Towards Real-Time Video Quality Assessment Onto Low-Latency Streaming

Jelle Manders
jelle.j.manders@gmail.com

July 9th, 2021, 32 pages

Academic supervisor:	Adam Belloum, a.s.z.belloum@uva.nl
Daily supervisor:	Jeroen Mol, j.mol@exmachina.nl
Host organisation/Research group:	Ex Machina Group / Livery Team https://www.liveryvideo.com/

UNIVERSITEIT VAN AMSTERDAM
FACULTEIT DER NATUURWETENSCHAPPEN, WISKUNDE EN INFORMATICA
MASTER SOFTWARE ENGINEERING
http://www.software-engineering-amsterdam.nl

Abstract

In previous work, Vega et Al[1] studied the effectiveness of various machine learning models to see how well they qualify video quality compared to the Video Quality Model (VQM). Their focus was to find if any No-Reference (NR) Video Quality Assessment (VQA) model could perform as well as a Full-Reference (FR) model such as VQM, and if so, how strongly it would correlate. They demonstrated that an LS-boosted Ensemble Regression Tree model was capable of reaching a Pearson Correlation Coefficient (PCC) of 0.91. The segment length used by Vega et Al was 10 seconds, for all of their video samples. In this paper, we show that VQA with shorter segments is also possible, reaching a correlation of over 90% with the Full-Reference (FR) model VMAF.

Contents

Chapter 1

Introduction

Video Quality Assessment (VQA) is the practice of assessing the quality of a video sample, most often in a streaming context. Original high-definition material is compressed and sent to a user over an internet connection. During this compression and consequent decoding process, some of the quality of the video is lost. A VQA algorithm then attempts to quantify how much quality-loss the compression has caused.

VQA has gained more attention over the past years. Older models such as the Video Quality Model (VQM)[2] are quickly followed by newer, commercial methods such as Video Multi-method Assessment Fusion (VMAF)[3]. These models are both Full-Reference (FR) algorithms, meaning they compare the original video against the compressed, sent, and decoded result. These models have reached very high accuracy to the Difference Mean Opinion Score (DMOS) [4] set by human subjects, and been used to determine live-streaming video quality[5]. However, VMAF is too computationally costly to be applied in real-time situations. Furthermore, in practical applications, the original and compressed video material is not always available. Where Netflix has the luxury of being a Video on Demand (VoD) service, allowing for pre-processing of the video material, live-streaming has gained an ever larger foothold in the industry.

There have been successful attempts to develop real-time VQA models, for example by Vega et Al[1]. Their solution makes use of a No-Reference (NR) model, which only analyses the decoded video through a selection of metrics. The incoming video is decoded, analysed in real-time, and a video quality estimation is computed. While this is a great step forward in VQA for live-streaming services, there is a niche lacking research.

Live-streams always have a certain screen-to-screen latency. In this paper, the latency shall be defined as the time it takes for an image to be recorded, encoded, sent, decoded, and displayed on the end-user's screen. In the live-streaming context, some applications have a focus on minimising this latency. In order to improve interactivity, a short latency can be a great difference. A latency of ten seconds can be detrimental to a live-streamed game show with crowd participation. To minimise the latency in their platforms, the Ex Machina Group (or more specifically, their Livery team) has developed a streaming service that reduces the latency to 2 seconds, and aims to reduce this even further.

When dealing with these kinds of latency's, the aforementioned solution provided by Vega et Al. is not sufficient. This is because their model uses 10-

second segments to make their quality assessment. In a Livery-hosted stream, the high interactivity does not allow for 10 seconds of quality analysis. The subject of this research is to determine whether it is viable to adapt the existing solution to make a comparably accurate quality assessment using shorter video segments as input.

To this end, the work by Vega et Al[1] is used as the mould in which I will work. The first step will be to reproduce their work, i.e. to develop a model that uses ten second video samples to determine their quality. From this point, I will shorten these video segments, and determine whether a model can still reach competitive accuracy using less data to go on.

In Chapter 2, I will give an overview of relevant previous work. In Chapter 3, I describe the aforementioned niche in more detail, and give some context to the objective. Research questions will also be presented in this chapter. The method by which I hope to accomplish my goals is presented in chapter 4, and their results are presented in chapter 5. The results and their implication are discussed in the Discussion, chapter 6. In chapter 7 I give some recommendations as to what I feel would be valuable avenues of further research. And finally in chapter 8, I acknowledge all those around me who helped make this thesis a reality.

Chapter 2

Previous Work

In this chapter, a selection of previous work in the field is presented. Each work has some overlay with this research, whether that be as a benchmark, a work to build upon, or a side study that provides some insight into the edge cases of Video Quality Assessment (VQA).

Where various articles use the Video Quality Model (VQM) as their benchmark, Netflix made waves in the industry with the introduction of their Full-Reference (FR) VQA model, named Video Multi-method Assessment Fusion (VMAF). Originally presented in their Techblog[3], Li et Al explain the position of Netflix in the industry. The nature of their Video on Demand (VoD) service requires a more precise model than is currently on the market. Where most models also analyse the material for frame freezing, Li et Al argue that because of the robust Transmission Control Protocol (TCP) these artefacts never present themselves, since packet loss is no longer a source of quality degradation.

The approach of Netflix is unique in that they generate their own training set by taking a selection of video material from their servers, and constructing a user-based scoring for each of them. The participants are asked to view both the original and the compressed versions of the material, and scoring each. Taking all of this data, Netflix constructed a Difference Mean Opinion Score (DMOS) scale to compare VMAF with other quality models. Specifically, PSNR SSIM FastSSIM and PSNR-HVS[6]. The various graphs show that these models do not correlate perfectly with their self-constructed DMOS.

Finally, VMAF itself is presented. It is described that it consists of three major components, namely the Visual Information Fidelity (VIF)[7], the Detail Loss Metric (DLM)[8], and a motion measure achieved by comparing pixel differences between frames. Combined using Support Vector Machine (SVM) regression, the model proved to be even more accurate than Video Quality Model with Variable Frame Delay (VQM-VFD), which is reportedly "state of the art in the field". VMAF was for this reason selected as the benchmark of my research.

Vega et Al[1] have preceded this research in the endeavour towards real-time VQA. Their article describes a proposed Machine-Learning (ML)-based algorithm that reaches over 90% correlation with VQM. They achieve this by analysing the Live Video Database (LiveVD) using a selection of No-Reference (NR) metrics, and using these as input for a variety of ML models, benchmarked by VQM.

The research describes the usefulness of LiveVD by showing the complimentary nature of the different set of metrics they selected. Some metrics perform very well under certain conditions, but fall off in others. This is covered by the other metrics, complementing each other in hypothesised complete coverage. They continue to describe the selection of different ML models, finally deciding that an LS-Boosted Ensemble Regression Tree (ERT-LSB) has the greatest correlation with their benchmark.

As mentioned in the introduction, this work by Vega is used as inspiration and a diving board for this thesis. The way in which Vega et Al differ from my research is twofold. First, their artefact generation is done through both compression and packet loss. By the logic of the last article and the method of most modern live-streams, I chose to limit myself to only compression-based artefacts. Second and more importantly, Vega et Al have a focus on RT applications, but not on low latency.

Another work by Vega et Al[9] proposes an unsupervised deep learning model, that unifies the added accuracy of reduced reference with the performance of no reference metrics. They achieve this by performing metric analysis on the server side, and sending a data vector to the client side where inexpensive NR metrics are applied to construct the same vector. The difference in quality is then calculated by an unsupervised model on the client side using those two data vectors.

The training data is provided by the LIMP Video Quality Database, made by Vega and Liotta themselves. The database contains a large set of packet-loss impaired videos. The client side model, a Restricted Boltzmann Machine (RBM), reaches a correlation between 78 and 91% to VQM. The differences with this research are slightly larger than the last article by Vega et Al. Not only are the compression and low-latency focus missing, the proposed model is highly optimised for adaptability and scalability. Neither of which is prioritised in the following sections.

