch models the class indexing must start at 0. Figure 4.6 illustrates the pre-processing procedure for RS and CS approaches.

For the CV and RV approaches, the AV features and target sets form a training feature matrix of dimension 720-by-m-by-115, and a training target matrix of dimension 720-by-1. Finally, we normalize the feature matrix and batch the data for training. For the sample prediction approach, the AV features and target sets of all training videos are concatenated, forming a training feature matrix, of dimension M-by-115-by-1 (M being the number of AV samples of all training signals), and a training target matrix, of dimensions M-by-1. The last dimension in the input feature matrix is added for linear operation in the embedding layer of the transformer. These matrices are then divided into batches for training. As with the video prediction approach, we normalize the feature matrix and batch the data for training.

4.5.2 Output processing

After passing through the encoder, the output signal has the same shape as the embedded representation of the input (720-by-m-by-D or M-by-115-by-D, where D is the embedding dimension). In order to make predictions for every input AV sample (either a video sequence or a sample of a video), we use a linear layer, followed by an activation function and a squeeze operation, to map the embedding dimension to a single value. Thus, the output of the model has dimensions 720-by-m or M-by-115. For the regression approach, we only need to append a linear layer with 1 neuron to that output, to map the second dimension to a single continuous prediction value.

However, for the classification approach, a linear layer with 4 neurons, followed by a softmax activation function, is attached to the output of the transformer encoder. The output of the

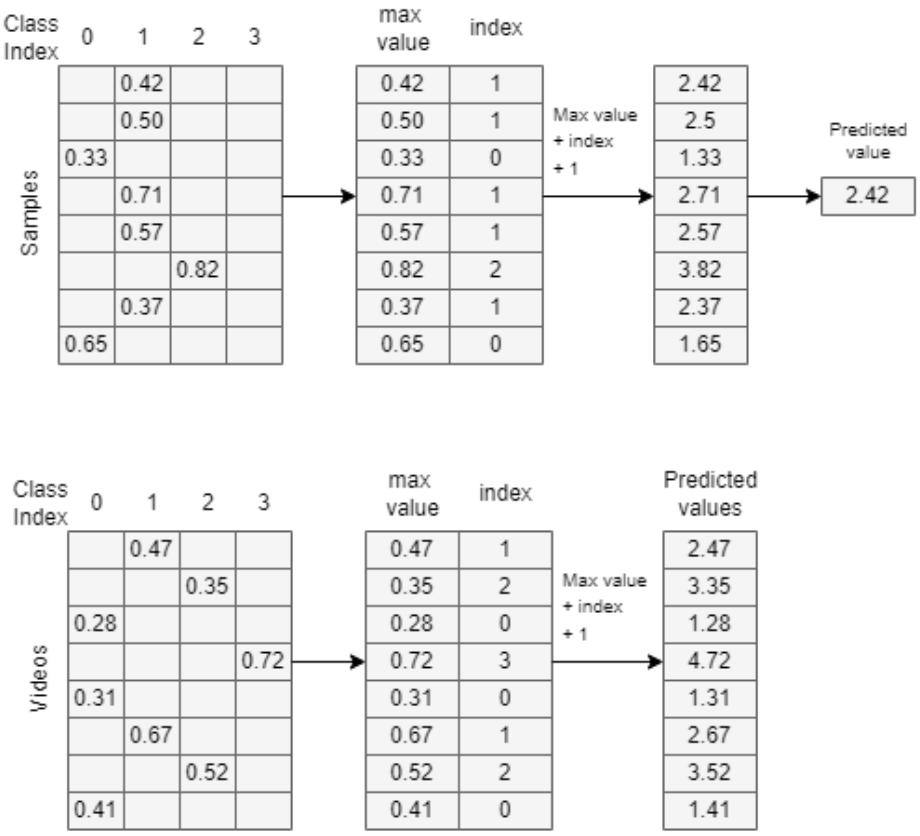


Figure 4.7: Illustration of the output processing stage for the CS (top) and CV (bottom) approaches. The output value is the combination of the class probability and the index of the most probable class.

model is either a M-by-4 or a 720-by-4 matrix, for the sample and video prediction approaches, respectively, indicating the probability that each AV sample or video belongs to a quality group. To assign a single predicted quality value to each input signal, the output of the model will be processed in a similar way to the procedure presented by Martinez [1]. First, we take the index of the highest probability in each row (AV signal sample) of the output matrix. Then we add 1 to that value to adjust to the range 1-4. Finally, we build the final prediction matrix by adding that value with the maximum value from each row, and assign to each video that value for the video prediction approach, or the mean of the quality scores of the video AV samples, for the sample prediction approach. This process is illustrated in Figure 4.7.

Chapter 5

Experimental Results

In this chapter we present the experiments made. First, we present the parameters and configurations used for training and testing the model. Then, we show the results for each combination of model and training data configuration. After that, we compare our best result with those obtained using other quality metrics from the literature, ending with a discussion of the results.