Barman et Al[10] pose that the passive gaming stream industry has been largely ignored by the scientific community, and that consequently there is a need for real-time Quality of Experience (QoE) algorithms to ensure the industry's growth. To this end they propose two ML-based models that make use of the specificality of their niche. They trade generality for precision by focusing solely on passive gaming streams, capitalising on the fact that gaming streams typically only contain computer generated graphics, resulting in a type of material that could theoretically more easily be assessed by an algorithm.

They go on to discuss their choice of video source material, a self made database called the Kingston Gaming Video SET[11]. This database contains footage from twelve different games, compressed in a multitude[1] of ways to create 576 distorted videos. These videos are benchmarked by VMAF, and the models are Neural Network and Support Vector Regression based. Their correlation with VMAF is brought as high as 97%.

[1]Specifically, the techniques applied are rescaling to other resolutions, and compression by certain bitrates.

Xue et Al[12] specialise in another department. Like Barman they trade generality for accuracy, but instead of the type of video they specialise in a type of artefact. In their research, they present a method capable of detecting temporal artefacts, such as jerkiness or frame freezing. They do this by analysing the video for any freezes, and then applying a set of metrics to these freeze sections. These include the number of freezes in the video segment, the average duration of a freeze, the average distance between freezes, and so on. The list of features reaches a length of thirteen, and can theoretically be applied in a real-time no-reference contsxt.

While the proposed list of metrics is quite extensive, their number of testing material is quite limited. With the combined material of both the VGEQ and LiveVD, the total number of test videos becomes 52. They overcome this limitation by calculating the number of hidden nodes their neural network should have, and how many input parameters this network can handle. They limit themselves to six metrics, and decide upon which should be included by experimental analysis. With these parameters decided upon, the network reaches a correlation of 90% and 80% with DMOS on the train and test sets, respectively. With these numbers they outperform a full-reference metric, namely VQM-VFD.

Chapter 3

Low Latency Video Quality Assessment

Where Video on Demand (VoD) services have no real time constraints for Video Quality Assessment (VQA), the opposite is the case for live streaming platforms. A service such as Netflix knows in advance what their content will look like, and can take all the time they need to assess their image quality after compressing and decoding the original. In a live streaming setting however, this is not the case. The content is created every second, and this content is not predictable. If quality assessment is a requirement, it will have to be done in real-time. There is the possibility of recording the entire stream and assessing its overall quality after it has ended, but there exist many scenario's where real-time assessment holds a strong advantage.

An example might be a live streamed quiz, where the host asks the crowd the questions. If with every correct answer the host launches a firework effect over the stream to celebrate a correct answer, this might cause an issue for the encoder, resulting in a drop in quality. If the producer only notices this after the entire stream has already ended, the show has shown poor quality throughout the stream. If the issue had shown itself in real-time, the producer could have chosen not to use the particle effect again, or chosen a less impactful alternative.

Making sure the VQA is performed in real-time is a good start, but not a complete solution to this problem. All of the existing solutions described in section 2 use video segments of around ten seconds or longer, even those with real-time being part of their requirements. If the aforementioned particle effect lasts no more than a second, and the quality is assessed by a sliding window of ten seconds, then the average quality might reveal nothing more than a vague dip in a window much larger than the cause itself. To solve this issue, the sliding window has to be made more narrow.

A real-time VQA model that uses shorter than ten second segments to make its assessment does not yet exist. The aim of this research is to provide a proof of concept for the proposed technology. By taking an existing RT VQA algorithm and adapting it to also work for shorter video segments, it will be possible to assess whether or not the models lose accuracy when assessing the quality of their ever shorter video segments.

Framed as a research question, this endeavour can be summarised as:

RQ1: "What is the effect of shorter video samples as training data on the accuracy of existing VQA algorithms?"

The remainder of this section explores the different facets and challenges of this question, and how they will be handled.

3.1 Data Format

The objective of this research is to improve the state of the art of Video Quality Assessment (VQA). The assessment of Quality is often done by analysing each frame in a video, possibly their relation to each other and the pixels in following frames. Therefore, it makes sense to choose a video format that allows for this kind of analysis. Hence, the first Sub Research Question becomes:

SQ1: "What format should the test/train data be in?"

While an mp4 is more common and takes up less storage space, the chosen format here is the `.yuv` format. This is because of its availability, and its proven effectiveness in comparable studies[1]. Specifically, the LIVE Video Database[13] has been selected as the main source of test and train video material. The database offers a range of video samples of different material, for example a panning shot in a park or a standing shot of a busy street. The variance in video type allows for an effective training environment that allows the model to anticipate many kinds of video material.

While the LIVE Video Database is a logical choice in both a format and precedent[2] sense, it is less ideal in other aspects. For one, the videos are in a low resolution of 768 by 432 pixels. This may have been state of the art streaming material back when the database was published, but improvements in internet technology have made for a jarringly different landscape. For another, while the database provides a good spread of video material to properly prepare a model for anything that is recorded by a real-world camera, many streams nowadays consist largely of computer animated material. Whether it is a game show host who has an animated background pasted over their green screen, or the entire stream being a live cast of a game full of computer generated graphics. These kinds of video material are wholly absent from the LIVE Video Database's videos.

These issues are taken for granted in the interest of time, and because that despite them it is very likely the research will yield a conclusive result. By the same logic, the choice was made to only analyse the luma component of each frame, effectively rendering the frames to 2D arrays containing a greyscale image. This allows for faster analysis and more efficient storage, since every video is stripped of half of its data.

3.2 Artefact Generation

At this stage, a video library consisting of ten videos are gathered. These videos are in pristine quality, with no quality degrading artefacts present. These need to

[1]Vega et Al[1] achieved a Pearson Correlation of over 90% using `.yuv` videos.

[2]As will become clear shortly, the model chosen to test our research question on is based on the work by Vega et Al[1], who used this same database.

be generated, preferably in a fashion that is comparable to their natural origin. Remember here that the research focuses on live-streamed content, hence the question becomes, how is quality typically lost in live streaming environments?

Vega et Al[1] posed that there are two main sources for quality degradation: Data Compression, and Packet Loss. Consequently, the paper describes how the original videos are processed by a video encoder, and sent over an artificially lossy internet connection. The encoder was given a bitrate ranging from 64 to 5120 kbps, and the internet connection ranged from 0% to 10% packet loss.

While this is a good beginning, the second sub question arises:

SQ2: "How should the quality lessening artefacts be generated?"

For the purposes of this research, Vega's example is partially followed. The modern streaming service has to compress data with varying bitrates, so the quality loss of this dimension is worth generating. However, the modern TCP-protocol[14] and general improvements to the stability of internet connections make packet loss a negligible source[3] of artefacts. Hence, this research focuses its efforts solely on the artefacts generated by video compression. For the sake of consistency with and comparability to earlier research, using the same bitrates that were used by Vega et Al. Specifically, the bitrates 64, 640, 768, 1024, 2048, 3072, 4096, and 5120 kilobits per second.

3.3 Machine Learning Model Selection

After having gathered both reference material and their compressed equivalents, the next step is to choose a method to tell one from the other. Or more specifically, a model will be needed that is able to assess the quality of a video. In accordance with much of the previous research, this method will be a Machine-Learning (ML) model. While this does give us a certain frame to work with, a choice in model allows us to focus more on a specific type of training data. Hence, the next sub question becomes:

SQ3: "Which ML model(s) are most likely to yield useful results?"