5.1 Training parameters

The Transformer encoder is specified by four key hyperparameters: L (number of layers),D (embedding dimension), H (number of attention heads) and d_{ff} (dimension of the feed-forward network). Other important parameters are the dropout and the activation functions used in the model. Since the input data given by the AV features from the UnB-Audiovisual Database are tabular and present high autocorrelation, the hyperparameters chosen to train the model form a shallow Transformer network, aiming to prevent overfitting the training data.

We used different configurations for sample predictions and video prediction approaches. Since in the sample prediction the last dimension of the input training matrix is 1, the embedding layer will expand the dimensionality of the input data rather than shrink it, as is in the case of video prediction. Because of that, the network for sample prediction needs to be shallower, in order to avoid overfitting. The hyperparameters used for each configuration are displayed in Table 5.1. For all cases, we used $dropout = 0.1$ and the GELU activation function. We used the MSE Loss for RV and RS training and the Categorical Cross-Entropy Loss for CV and CS training. In all cases, we used Adam optimizer with an initial learning rate of 0.001, using a learning rate scheduler that reduces the learning rate by half every 5 epochs.

Table 5.1: Training parameters for each configuration

Configuration	D	H	d_{ff}	L
CV/RV	8	4	32	4
CS/RS	4	4	32	2

The model was trained using fewer videos to tune the hyperparameters. After analyzing the loss progression throughout the epochs, it was observed that the validation loss stagnated after 10 epochs, thus we only trained for 10 epochs. The model was trained in batches of 4 videos for the video prediction approach and 2 videos for the sample prediction approach. At the end of every epoch, the outputs of all batches were concatenated and the PCC, SCC, and RMSE metrics were calculated with the target matrix. The model was trained 10 times for each configuration, once for each fold of K-Fold cross-validation. The model weights and parameters were reset after every fold training, not taking advantage of the training of any other folder.

5.2 Training setups results

Table 5.2 shows the average values of RMSE, PCC, and SCC over 10 folds for each combination of training setup and selection of features used in this work. Values in bold represent the best value for each metric within each type of training feature (audiovisual, audio-only, and visual-only). Figures 5.1-5.3 show the box plots of the PCC and SCC values across the 10 folds for each feature selection.

When observing the results, it is noticeable that the video prediction approach outperforms the sample prediction approach in all tested configurations. This was expected, since the quality of a video is shaped by the combination of various factors throughout the entire video, including temporal variations in quality. Thus, considering the quality to be constant with slight variations due to random noise leads to a poor estimation of the overall quality of a video. We also observe that the strategy of resampling the input sequences to reduce autocorrelation was not efficient and yields results similar or inferior to the models trained with the full input data. Notwithstanding the reduced autocorrelation and redundancy resulting from downsampling input features, the loss of information led the model to a poorer performance.

We can also note that the regression approach considerably outperforms the classification approach for the audiovisual and visual features training, and is close to the performance of classification in the audio features training. Although the subjective experiment determined categorical quality values for the videos using the ACR scale, the MQS is a continuous value, and thus it is expected that a regression approach achieves better results than a classification approach. Of all the configurations tested, the best is the RV training, using the original feature matrix, without resampling and using the full set of audiovisual features. We also notice from the box plots that this configuration has the lowest variation, showing that it is also the most consistent. This configuration obtained a RMSE value of 1.0249, with PCC of 0.32 and SCC of 0.3132. From previous observations, this is expected, since regression and video prediction outperform classification and sample prediction, respectively, and using the full set of audiovisual features gives more information for the model to learn.

Table 5.2: RMSE, PCC, and SCC results from the multiple training setup experiment in Experiment 3 of the UnB Audiovisual Database.

Feature type	**Setup**	**Sampling**	**RMSE**	**PCC**	**SCC**
AV	CV	Original	1.1297	0.2188	0.202
		Downsampling	1.0646	0.0671	0.0722
	CS	Original	1.7932	0.1472	0.1529
		Downsampling	1.6875	0.2001	0.218
	RV	**Original**	1.0249	**0.32**	**0.3132**
		Downsampling	1.0891	0.1793	0.1598
	RS	Original	1.0459	0.2208	0.2389
		Downsampling	**1.0031**	0.1495	0.1029
Audio	**CV**	**Original**	1.1779	**0.2745**	**0.2717**
		Downsampling	1.1231	0.1426	0.1405
	CS	Original	1.834	0.1859	0.089
		Downsampling	1.7182	0.1903	0.144
	RV	Original	1.0365	0.2616	0.1985
		Downsampling	**1.0181**	0.252	0.2219
	RS	Original	1.0763	0.2222	0.0138
		Downsampling	1.0371	0.207	0.1838
Video	CV	Original	1.1342	0.1582	0.1136
		Downsampling	1.1699	0.1372	0.1155
	CS	Original	1.7331	0.0678	0.04
		Downsampling	1.7844	0.1639	0.0988
	RV	**Original**	**1.0416**	**0.2574**	**0.2731**
		Downsampling	1.0687	0.2185	0.2258
	RS	Original	1.05	0.1235	0.1244
		Downsampling	1.0686	0.1681	0.1748