Once again, Vega et Al provide part of the answer to this question. Their research into the applicability of ML models in real-time VQA comes with an assessment of the possible accuracy of a wide selection of models. Eventually, they settle on an LS-Boosted Ensemble Regression Tree (ERT-LSB) model[1]. This research attempts to recreate that model with as much accuracy as possible, within the available timeframe. Hence, the aim is to create an ERT-LSB using our own compressed video material.

Such a model could conceivably be trained using the raw video data, but the amount of samples required to properly train a model with an input vector of 768x432[1] elements (one for each pixel) far exceeds the availability of video material. For this reason, and for the sake of remaining close to the original research by Vega et Al[1], the video material is to be pre-processed and analysed before being fed to the model. This is done according to a selection of metrics.

[1]This is the data carried by just one frame, if an entire video were to be analysed by a model, this number should be multiplied by the amount of frames in the video. For an optimistic scenario, a one second video at 25 frames per second would carry $8.3 * 10^6$ data points as an input vector.

3.4 Choice of Metric and Video Analysis

Vega et Al[1] motivated their choice of metrics by doing a correlation analysis for each one they considered. Upon performing this analysis and plotting their correlation with the Video Quality Model (VQM), their benchmark, they found that the metrics they decided on complemented each other in terms of correlation. No one metric showed complete correlation over the range of compression bitrates, packet loss, and video category, but combined the metrics hold a broader range of predictability which a model could use to achieve higher overall correlation with the benchmark.

Of course, the approach of this research slightly differs from that of Vega et Al, so it stands to reason that the optimal selection of metrics might be slightly different. The next sub question then becomes:

> **SQ4:** "Which of the metrics proposed by Vega et Al are most applicable to a Low-Latency environment?"

The first step towards answering this question is to categorise the eight metrics that Vega originally used. These are Scene Complexity, Motion Intensity, Blockiness, Jerkiness, average Blur, Blur ratio, average Noise, and Noise ratio. These metrics are all No-Reference (NR) and pixel-based. In the following, they shall each be described shortly and it shall be argued why they are or have not been included in the model produced by this research. Their more formal definitions and implementations can be found in my public Git repository[15].

- **Scene Complexity and Motion Intensity** are computed by the method proposed by Hu and Wildfeuer[16], and are based on the compression process of a video using a certain bitrate. When encoding a `.yuv` video, the scene is encoded using I-frames and P/B-frames. The I-frames represent a full image, and each consequent P- or B-frame encodes the changes that bring the previous frame to the next. When trying to ascertain whether a scene is complex or has a lot of motion, we can look at the relative size of the I-frames and P/B-frames. If the compression process spends a relatively large part of its data on the construction of I-frames, this means each frame contains a lot of complexity that requires more data to encode, indicating a complex scene. Consequentially, a large expenditure of data on P/B-frames indicates that there is a large change of pixels between each frame, probably caused by relatively high motion intensity.

  The relativity is ascertained by dividing the amount of bits by a factor containing the quantization parameter. By the words of Hu and Wildfeuer: "the coded I-frame and P-frame bits are good indicators only if the impact of QP on them is removed."[16] For more details, please refer to the original article. The precise method of application of this, and all the following metrics, will be discussed in the next chapter.

- **Blockiness** arises when a video is compressed using an algorithm based on the Block Discrete Cosine Transform (BDCT)[17]. The compression method I used, JPEG H.264[18], is one of these algorithms. Using this method of compression has been shown to produce images that appear blocky, or having rough edges that are not in the original image. The lower the bitrate used during compression, the higher the likelihood of the compression algorithm cutting a corner and deciding to colour a block of pixels all one colour. Blockiness is detected by dividing the picture into blocks of 8x8 pixels, and analysing them for blockiness. The average blockiness over the 8x8 blocks is the blockiness of the frame, and the average per frame becomes the blockiness of the entire video. This approach is more specifically defined in the paper by Perra[19], which has been followed in the implementation for this research.

- **Jerkiness** can be described as the jittery motion that occurs when either an internet connection or the video encoder can't keep up with the data stream, and decides to skip a frame in order to catch up. This skipping of frames is a perceptible artefact, commonly referred to as a freeze-frame. The loss in quality caused by freeze-frames is called jerkiness.

  Vega[20] describe a method they used to compute the jerkiness of their videos, and used it as part of their model. However, their artefact generation is partially based on packet loss, which this research chose to omit (as described in the above). Hence, without packet loss and with the introduction of the TCP-prootocol[14], the loss of frames and thereby the freezing of frames is something that should not occur in the training set used by this research. The implementation of a jerkiness metric has consequently been omitted.

- **Blur mean and ratio** are two metrics used by Vega et Al[1], the definition of which originated in an article by Choi et Al[21]. Blur is perhaps the most well-known artefact, since everyone has encountered a blurry image. In video compression, blur is caused by compression using too low a bitrate for the encoder to properly store each pixel. Instead, the encoder decides to store the data of multiple pixels in one memory address, causing the pixels in question to become an averaged colour. This results in a blurry image, the effects of which are akin to the upscaling of an image.

  The definition provided by Choi et Al[21] and implemented by Vega et Al[1], provide a method to compare edge pixels in the image, and decide which of these edge pixels are blurry. this allows for the calculation of both a measure of blurriness, and a measure of how many of the potentially blurry edges are actually blurry, resulting in a blurriness ratio. For this research, an attempt was made to implement the same method. Sadly however, the implementation proved to be quite difficult to implement, and was eventually abandoned in favour of a simpler method. To substitute this metric, a method was selected from an article by Pertuz et Al[22]. This metric is the variance of the laplacian over each frame. This method allows us a strong substitute for the method used by Vega, but does have the drawback of not being able to supply the model with a blur ratio. Hence, the average blur has been implemented as one of the metrics, while the blur ratio has been omitted.

- **Noise mean and ratio** have the same origin as Blur mean and ratio, as they are originally proposed in the same article by Choi et Al[21]. The method proposed has a comparable approach to that of the blur method, and hence provides both a measure of how noisy an image is, and analyses the amount of possibly noisy edges divided by the amount of actually noisy edges, providing us with both a noise mean and ratio. While the method and implementation of the noise metric is highly comparable to the blur method, the noise gave substantially less issues during implementation. Hence, both Noise mean and Noise ratio have been included in the model of this research.

3.5 Labelling and Benchmarking

Now that a method has been defined by which the video material shall be assessed and vectors can be constructed, the question becomes how to label these vectors. As has been mentioned in the above, the benchmark that will be used for this research is the Video Multi-method Assessment Fusion (VMAF)[3] algorithm by Netflix. This algorithm is perfectly capable of assessing the quality of any video it is supplied with. However, it was originally designed to work with longer video segments, which means there is no guarantee that the analysis of a shorter video (with a duration of one second for example) provides an accurate assessment. Given this challenge, the following question requires an answer:

> **SQ4:** "How can the labelling be done such that the comparison to the benchmark is most scientifically valid?"

The challenge that needs to be overcome is the labelling of shorter videos. Simply passing them to VMAF and asking for an assessment is inherently risky. While the algorithm provides an assessment of both the video as a whole and each frame in the video, a case can be made that the assessment of shorter videos should provide no issue. However, the methods used to assess singular frames are inherently inferior to those assessing longer segments of video because they lack the ability to compare consecutive frames. This comparison is what provides the ability to analyse jerkiness, to give only one example.

Another option is available, and that is to make the assumption that the assessment of the whole is a strong representation for the parts. If this assumption is correct, the shorter segments can simply copy the labels of their complete counterparts. To support this assumption, an analysis has been made of the uniformity of the individual frame assessment. The rationale here is that if the assessment of single-frame quality is uniform across the entire length of the video, then the quality will not change much throughout its length, and therefore the assessment of the whole strongly represents the hypothetical assessment of the parts. The analysis of the uniformity that this method hinges on, will be discussed in the Method section.

Chapter 4

Method

In the previous chapter, the challenges of answering the research questions are discussed, and preliminary solutions have been put forward. In this chapter, I will present the actual work that went into facing the challenges, and how I came to decide which approach I would take to tackle each.

4.1 Data Used

The first step was to collect the video material that would be analysed. As discussed in the last chapter, the video database I decided to use was the Live Video Database (LiveVD)[13]. This database provides 10 categories of video, each with a reference video, and a set of pre-distorted files. The LiveVD is meant to be used to train a Video Quality Assessment (VQA) model, where the pre-distorted files serve as a good spread of artefacts that a model could learn from. For my research, I decided to forego the pre-distorted images, choosing instead to perform the artefact generation by compression myself. This ensures accurate reproduction of the results by Vega et Al[1].

The LiveVD is available to all researchers that get permission from the original authors of the article[13] that it was first presented in, as described in the same article. The article also includes the exact specifications of the video material in the database.

Category	**Abbr**	**Fps**	**Length (s)**
BlueSky	bs	25	8.68
Mobile Calendar	mc	50	10.00
Park Run	pr	50	10.00
Pedestrian Area	pa	25	10.00
Riverbed	rb	25	10.00
Rush Hour	rh	25	10.00
SunFlower	sf	25	10.00
Shields	sh	50	10.00
Station	st	25	10.00
Tractor	tr	25	10.00

Table 4.1: Live Video Database Specifications

The videos are separated into ten categories, each with their own abbreviation to make clear which is which. Most of the categories have a framerate of 25fps, some have an increased rate of 50fps. This can be seen in more detail in table 4.1. In this table, the length of each video is also included. The reason for the somewhat strange length of the Blue Sky video is unclear. For a more detailed description of the content of each video, please refer to the original LiveVD article[13].

After acquiring the raw material from the database, each video had to be compressed, and later in the process be segmented in various pieces. There are many tools to perform these kinds of task, but I chose to use FFmpeg[23]. Partly because of the many builtin functions that will provide me with all the necessary functionality, but also because of the large amount of experience and expertise of my host company. I used FFmpeg for two aspects of the video processing: To compress the videos using a certain bitrate, and to cut the videos into shorter segments.

The compression of a video can be done through a multitude of methods, but among the most commonly used is the MPEG/H.264[18] standard. It is also used by Vega et Al[1], meaning it is also a logical choice in the reproduction department. Using this standard, each reference video was converted using a certain bitrate to generate a full-length compressed version of each category video. The bitrates used were the same as in Vega's research, namely 64, 640, 768, 1024, 2048, 3072, 4096, and 5120 bits per second. With ten categories and eight compression bitrates, this leaves us with eighty compressed videos. These videos are however in the `.mp4` format, so before processing these would have to be converted back into `.yuv`. Both of these operations were performed using built-in FFmpeg functions.

After having generated the eighty full-length videos, the material was also split into segments of various lengths. The choice of how long to make these segments exactly, I based on the amount of frames in each video. Since the videos are (almost) all ten seconds long, the 25fps and 50fps material consists of 250 and 500 frames per video, respectively. When cutting these videos into segments, I would most ideally want to avoid any cutting that tampers with the quality. If a video is cut in between frames, the encoder will have to decide on an in-between frame to fill up space. These in-betweens are generated, and are therefore inherently of less quality than the pristine original. With this in mind, the most preservative method of cutting the material into segments is to cut in between frames. Adding the further requirement that each segment has to have the exact same length, the resulting segment lengths that the original will be cut into, are five, two, and one second. These will respectively result in videos consisting of 125, 50, and 25 frames, or double that if the original happens to be shot in 50fps.

FFmpeg also carries a built-in function to achieve this, the only special case is the video in the Blue Sky category. Since this video is 8.6 seconds long, it cannot be cut into 10 segments of 1 second each. Instead it has been cut into nine segments, the last of which only lasts 0.6 seconds. It is worth mentioning that the same approach was applied to the five and two second long segments, resulting in one segment of 3.6 seconds, and another of 0.6 seconds. Given that these are singular items in a list of 160, 400, and 792 items for the respectively five, two, and one second video array, I deemed this harmless and allowed the video into the training data to maintain the spread over the categories.

4.2 Labelling and Benchmarking

After having generated the compressed versions of the video, in both the full length and the segmented versions, the next step is to get our benchmark to label each. As discussed in the previous chapter, the benchmark I will be using is the Video Multi-method Assessment Fusion (VMAF)[3] algorithm by Netflix. VMAF was installed on an Ubuntu 18.04, the VMAF version is 2.1.1. For each compressed video, the compressed version was compared to the reference to reach a score. This number between zero and a hundred indicates how much of the quality is lost. These scores are used as the labels that the model will train with. As previously indicated, the labels generated by VMAF are only completely reliable on the full length videos. Any shorter than ten seconds is not what VMAF was built for. Hence, I had to implement some sort of method to extrapolate a label for the shorter segments from the label of the full-length videos.

4.2.1 Uniformity of VMAF Classification

The method I implemented to achieve this has already been described in the previous chapter. I simply assume the videos from the Live Video Database (LiveVD) are of the same quality in every segment. If this is the case, we can stick the label of the full-length version on each segment that splits off from it. To support this assumption, I look to one of VMAF's features. It does not only supply the user with an aggregate score for the entire video, it also generates an assessment of the quality per frame. This is not as strong an assessment, since the comparison of frames holds information towards the quality of a video, but it can be used as an indication of how uniform the assessment of the video is.

To get an idea of the uniformity of the assessments, I had Video Multi-method Assessment Fusion analyse each compressed video, and analysed the concurrent frame-by-frame analysis. The results can be seen in figure 4.1. Each plotted line represents the aggregate score given by VMAF for the video of that category, compressed by that bitrate. The categories used in the legends are abbreviations, their explanations can be found in table 4.1. Each point with an error bar represents a compressed video. The error bars represent the standard deviation of the frame-by-frame score.

Looking at this graph, one can see that in most cases the error bars are small enough that there is hardly any doubt that the quality of the video is consistent across frames. The only places where this isn't the case is in the lowest bitrates, and in the River Bed category. The higher variance in the lower bitrates is somewhat expected, video compression is randomly more or less efficient depending on the exact image. If the objects moving through an image happen to line up in such a way that they are easier or harder to be compressed, we perceive a momentary rise or fall in quality at the frame level. The higher the quality, the less likely it is that any such fluctuation fall out of the bounds of the higher bitrates capacity.

This also directly clarifies the outlier in the subplots, namely the River Bed deviations. This video pictures a shallow streaming river, which is clearly harder to compress even with higher bitrates. In the lower regions, the same high standard deviations are found. This is a threat to the validity of assuming the VMAF score of the whole is representative of the shorter segments' scores.

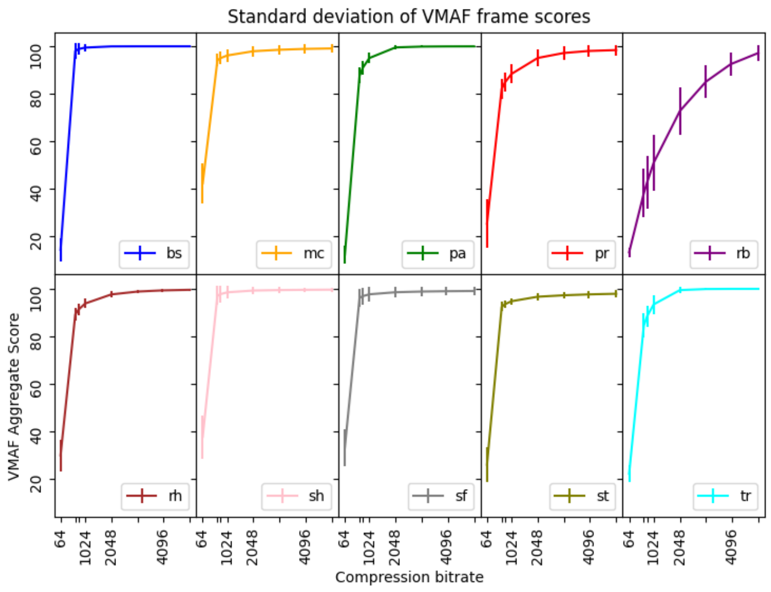

Figure 4.1: Standard deviations of VMAF frame scores, plotted over the aggregate assessment of that category, per bitrate.