5.3 Comparison with other quality metrics

Table 5.3 shows the comparison of the results of the best Transformer model in this work with multiple NR and FR metrics, reported by Martinez [1]. Figure 5.4 shows the box plot of the PCC and SCC results across the 10 folds of these results. FR metrics are video adaptations of SSIM [32] and PSNR. The NR metrics are VIIDEO [40] and video adaptations of DIIVINE [39], BIQI [87], NIQE [88] and BRISQUE [35]. A last set of audio and visual combination models is used: Linear, given by equation (2.1), Minkowski and Power, both proposed by Becerra *et al.* [49]. They use as input one video metric and one audio metric, DIIVINE and P.563 [89], respectively. In addition, we experimented with the XGBoost model [90], a classical and simple ML model composed of an ensemble of decision trees models, for comparison with a simpler model.

The expected results were for the Transformer model to outperform or at least get similar results to other metrics in the literature. However, we can observe from the results that both

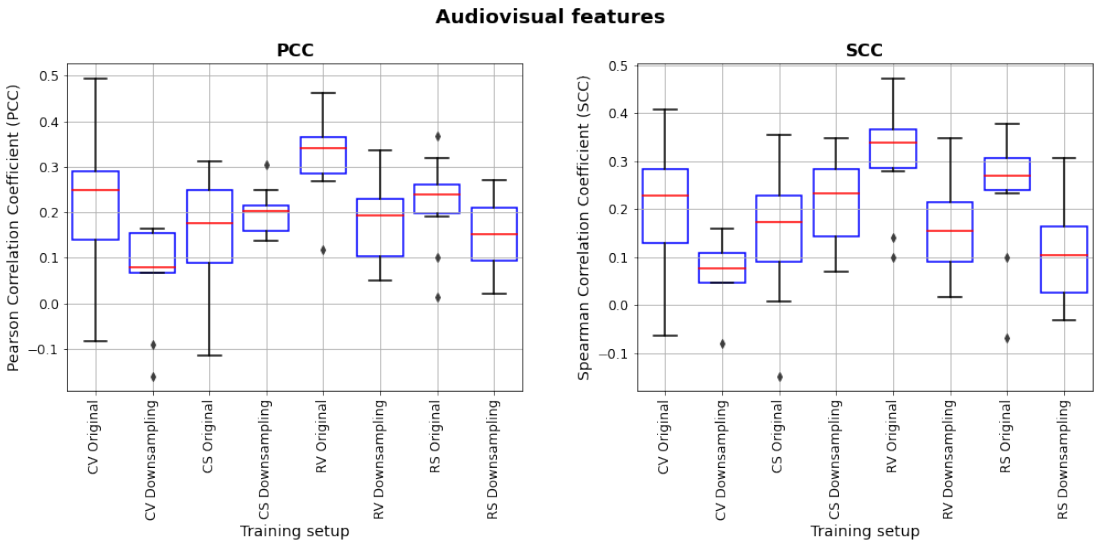


Figure 5.1: Box plot of the Pearson (left) and Spearman (right) Correlation Coefficients (PCC and SCC) results, across the 10 folds, from experiments with the different training setups using audiovisual features.

PCC and SCC values are consistent and very low, being outperformed by most compared metrics. This shows that the Transformer model was not able to generate a representation of the input data that was capable of mapping it to the target quality value. A few points can explain this behavior. The first possible explanation is that the sequences are strongly autocorrelated and redundant. Because the samples are too similar to each other, the Transformer's capacity of learning relationships between samples was harmed.

Furthermore, since we used tabular data, the complexity of the input data is low compared to the complexity of the model parameters. Transformers were originally created to solve problems using textual data, with enormous complexities originating from the structures of the language. When applied to an oversimplified set of features with high redundancy, the model presented both high variance and bias, at the same time underfitting by not learning patterns on the training data, but overfitting in the sense of being unable to generalize the learned parameters to unseen data. We can confirm this by observing the results obtained from the XGBoost model. Despite its considerably lower complexity, the model performs well on tabular data, achieving similar results to the deep learning models experimented and considerably outperforming the Transformer model.