However, the fact that they are in the lower quality region somewhat makes up for it. If our model mislabels a video as having a score of 30, but should have given a score of 40, this reflects poorly on our correlation. In practical purposes however, the difference between 100 and 90 is much more important than the difference between 40 and 20. A video with bad quality or worse quality is still bad quality. No problems arise so long as we don't mislabel something with flawed quality as having perfect quality.

4.3 Metrics

After having acquired the different videos, their compressed versions, and a label for each of these, the next step is to analyse this material. The metrics and their motivation have already been discussed at length in the previous chapter. As mentioned in that section, the metrics were largely based on the choices made by Vega et Al in their research[1]. The implementation of these metrics was done completely by hand.

The full code can be found on this research its Github[15]. There one can go over the implementation line by line if required. Note here that the focus of the implementation was one of clarity over performance. I chose to keep the implementation as simple as possible to prevent any mistakes. The implementation is not meant to be fast. The metrics offer the possibility of much parallel programming, since most of the metrics analyse the file frame by frame. A fast solution would make use of this fact by having only one file reader, and offering each frame to a set of analytic engines. I chose another method, one in which I provide each metric with its own file reader, in an attempt to keep the implementation as contained as possible.

This approach does come with a downside. The implementations have barely been optimised. The combination of nested for-loops, heavily duplicated code, and lack of parallel programming make for a very slow analysis process. This issue was circumvented by having the metric analysis functions run on a remote server with high amounts of computation power.

4.4 Model Setup and Calibration

In their research, Vega et Al analysed a range of Machine-Learning (ML) models to determine which is best suited for the task of Real-Time (RT) VQA. Of the nine models that were experimented with, one was deemed the most accurate: an LS-Boosted Ensemble Regression Tree (ERT-LSB). Since this research aims for reproducibility, the original aim was to implement this same model.

However, the model proved somewhat elusive in its implementation, and in order to not lose too much time I chose to implement a more simple substitute. From the sklearn[24] modules, I implemented the RandomForestRegressor class. This model is as close to the ERT-LSB as I could find in reasonable time, the only aspect that is missing is the LS-boost. Both models function on the basis that a large set[1] of models is constructed, each of which are then analysed and combined in such a way that the combined model's accuracy is optimised. This process is inherently well-suited to biased datasets[25] and not prone to overfitting. Because the data gathered by VMAF on the video dataset is very skewed, this model is perfect for my purposes. It is however not the model I set out to implement. While I did consider to invest some extra time into the development and implementation of an LS-Boosted Ensemble Regression Tree, The first results of the Random Forest Regressor will show that there is really no need, especially considering the time restraints. More on this will follow in the results and discussion chapters.

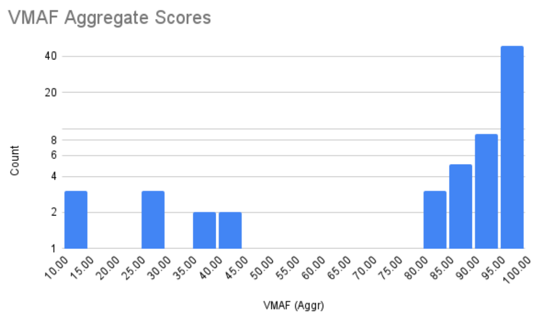

Figure 4.2: VMAF Scores Histogram over the 10s video dataset

The analysis of each compressed video showed that most of the videos have fairly good quality. Even at a glance, it is clear to see that many of the videos received a VMAF score between 95 and a 100. The exact histogram can be

[1]500 in this case

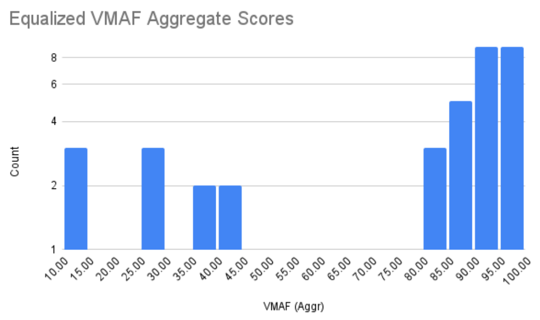

Figure 4.3: VMAF Scores Histogram over the equalised 10s video dataset, note that compared to figure 4.2, 40 samples in the 95-100 range have been removed.

seen in figure 4.2. Exactly 49 out of the 80 videos received a score in the top 5th percentile. The first set of tests was done with this dataset, just to see what would happen and whether or not the results would be as skewed as the dataset. The results of this experiment are discussed in the next chapter. The exact configuration of the model and its parameters can be found in the project's public repository[15].

While the choice of model is an adequate answer to the issue of skewed data, I still wanted to see if the model could improve by doing something about the issue directly. I did so by developing a second data gathering function that filters 50% of the top percentile data. In doing so, it balances the data out more, which could lead to a decrease in overfitting. The selection of which datapoints are filtered before training is randomised completely, but I did plot out a histogram for the VMAF scores after filtering. This can be seen in figure 4.3.

Note that the histogram in both figure 4.2 and 4.3 are concerning the ten second database. Hence, 50% of the data comes down to 40 datapoints being stripped from the list. In the larger datasets, this same 50% is scaled up to reflect the higher number of data points.

With all this set up, I ended up performing eight experiments. I trained and evaluated four Random Forest Regressors for the original skewed dataset, one model for each segment length[1]. I calculated their Pearson Correlation Coefficient (PCC) with the original VMAF labels, and plotted each predicted value against its original label. These are the first four experiments, the second four are identical, but use the randomly equalised dataset. In all, this results in two images each containing four graphs, which will be presented in the Results chapter. The precise implementation of the model and the different tests can be found on the project's github[15].

[1]These lengths are 10, 5, 2, and 1 seconds

Chapter 5

Results

In this chapter, I will present the results of the experiments described in the method.

5.1 Sklearn RandomForestRegressor

This section will present the predictions made by the Random Forest Regressor models, trained on the labels gathered by VMAF and the metric data calculated by myself. First I will present the results of using the complete dataset, followed by the equalised[1] dataset.

5.1.1 Complete Dataset

As discussed in the method chapter, I conducted four experiments with the complete dataset. One for each video segment length, namely 10, 5, 2, and 1 second videos. I trained a model on each set of labels and metric data, performed a tenfold cross validation, and asked the model for predictions of the data after training. These predictions have been plotted into a scatterplot, and can be seen in figure 5.1. Each graph in the figure denotes the video length in the top-left corner, and displays the Pearson Correlation Coefficient (PCC) in the top-right.

In the bottom-right graph we see the results of the ten second segment predictions. It is quite clear that the bulk of the data has a high label, with a large cluster of points in the [95−100] region. Also notable is the accurate cluster in the lower and higher regions, but there are some points that are misclassified as either having too low or too high a score. These outliers are most likely to blame on the small size of the dataset, especially since these outliers become rarer and less pronounced in the shorter segment graphs, which are substantially larger. The larger datasets allow for the model to train itself more completely, becoming ever more likely to catch the outliers and improve its accuracy.