In the work by Martinez [1], nevertheless, the autoencoder model being also a complex algorithm, the great results obtained are due to the anti-overfitting nature of the autoencoder model. When the data go through the bottleneck layers, the compact representation compensates for the autocorrelation and redundancy in the video features, allowing it to learn general patterns rather than individual sample information. The Transformer goes in the opposite direction by calculating the attention of each sample to generate its encoded representation, rather than generalizing. In addition, the Transformer model is an autoregressive model. This behavior is derived from the self-attention modules, which use the information from previous and future time steps in order to calculate the weights applied to each time step. However, the task of assigning a quality score to

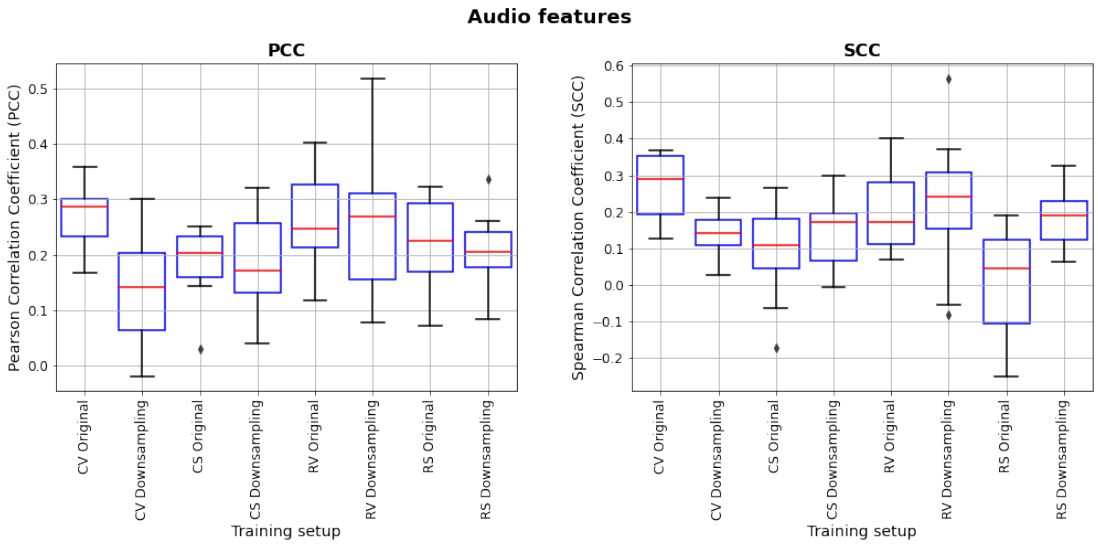


Figure 5.2: Box plot of the Pearson (left) and Spearman (right) Correlation Coefficients (PCC and SCC) results, across the 10 folds, from experiments with the different training setups using only audio features.

a video sequence is not autoregressive, since the video samples are independent, i.e. not affected directly by the previous and future samples.

Finally, one downfall of the Transformer architecture is its need for a large amount of data in order to get good results. For most video quality problems, the amount of data available for training models is limited. For that reason, most researches involving Transformers applied to videos use pre-trained weights, trained on large image databases and adapted for video. Since there are no other works using Transformers for audiovisual quality prediction, the model had to be trained from scratch, and given that the database used consists of only 800 videos, the model did not have enough data to be trained appropriately.

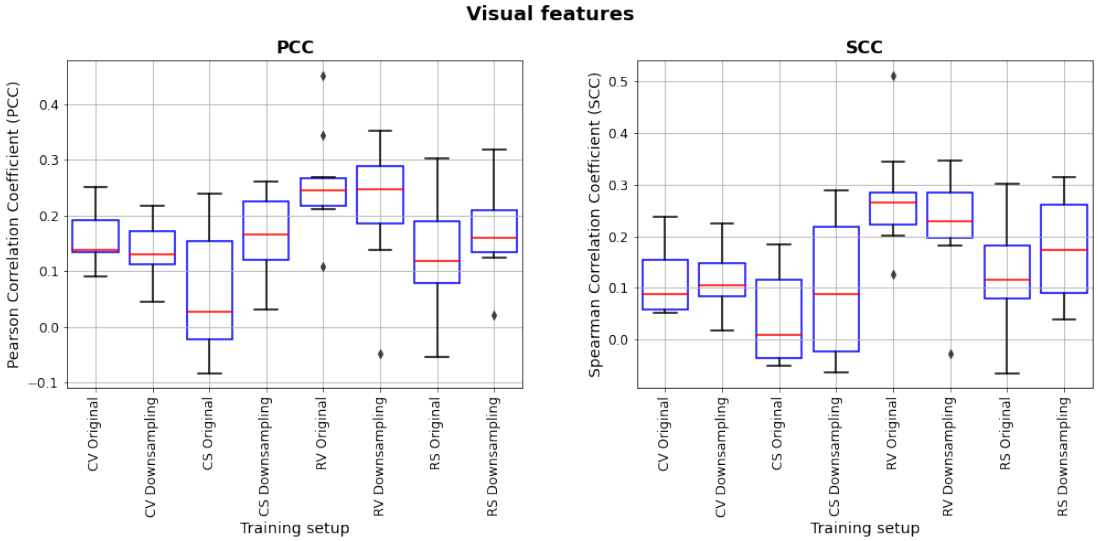


Figure 5.3: Box plot of the Pearson (left) and Spearman (right) Correlation Coefficients (PCC and SCC) results, across the 10 folds, from experiments with the different training setups using only visual features.

Table 5.3: RMSE, PCC, and SCC results from the experiments on Experiment 3 from the UnB Audiovisual Database. The XGBoost and Transformer metrics were obtained in this work, and the others were reported by Helard [1].

Type	Modality	Metric	PCC	SCC	RMSE
Full-Reference	Video	PSNR	0.7694	0.7368	18.0728
	Video	SSIM	0.3620	0.3579	2.4579
No-Reference	Video	DIIVINE	-0.8344	-0.7519	2.6939
	Video	VIIDEO	-0.8496	-0.7834	2.4449
	Video	BIQI	-0.8310	-0.8799	33.8917
	Video	NIQE	-0.8394	-0.7195	2.7119
	Video	BRISQUE	-0.8395	-0.7728	43.5049
	Audiovisual	Linear	0.4431	0.3368	10.7430
	Audiovisual	Minkowski	0.3422	0.3143	2.1973
	Audiovisual	Power	-0.6616	-0.6075	24.0462
	Audiovisual	NAViDAd	0.8819	0.8904	0.5850
	Audiovisual	XGBoost	0.7334	0.7351	0.7088
	Audiovisual	**Transformer**	**0.32**	**0.3132**	**1.0249**

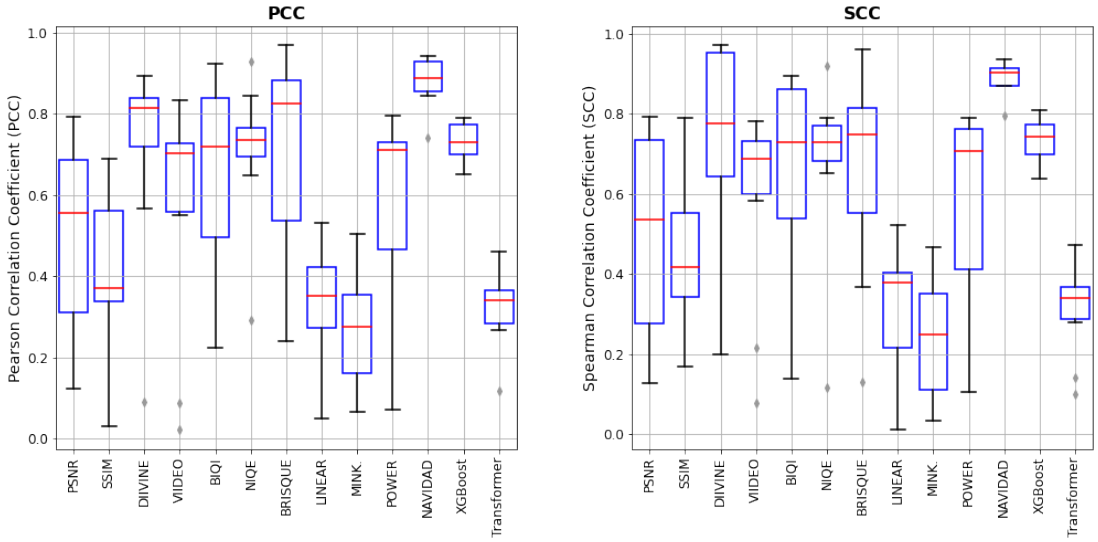


Figure 5.4: Box plot of the Pearson (left) and Spearman (right) Correlation Coefficients (PCC and SCC) results, across the 10 folds, from experiments with the Transformer best configuration and the other quality metrics. Source: Adapted from the original image in [1]

Chapter 6

Conclusion and future work

The proposal of this work was to use the Transformer architecture to train a model capable of objectively quantifying the quality of a video content. To our knowledge, no work before has proposed using Transformers for predicting AVQ using audio and visual features concurrently.

The data used for training was obtained from a public database, called UnB-Audiovisual Database [30] consisting of 40 videos containing 16 combinations of distortions and 4 anchors each, adding up to 800 video sequences. This is a small amount of data to train a Transformer model. However, since no pre-trained models or weights could be used, given the uniqueness of this approach, we had to train the model from scratch with the little data available.

This work experimented with a total of 24 different training configurations for each set of hyperparameters tested. Those configurations are a combination of 3 different feature selection techniques, 4 different model architecture configurations and 2 different input data treatment methods. From all the configurations, it was found that training the model as a regression problem, using a continuous target matrix and MSE loss, predicting the quality of the entire video rather than of each sample, and using the whole set of audiovisual features is the best approach, giving a RMSE value of 1.0249, with PCC of 0.32 and SCC of 0.3132.