While the correlation is strong and the linear relationship is clear, the remark must be made that the model can score an easy 90% accuracy by blindly guessing at a score between 95 and a 100, no matter the metrics. This can be observed by seeing the large cluster in the top-right corner of each graph in the figure. This

[1]as discussed in the method chapter

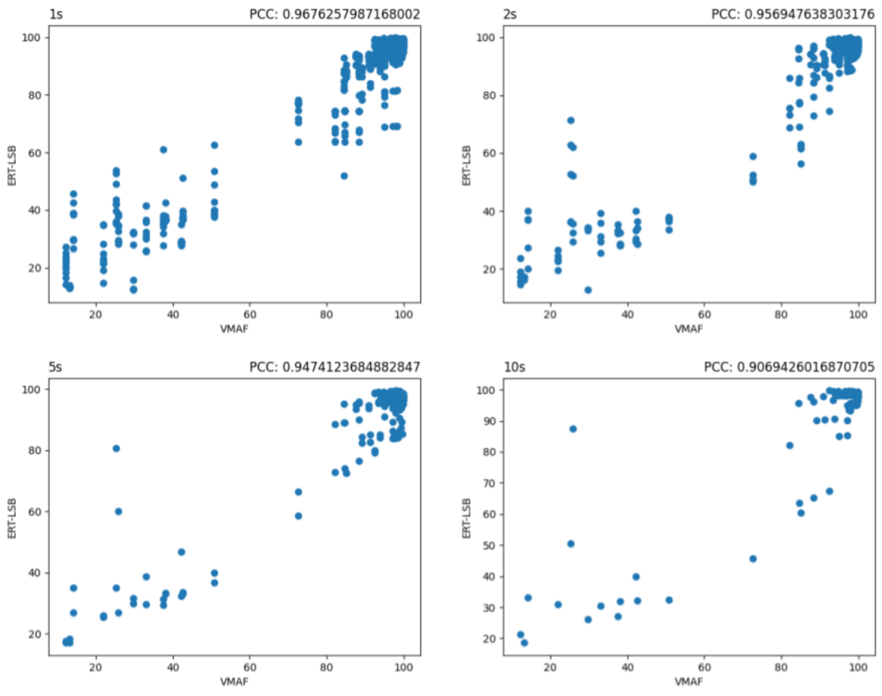

Figure 5.1: Random Forest Regression graphs, using the full dataset

is no surprise, the effect was anticipated and charted in figure 4.2. As discussed in the method chapter, I have tried to do something about this by culling some of the datapoints and conducting another experiment with this 'equalised' data.

5.1.2 Equalised Dataset

For the equalised dataset, the approach is almost exactly identical. The only difference is that the dataset is equalised before being fed to the models in question. This is done by calculating the length of the dataset, finding how many need to be filtered out, randomly taking that many from the labels between 95 and 100, leaving us with a more balanced dataset.

The results can be seen in figure 5.2, the layout is exactly the same as in figure 5.1. Especially in the ten second segment graph, the effects of the data filtering can clearly be seen. The cluster in the top-right is less dense, and some patterns can be recognised.

The outliers are still present, apparently the filter has not done much about the elusive nature of these few points. As we decrease the segment length however, we see that the data filter has helped create a more linear relation, especially in the top-right corners. Where in figure 5.1 the top-right corners of the graph have cornered clusters, the dots in 5.2 are more angled and in line with the expected correlations. This makes the claim that the high PCC values are due to a precise model more substantial. Before the test it was very possible that the correlation got an unfair boost because of the high-density cluster in the 95 to 100 range.

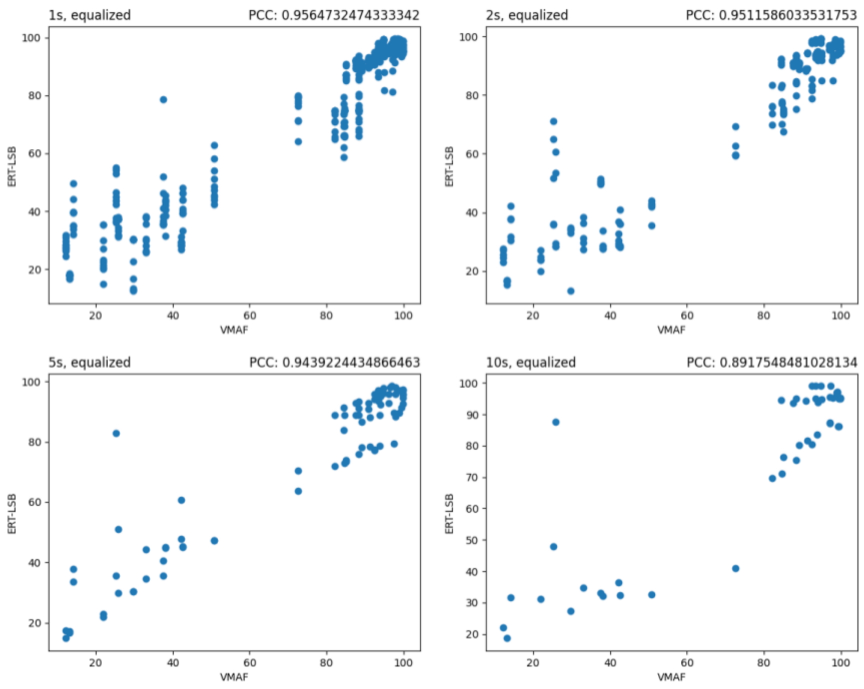

Figure 5.2: Random Forest Regression graphs, using the equalised dataset.

5.2 Correlation over Segment Length

To properly compare the effects of video segment length over correlation with the benchmark, the Pearson Correlations from figures 4.2 and 4.3 have been plotted in figure 5.3. While the lines in between the dots really don't represent anything, they do illustrate that the correlation only decreases when the metrics are calculated using more data. I would also like to point out that the equalised data has a consistently lower correlation with the benchmark. The explanation for this will be discussed in the following chapters.

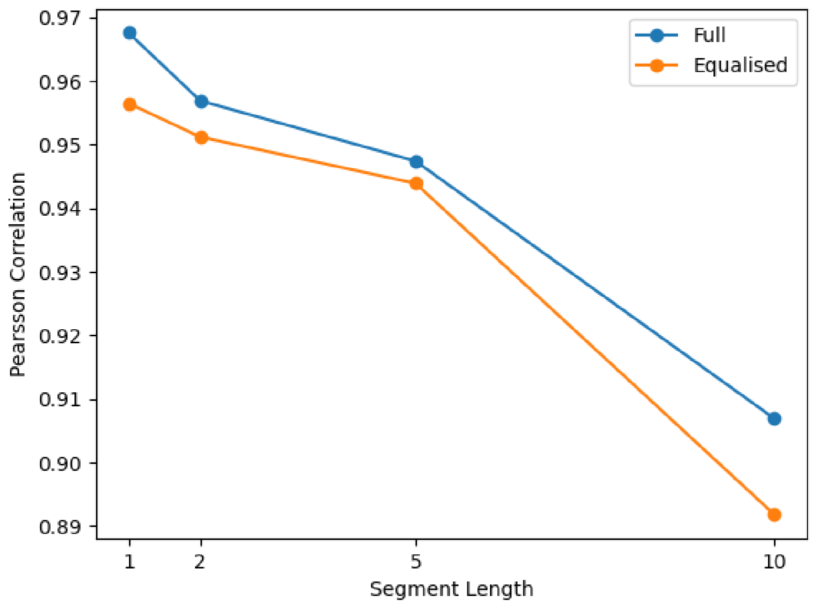

Figure 5.3: Correlation with the VMAF benchmark over video segment length used to calculate the metric data

Chapter 6

Discussion

The purpose of this chapter is threefold. First, I will evaluate the results in the last chapter, and discuss whether or not they provide an adequate answer to the research questions posed at the beginning of this thesis. After this the validity of the research is discussed by exploring what threats may still be present. Finally, I will discuss any lingering plans or issues I had planned to explore and describe, but did not have the time to bring to fruition.