Comparing these results with the results reported using other FR and NR metrics present in the literature, we observed that the Transformer architecture performs poorly in this task. This poor performance is due to various factors, including the little amount of data available for training, combined with the absence of usable pre-trained models or weights. Other important factor is the high autocorrelation and redundancy in the input features, since they are calculated for each sample of the video, which are very similar to its neighbors. It can also be explained by the low complexity of the input data, resulting in a model as complex as a Transformer to overfit the training data. Another reason is the autoregressive nature of the Transformer model, ensuing that non-autoregressive tasks have a lesser performance, which is the case for this task.

This work shows that, even though the Transformer architecture is widely used for many image and video related tasks, and is state of the art for many applications, its usage is still limited. When dealing with tabular data, as is the case of extracting features from video samples, using basic ML algorithms may solve the problem faster, easier and in some cases with better results.

However, in spite of the poor performance of the model, from the training configurations explored we observed that using the full set of audiovisual features is better than using only audio or only visual features separately. This result backs up the premise that audio and visual signals shape the QoE concurrently and should be a more explored area of research.

Future works could involve using a different feature extraction technique. Instead of extracting features sample by sample, perhaps using a method that gives a set of features that represent the entire video is a better approach, in order to avoid high autocorrelation and redundancy in the data. Other possibility is removing the feature extraction procedure entirely, and in order to keep the audiovisual characteristic of the model, combine the audio and visual signals, using this raw signal combination as input to a Transformer model. By doing this, pre-trained audio and visual models could combined and used in order to generate better results.

REFERENCES

[1] MARTINEZ, H. A. B. A three layer system for audio-visual quality assessment. 2019.

[2] WANG, Z.; BOVIK, A. C. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, IEEE, v. 26, n. 1, p. 98–117, 2009.

[3] MARTINEZ, H.; FARIAS, M. C.; HINES, A. Navidad: A no-reference audio-visual quality metric based on a deep autoencoder. *European Signal Processing Conference*, v. 2019-September, 2019.

[4] ANALYSTPREP. *Supervised Machine Learning, Unsupervised Machine Learning, and Deep Learning*. `https://analystprep.com/study-notes/wp-content/uploads/2021/03/Img_12.jpg`. Accessed: 09-09-2022.

[5] GUDIVADA, V.; APON, A.; DING, J. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, v. 10, p. 1–20, 07 2017.

[6] LEARNING, A. M. Developer guide. *Amazon Web Services*, 2018.

[7] W3SCHOOLS. *Neural Networks (NN)*. `https://www.w3schools.com/ai/ai_neural_networks.asp`. Accessed: 09-09-2022.

[8] COMMONS, W. *ReLU and GELU*. `https://upload.wikimedia.org/wikipedia/commons/thumb/4/42/ReLU_and_GELU.svg/1232px-ReLU_and_GELU.svg.png`. Accessed: 09-09-2022.

[9] SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

[10] READER, T. C. *Building a Convolutional Neural Network*. `https://www.theclickreader.com/building-a-convolutional-neural-network/`. Accessed: 09-09-2022.

[11] computersciencewiki.org. *MaxpoolSample2.png*. `https://computersciencewiki.org/images/8/8a/MaxpoolSample2.png`. Accessed: 09-09-2022.

[12] AMIDI, A.; AMIDI, S. *Vip cheatsheet: Recurrent neural networks*. 2018.

[13] LOYE, G. *Attention Mechanism*. Https://blog.floydhub.com/attention-mechanism/. Accessed: 09-09-2022.

[14] BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[15] VASWANI, A. et al. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.

[16] SANDVINE. *The Global Internet Phenomena Report October 2018*. `https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf`. Accessed: 09-09-2022.

[17] KORHONEN, J. Audiovisual quality assessment in communications applications: current status, trends and challenges. In: IEEE. *2010 International Symposium on Intelligent Signal Processing and Communication Systems*. [S.l.], 2010. p. 1–4.

[18] WANG, Z.; BOVIK, A. C.; EVAN, B. L. Blind measurement of blocking artifacts in images. In: IEEE. *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*. [S.l.], 2000. v. 3, p. 981–984.

[19] FARIAS, M. C.; MITRA, S. K. No-reference video quality metric based on artifact measurements. In: IEEE. *IEEE International Conference on Image Processing 2005*. [S.l.], 2005. v. 3, p. III–141.

[20] CAVIEDES, J. E.; OBERTI, F. No-reference quality metric for degraded and enhanced video. In: SPIE. *Visual Communications and Image Processing 2003*. [S.l.], 2003. v. 5150, p. 621–632.

[21] BOSSE, S. et al. Deep neural networks for no-reference and full-reference image quality assessment. *IEEE Transactions on image processing*, IEEE, v. 27, n. 1, p. 206–219, 2017.

[22] KANG, L. et al. Convolutional neural networks for no-reference image quality assessment. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 1733–1740.

[23] SHOEYBI, M. et al. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[24] DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[25] YU, Y. et al. Batman: Bilateral attention transformer in motion-appearance neighboring space for video object segmentation. *arXiv preprint arXiv:2208.01159*, 2022.

[26] AKHTAR, Z.; FALK, T. H. Audio-visual multimedia quality assessment: A comprehensive survey. *IEEE access*, IEEE, v. 5, p. 21090–21117, 2017.

[27] RECOMMENDATION, I. Vocabulary for performance and quality of service. *International Telecommunications Union—Radiocommunication (ITU-T), RITP: Geneva, Switzerland*, 2006.

[28] PINSON, M.; SULLIVAN, M.; CATELLIER, A. A new method for immersive audiovisual subjective testing. In: *Proceedings of the 8th International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM)*. [S.l.: s.n.], 2014.

[29] MIN, X. et al. Study of Subjective and Objective Quality Assessment of Audio-Visual Signals. *IEEE Transactions on Image Processing*, v. 29, n. c, p. 6054–6068, 2020. ISSN 19410042.

[30] MARTINEZ, H. B.; FARIAS, M. C. Full-reference audio-visual video quality metric. *Journal of Electronic Imaging*, SPIE, v. 23, n. 6, p. 061108, 2014.

[31] BAMPIS, C. G. et al. Towards perceptually optimized end-to-end adaptive video streaming. *arXiv preprint arXiv:1808.03898*, 2018.

[32] WANG, Z.; LU, L.; BOVIK, A. C. Video quality assessment based on structural distortion measurement. *Signal processing: Image communication*, Elsevier, v. 19, n. 2, p. 121–132, 2004.

[33] WANG, Z.; SIMONCELLI, E. P.; BOVIK, A. C. Multiscale structural similarity for image quality assessment. In: IEEE. *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003.* [S.l.], 2003. v. 2, p. 1398–1402.

[34] ZHANG, L. et al. Fsim: A feature similarity index for image quality assessment. *IEEE transactions on Image Processing*, IEEE, v. 20, n. 8, p. 2378–2386, 2011.

[35] MITTAL, A.; MOORTHY, A. K.; BOVIK, A. C. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, IEEE, v. 21, n. 12, p. 4695–4708, 2012.

[36] WANG, Z.; BOVIK, A. C. Reduced-and no-reference image quality assessment. *IEEE Signal Processing Magazine*, IEEE, v. 28, n. 6, p. 29–40, 2011.

[37] SHEIKH, H. R.; BOVIK, A. C.; CORMACK, L. No-reference quality assessment using natural scene statistics: Jpeg2000. *IEEE Transactions on image processing*, IEEE, v. 14, n. 11, p. 1918–1927, 2005.

[38] MOORTHY, A. K.; BOVIK, A. C. Blind image quality assessment: From natural scene statistics to perceptual quality. *IEEE transactions on Image Processing*, IEEE, v. 20, n. 12, p. 3350–3364, 2011.

[39] ZHANG, Y. et al. C-diivine: No-reference image quality assessment based on local magnitude and phase statistics of natural scenes. *Signal processing: image communication*, Elsevier, v. 29, n. 7, p. 725–747, 2014.

[40] MITTAL, A.; SAAD, M. A.; BOVIK, A. C. A completely blind video integrity oracle. *IEEE Transactions on Image Processing*, IEEE, v. 25, n. 1, p. 289–300, 2015.

[41] VARGA, D.; SZIRÁNYI, T. No-reference video quality assessment via pretrained cnn and lstm networks. *Signal, Image and Video Processing*, Springer, v. 13, n. 8, p. 1569–1576, 2019.

[42] HINES, A. et al. Visqol: an objective speech quality model. *EURASIP Journal on Audio, Speech, and Music Processing*, SpringerOpen, v. 2015, n. 1, p. 1–18, 2015.

[43] TAAL, C. H. et al. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In: IEEE. *2010 IEEE international conference on acoustics, speech and signal processing*. [S.l.], 2010. p. 4214–4217.

[44] MITTAG, G. et al. Nisqa: A deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced datasets. *arXiv preprint arXiv:2104.09494*, 2021.

[45] REDDY, C. K.; GOPAL, V.; CUTLER, R. Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In: IEEE. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2021. p. 6493–6497.

[46] AVILA, A. R. et al. Non-intrusive speech quality assessment using neural networks. In: IEEE. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2019. p. 631–635.

[47] DONG, X.; WILLIAMSON, D. S. An attention enhanced multi-task model for objective speech assessment in real-world environments. In: IEEE. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2020. p. 911–915.