6.1 Conclusions Drawn

The original goal of this paper was to answer the research question posed in chapter 3: "What is the effect of shorter video samples as training data on the accuracy of existing VQA algorithms?" From the results presented in the last chapter, I pose that the answer can be summarised as 'hardly any'. While there is a clear downward trend visible in the correlation over segment length in figure 5.3, this is most likely caused by the size of the dataset used to derive these correlations. We see that if we filter some of the datapoints from the set, a slight decrease is also visible. This clearly indicates that the model suffers from being denied its data.

Another scenario that would provide the same outcome is if the model does suffer from the shortening of the segment length, but this effect is more than compensated for by the increase in data to train from. I deem this possibility unlikely, since that would require the predictability of the metric labels to dive significantly in the shorter video samples. However, when comparing[1] the metric assessments of the shorter videos with the labels for the same longer segment, the values are not all that different. This is not surprising, since we already know the Live Video Database (LiveVD) video quality is quite consistent from our analysis presented in figure 4.1. Given this consistency, it can only be expected that the temporally independent metrics result in a score that is at least in the same range as the verdict of the longer videos. With all this in mind, the alternative scenario seems far too unlikely to be taken seriously.

In conclusion, the combination of the uniformity of VMAF frame-by-frame scores, and the steady decline of correlation when increasing the length of the analysed video, can only point in the direction that the industry was hoping for.

[1]admittedly, on a glance level

Shorter video segments, up to but possibly not limited to one-second segments, can perfectly well be used to make a valid assessment of the quality. For the experiments described in this thesis, the correlation does not decrease when the video segment is shortened. If anything, the increase in number of video samples caused by the splitting of the longer reference videos, only improves the accuracy of the proposed model.

6.2 Threats to Validity

In this section I discuss what might have gone wrong or why the experiments may not have been conclusive. While most of these points do carry some weight, I would like to preface them with my conviction that none of them form a solid threat to the answer I provided to the research question.

A possible threat is the size of the database used to obtain the metrics and train the model. While Vega et Al used the same database in their research[1], the size of their training set was multiplied by a factor of twelve by having the material not only passed through compression, but also through an artificially lossy network connection. This approach broadened the original set of ten videos to 960 samples in total.

The dataset used by this research is not necessarily that small by comparison, even though I chose to forego the packet loss angle in favour of shortening the samples. The sample cuts make for an even larger set than the one used by Vega et Al, totalling to 1432[1] samples. The problem lies in the fact that none of these samples are labelled individually, inheriting their samples from the original they were cut from instead. This results in quite a large dataset, but in reality this dataset only consists of 80 unique labels, over half of which are in the top fifth percentile.

The effect of this imbalance can most clearly be seen in figure 5.1, more specifically the top-left graph depicting the results for the one second segment experiment. There, many of the over 700 samples generated predictions end up in a discrete set of vertical lines. This is an example of not necessarily overfitting, but the model realising that more accuracy can be reached by only guessing from a set of numbers. This most likely elevates the correlation to a higher level than can reasonably be expected. However, while this is possibly a light boost, the absence of any loss of correlation or accuracy still points to the validity of the premise. The shortened video samples' metrics still hold sufficient predictability to obtain a valid quality assessment.

Apart from a difference in number of samples, my approach also differs in the selection of metrics. As has been described in the method chapter, one metric was omitted in the interest of time, namely Blur Ratio. This metric was originally chosen for its predictive qualities, but seeing that the same kind of accuracy can be reached eve while not implementing the complete proposed set shows that Blur Ratio is not vital to the prediction. Perhaps an arbitrarily

[1]The original dataset is 10 times 8 compression rates, making for 80 samples. These videos are then halved, creating two halves each, making 160 new samples. The videos are also cut into fifths, and finally tenths, making for 400 and 792 (the Blue Sky category does not have a tenth part, removing 8 samples) respectively. These together total 1432 samples.

higher accuracy could have been reached had these metrics been implemented. But for the purposes of this research there really is no point in spending days of development on something that will not help prove or disprove the premise. When applying this method in a practical context however, it would most likely be worth the time to implement all of the metrics, provided that the real-time application can handle the workload.

Blur Ratio is not the only omitted metric, jerkiness was in the end also left unimplemented. The reasoning can be found in the method section. It is here included as a mere reference to note that it is another deviation from the path laid out by Vega et Al, but as discussed in the above, I do not believe it should or could have had any effect on the outcome.

A metric that could have had an impact that was also omitted, is the bitrate the video was compressed with. This metric is included Vega et Al, and when looking at the numbers it is clear to see that there is a strong correlation between bitrate and quality. Which is no surprise, the bitrate compression is literally the device with which the quality degradation is forced upon the material. Then why would I have omitted the metric, if it is so easily obtained and trained for? Out of fear of overfitting. There is such a strong linear relationship between bitrate and quality, that I feared the model would train itself to look at nothing else, the rest of the metric becoming nothing more than a nuance to this one all-deciding factor. Feeling this would be a terrible waste and most probably more a negative impact on the correlation than anything else, I chose to omit it from the training data. When this decision was made, I had the option in mind that if the data turned out to have a disappointing predictive quality, I could always include it later. Once the initial results were in and it became clear that the rest of the metrics had more than enough merit on their own, I never included it. Future implementations of this model that are more focused towards optimal accuracy and less on scientific viability might be more inclined to include bitrate as a computationally cheap accuracy boost.

Finally, the choice of video database is somewhat of a threat. While it has a grand advantage in that it represents a connection to earlier work, it is not exactly representative for common live-streamed material. This category of material is mostly suffused with talking heads, green screen effects, and gaming images. These are not at all represented by the Live Video Database (LiveVD). While there is no sign that this same method should not work in other categories of video, it is worth mentioning that the proposed model has not been tested with the material most appropriate for its practical application.

6.3 Work Left Undone

In this section I will discuss what work I would have liked to do had I had more time in the project. I think these items are not exactly Future Work, since they are most likely not subject enough for an entire new study, but could have shed some extra light on the findings presented in this thesis.

- **Outlier and Category Analysis:**
  One of the recommendations of my daily supervisor was to analyse where the outliers originate. In his own company research[5], he encountered a large difference in quality depending on the category of video material

used. By the same logic, my model might have problems identifying certain types of video material. It has already been shown in figure 4.1 that VMAF has this kind of trouble with a specific video category. Determining whether this has caused the outliers apparent in figures 4.2 and 4.3 might improve the confidence one can realistically place in the proposed model.

- **Expansion of Video Sample Database:**
  As discussed in the previous section, the choice to use LiveVD was primarily motivated by the fact that Vega et Al[1] based their research on the same database. This ensures a measure of reproduction of earlier results that grants a strong basis for this thesis to be built on. However, slow shots of various kinds of scenery are not at all representative of the kind of video material that is typically streamed. Had I been able to spend more time on the thesis, I would have expanded my video database to see whether or not the model would also perform on higher resolution material from different sources, preferably the kind of material that is most popular in interactive streaming environments. While the original research question has been answered, therefore an inference along the lines of "If real-time VQA based on ten second segments is effective, and VQA accuracy doesn't suffer from shortening of training samples, then short sample-based VQA must be effective in interactive streaming environments" can be made. Still, there is no guarantee that the Live Video Database (LiveVD) has some unintelligible quality that allows for short segment analysis, a quality common streaming material may not possess.

- **Real World Application:**
  While a theoretically positive results is really all I set out to accomplish, and an actual working prototype is more than a little beyond the scope, I would have liked to see theory become practice. While I had never had performance in mind while developing the metrics, I was completely taken by surprise when the analysis took multiple orders of magnitude longer than they should have. I had not expected real-time, but neither had I anticipated the necessity to have remote multiple cores work away for literal days to get me my data. Finding out what is the root cause is of this, and improving the solution by such a large margin to remove days of computing time, seems to me like a very valuable experience.

Chapter 7

Future Work

In this chapter I will present some avenues of further research I consider worth pursuing, based on my findings so far and what questions still linger after my research question has been answered. These recommendations are in no particular order.

- **Reconfiguration of Metrics:**
  In their work, Vega et Al[1] motivated their choice of metrics by attempting to make predictions based on each metric individually, and noticing that none of them predict every category/impairment completely. The thought is that combining them would make for more coverage, and therefore improve prediction accuracy.
  In my work, I left out a selection of metrics for various reasons. Yet, my accuracy is comparable to that of Vega and Al. This is most likely partly due to the fact that my material is impaired in only one dimension (compression) while Vega also generated artefacts through packet loss. Another explanation could be that I used a different benchmark, VMAF may for some reason be easier for a model to train with. Despite all this, the fact that I cut 30% of the metrics and still have a very strong correlation, at least implies that not all of them may be necessary. An improvement in performance may be possible if it turns out that only a selection of metrics is sufficient to achieve the same accuracy.

- **Ever Shorter Segments:**
  In the above, I described that my motivation for the choice of sample length was a mathematical one. I wanted each segment to have the same amount of frames, in order to make each sample as similar to the rest as possible. This meant that the length I wanted to cut my samples into had to account for the prime factorisation of 250, resulting in the four lengths I ended up with.
  While I still do believe that a video analysis will give more insight into the quality than picture analysis, meaning frame-by-frame Video Quality Assessment (VQA) is unlikely to provide real insight, I am curious to see where the point of diminishing returns is. If we disregard my attention to detail and simply cut the videos into any number of frames and disregard any leftover bits, we could analyse the effect of ever shorter video lengths on VQA models. This could provide some insight as to what the minimal

length of a video has to be in order to make a valid assessment. I can make a prediction based on which metrics should work at that video length, but a definitive answer can only be reached through experimentation. This work has shown that the limit is at least lower than one second of material.

- **Adaptability over Time:**
  In another work by Vega et Al[9], they explored the applicability of a reduced-reference VQA model with not only real-time, but adaptability over time in mind. This model would be trained on the server, applied on the client side, and continuously trained afterwards. Every so often, the server would send an update to the clients to keep the model up-to-date. This is a great asset in a world that is continuously and unprecedentedly quickly changing. This model is somewhat different than the one used in this research, namely a Restricted Boltzmann Machine (RBM). This model has never before been trained on shorter video segments, I for one would be more than curious to find out if the effect described in this thesis extends beyond a choice of model, and would be as effective in that scenario.

Chapter 8

Acknowledgements

Many people have supported and helped me in the pursuit of graduation. First I would like to thank my daily Supervisor Jeroen Mol, who made sure I built a realistic planning each sprint and was very supportive in the stretches of time that I was not as efficient as I had hoped to be. These thanks are extended out to everyone I interacted with at the Ex Machina Group, I have been surprised by the amount of support and help everyone was eager to extend my way.

Second I would like to thank my Academic Supervisor Dr. Adam Belloum, who helped me solidify my ideas into an actual work of science. His attitude towards taking a little longer than originally planned helped put me at ease when I was certain all was lost.

Another quick note should be made of the much aforementioned Dr. Maria Torres Vega, whose research did not only serve as the inspiration for my own work, but was willing to have a sit-down with me and discuss my plans. She put me on the right track weeks before I would have found it.

Finally, I would like to thank all of my friends and family who kicked my ass to actually get to work on this thing. The "work-from-home" ethic that the pandemic brought has been hard on me, and you made it bearable. In no particular order, my thanks go out to Eva, Victor, Vera, Tessel, Maartje, Floor, Dominique, and my parents. If I forgot anyone, I hope you'll find as much forgiveness in your heart as you have fuelled mine with determination to finally graduate.

Bibliography

[1] Maria Torres Vega, Decebal Constantin Mocanu, Stavros Stavrou, and Antonio Liotta. Predictive no-reference assessment of video quality. *Signal Processing: Image Communication*, 52:20–32, 2017.

[2] Stephen Wolf and MH Pinson. Video quality model for variable frame delay (vqm-vfd). *US Dept. Commer., Nat. Telecommun. Inf. Admin., Boulder, CO, USA, Tech. Memo TM-11-482*, 2011.

[3] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*, 6:2, 2016.

[4] Zhi Li, Kyle Swanson, Christos Bampis, Lukáš Krasula, and Anne Aaron. Toward a better quality metric for the video community. *The Netflix Tech Blog*, page 2, 2020.

[5] Jeroen Mol. Harder, better, faster, cheaper — optimizing video bitrate for ultra low latency live content. *medium.com/exmachinagroup*, 2019.

[6] Jiang Chu, Qiang Chen, and Xichen Yang. Review on full reference image quality assessment algorithms. *Application Research of Computers*, 31(1):13–22, 2014.

[7] Hamid R Sheikh and Alan C Bovik. Image information and visual quality. *IEEE Transactions on image processing*, 15(2):430–444, 2006.

[8] Songnan Li, Fan Zhang, Lin Ma, and King Ngi Ngan. Image quality assessment by separately evaluating detail losses and additive impairments. *IEEE Transactions on Multimedia*, 13(5):935–949, 2011.

[9] Maria Torres Vega, Decebal Constantin Mocanu, Jeroen Famaey, Stavros Stavrou, and Antonio Liotta. Deep learning for quality assessment in live video streaming. *IEEE signal processing letters*, 24(6):736–740, 2017.

[10] Nabajeet Barman, Emmanuel Jammeh, Seyed Ali Ghorashi, and Maria G Martini. No-reference video quality estimation based on machine learning for passive gaming video streaming applications. *IEEE Access*, 7:74511–74527, 2019.

[11] Nabajeet Barman, Saman Zadtootaghaj, Steven Schmidt, Maria G Martini, and Sebastian Möller. Gamingvideoset: a dataset for gaming video streaming applications. In *2018 16th Annual Workshop on Network and Systems Support for Games (NetGames)*, pages 1–6. IEEE, 2018.

[12] Yuanyi Xue, Beril Erkin, and Yao Wang. A novel no-reference video quality metric for evaluating temporal jerkiness due to frame freezing. *IEEE Transactions on Multimedia*, 17(1):134–139, 2014.

[13] Kalpana Seshadrinathan, Rajiv Soundararajan, Alan Conrad Bovik, and Lawrence K Cormack. Study of subjective and objective quality assessment of video. *IEEE transactions on Image Processing*, 19(6):1427–1441, 2010.

[14] Vinton Cerf, Yogen Dalal, and Carl Sunshine. Specification of internet transmission control protocol. *Internet History [6]. Ronda Hauben. From the ARPANET to the Internet. TCP Digest (UUCP)*, 1974.

[15] Jelle Manders. Msc software engineering thesis. https://github.com/JelleManders/MSc_SE-Thesis, 2021.

[16] Jing Hu and Herb Wildfeuer. Use of content complexity factors in video over ip quality monitoring. pages 216–221, 2009.

[17] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[18] Chung-Jr Lian, Yu-Wen Huang, Hung-Chi Fang, Yung-Chi Chang, and Liang-Gee Chen. Jpeg, mpeg-4, and h.264 codec ip development. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 2*, DATE '05, page 1118–1119, USA, 2005. IEEE Computer Society.

[19] Cristian Perra. A low computational complexity blockiness estimation based on spatial analysis. pages 1130–1133, 2014.

[20] M Torres Vega. Cognitive management and control of high speed indoor optical wireless networks. 2017.

[21] Min Goo Choi, Jung Hoon Jung, and Jae Wook Jeon. No-reference image quality assessment using blur and noise. *International Journal of Computer Science and Engineering*, 3(2):76–80, 2009.

[22] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013.

[23] Suramya Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.