[48] SCHNEIDER, S. et al. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[49] MARTINEZ, H. B.; FARIAS, M. C. A no-reference audio-visual video quality metric. In: IEEE. *2014 22nd European Signal Processing Conference (EUSIPCO)*. [S.l.], 2014. p. 2125–2129.

[50] WINKLER, S.; FALLER, C. Perceived audiovisual quality of low-bitrate multimedia content. *IEEE transactions on multimedia*, IEEE, v. 8, n. 5, p. 973–980, 2006.

[51] ALPAYDIN, E. Introduction to machine learning (adaptive computation and machine learning series). *The MIT Press Cambridge*, 2004.

[52] SAMARASINGHE, S. *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. [S.l.]: Auerbach publications, 2016.

[53] HENDRYCKS, D.; GIMPEL, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[54] LIU, Y. et al. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[55] LEWIS, M. et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. [S.l.: s.n.], 2020. p. 7871–7880.

[56] TOUVRON, H. et al. Training data-efficient image transformers & distillation through attention. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2021. p. 10347–10357.

[57] SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, v. 6, n. 12, p. 310–316, 2017.

[58] KRUEGER, D. et al. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

[59] RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[60] KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[61] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016.

[62] BA, J. L.; KIROS, J. R.; HINTON, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[63] DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[64] YOU, J.; KORHONEN, J. Transformer for image quality assessment. In: IEEE. *2021 IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2021. p. 1389–1393.

[65] IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. *International conference on machine learning*. [S.l.], 2015. p. 448–456.

[66] GONZALEZ, R. C. *Digital image processing*. [S.l.]: Pearson education india, 2009.

[67] SCHUSTER, M.; PALIWAL, K. K. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, Ieee, v. 45, n. 11, p. 2673–2681, 1997.

[68] HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

[69] KIM, Y. et al. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

[70] CHO, K.; COURVILLE, A.; BENGIO, Y. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, IEEE, v. 17, n. 11, p. 1875–1886, 2015.

[71] LUONG, M.-T.; PHAM, H.; MANNING, C. D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[72] CARION, N. et al. End-to-end object detection with transformers. In: SPRINGER. *European conference on computer vision*. [S.l.], 2020. p. 213–229.

[73] PARMAR, N. et al. Image transformer. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2018. p. 4055–4064.

[74] YAN, H. et al. Contnet: Why not use convolution and transformer at the same time? *arXiv preprint arXiv:2104.13497*, 2021.

[75] ZHANG, Y. et al. Vidtr: Video transformer without convolutions. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 13577–13587.

[76] NEIMARK, D. et al. Video transformer network. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 3163–3172.

[77] GABEUR, V. et al. Multi-modal transformer for video retrieval. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2020. p. 214–229.

[78] GIRDHAR, R.; GRAUMAN, K. Anticipative video transformer. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 13505–13515.

[79] CHEON, M. et al. Perceptual image quality assessment with transformers. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 433–442.

[80] GOLESTANEH, S. A.; DADSETAN, S.; KITANI, K. M. No-reference image quality assessment via transformers, relative ranking, and self-consistency. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. [S.l.: s.n.], 2022. p. 1220–1230.

[81] YOU, J. Long short-term convolutional transformer for no-reference video quality assessment. In: *Proceedings of the 29th ACM International Conference on Multimedia*. [S.l.: s.n.], 2021. p. 2112–2120.

[82] MARTINEZ, H. B.; FARIAS, M. C. Analyzing the influence of cross-modal ip-based degradations on the perceived audio-visual quality. *Electronic Imaging*, Society for Imaging Science and Technology, v. 2019, n. 10, p. 324–1, 2019.

[83] OSTASZEWSKA, A.; KŁODA, R. Quantifying the amount of spatial and temporal information in video test sequences. In: *Recent Advances in Mechatronics*. [S.l.]: Springer, 2007. p. 11–15.

[84] GONG, Y.; CHUNG, Y.-A.; GLASS, J. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021.

[85] HINES, A. et al. Visqol: The virtual speech quality objective listener. In: VDE. *IWAENC 2012; International Workshop on Acoustic Signal Enhancement*. [S.l.], 2012. p. 1–4.

[86] BOX, G. E. et al. *Time series analysis: forecasting and control*. [S.l.]: John Wiley & Sons, 2015.

[87] MOORTHY, A.; BOVIK, A. A modular framework for constructing blind universal quality indices. *IEEE Signal Processing Letters*, IEEE, v. 17, p. 7, 2009.

[88] MITTAL, A.; SOUNDARARAJAN, R.; BOVIK, A. C. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, IEEE, v. 20, n. 3, p. 209–212, 2012.

[89] MALFAIT, L.; BERGER, J.; KASTNER, M. P. 563—the itu-t standard for single-ended speech quality assessment. *IEEE Transactions on Audio, Speech, and Language Processing*, IEEE, v. 14, n. 6, p. 1924–1934, 2006.

[90] CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 785–794.