


Article

On the Identification and Prediction of Stalling Events to Improve QoE in Video Streaming

J.-M. Martínez-Caro and M.-D. Cano * 

Department of Information Technologies and Communication, Universidad Politécnica de Cartagena, Plaza del Hospital 1, 30202 Cartagena (Murcia), Spain; josem.martinezcaro@upct.es

* Correspondence: mdolores.cano@upct.es

Abstract: Monitoring the Quality of user Experience is a challenge for video streaming services. Models for Quality of User Experience (QoE) evaluation such as the ITU-T Rec. P.1203 are very promising. Among the input data that they require are the occurrence and duration of stalling events. A stalling event is an interruption in the playback of multimedia content, and its negative impact on QoE is immense. Given the idiosyncrasy of this type of event, to count it and its duration is a complex task to be automated, i.e., without the participation of the user who visualizes the events or without direct access to the final device. In this work, we propose two methods to overcome these limitations in video streaming using the DASH framework. The first method is intended to detect stalling events. For simplicity, it is based on the behavior of the transport layer data and is able to classify an IP packet as belonging (or not) to a stalling event. The second method aims to predict if the next IP packet of a multimedia stream will belong to a stalling event (or not), using a recurrent neural network with a variant of the Long Short-Term Memory (LSTM). Our results show that the detection model is able to spot the occurrence of a stalling event before being experienced by the user, and the prediction model is able to forecast if the next packet will belong to a stalling event with an error rate of 10.83%, achieving an F1 score of 0.923.



Citation: Martínez-Caro, J.-M.; Cano, M.-D. On the Identification and Prediction of Stalling Events to Improve QoE in Video Streaming. *Electronics* **2021**, *10*, 753. <https://doi.org/10.3390/electronics10060753>

Academic Editor: Taeshik Shon

Received: 6 November 2020

Accepted: 16 March 2021

Published: 22 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: stalling events; QoE; video streaming; DASH; deep learning

1. Introduction

Video streaming services have become very popular. In 2020, the revenue of video streaming platforms amounted to US\$25,894 million, and the number of users is expected to amount to 1.3 billion by 2024 [1]. In this context, the entertainment business is actively investing in content creation, acquisitions, and also in technology. As a matter of fact, video streaming platforms are showing that “dynamic markets can benefit consumers with lower prices and better quality” [2].

When it comes to evaluating quality, two concepts arise, Quality of Service (QoS) and Quality of user Experience (QoE). The former is mainly based on well-known network metrics (delay, jitter, etc.). The latter covers a broader range of metrics, trying to infer a user’s perception of the level of quality provided. Thus, QoE has been adopted by the scientific community as the best approach to evaluate the performance of video streaming services. Most multimedia services consider visual quality, loading time, stalling events, and overall quality as the four basic metrics for evaluating the quality of the playback [3]. Stalling events are particularly relevant, considered by many as the most degrading factor in QoE [3–8]. In general, it can be stated that the more stalling events, the lower the quality. Stalling events are caused by the exhaustion of the receiving buffers. They are experienced by users as sudden stops of the video playback, with a duration of the order of seconds or even minutes. Several works have studied the impact of stalling events and how these events can be estimated or predicted [3–9].

In previous work [5], we validated the objective model for QoE estimation in video streaming services proposed by the ITU-T Rec. P-1203 [10] under an LTE scenario via

experimentation and emulation. As in [4], stalling events had the most negative impact on QoE measurements. In order to solve this issue, Casey and Muntean [7] proposed a solution called DASH-based Quality-oriented Video Delivery (DQVD) to reduce stalling events during video playback under a heterogeneous architecture. The idea was based on using dynamic wireless interface selection. This approach activates a second connection channel on mobile devices to download the necessary content and temporarily improve performance. The solution monitors the playout buffer levels and if it cannot be replenished quickly enough using a single connection to avoid the video stalling, the cellular connection is re-established to increase overall throughput to the buffer. When conditions improve to the point at which the WiFi connection can sustain replenishing the playout buffer on its own, the cellular connection is disconnected again.

From a different perspective, Tao et al. [3] presented a novel technique for processing data collected by the user during the playback of multimedia content over mobile networks. The authors applied feature selection and deep learning methods with seven layers in order to predict the subjective QoE scores based on the selected parameters (visual quality, loading, stalling, and overall quality score). Similarly, Seufert et al. [4] concluded that “stalling events due to re-buffering still are by far the worst QoE degradation” and focused their work on the prediction of stalling events in real-time. They proposed a machine learning-based approach for monitoring QoE-relevant metrics in YouTube, which was able to predict the occurrence of stalling events in real-time from such basic features. For training and testing, the authors generated a dataset with 4714 YouTube video sessions through the Google Chrome web browser, using HTTP-adaptive streaming (HAS).

Following the same trend, Bampis et al. [11] presented a QoE prediction model based on Support Vector Regression (SVR). It uses three different features to make predictions: the density of stalling events, time since the last stalling event, and the quality of the video. The model was tested offline on a Netflix database under the HAS over TCP protocol. Wassermann et al. [12] studied the behavior of QoE under LTE, taking data from the network layer and the context (screen size, orientation, playing mode, audio volume, etc.) when playing YouTube videos. The authors used the ITU-T P.1203 model to estimate the Mean Opinion Score (MOS) value and employed different ML algorithms such as Random Forest (RF), Decision Tree (DT), Supported Vector Machine (SVM), K-Nearest Neighbors (KNN), or Naïve Bayes (NB) to predict the QoE value from a selection of 30 features. Gutterman et al. [13] proposed the use of Adaptive Bit Rate (ABR) through DASH and WiFi connection, knowing at every moment the state of the buffer. The multimedia content was divided into chunks using the *ChunkDetection* algorithm; the features of the chunks were obtained and from these features, the QoE value was predicted using the RF algorithm. Although Krishnamoorthi et al. [14] used HAS as the streaming protocol, features were obtained through chunks from YouTube videos as in [13]. The mechanism had access to the buffer state and used two methods for prediction: a threshold-based classifier and a machine learning model based on DT and SVM. Finally, Mazhar et al. [15] used a supervised machine learning algorithm to predict QoE values from the C4.5 DT and Adaboost to reduce misclassifications. For this purpose, the authors extracted 226 features from the reception time windows and from the received packets in the network and transport layer. Tests were carried out using the Quick UDP Internet Connections (QUIC) protocol at the transport layer. Finally, Abar et al. [16] proposed a new solution based on Software Defined Network (SDN), DASH, and machine learning to reduce the occurrence of stalling events during playback when a large volume of clients was connected to the platform. The model estimated a threshold value of QoE to offer good quality in a heterogeneous network.

In sum, the identification, measurement, and prediction of stalling events in real-time are highly valuable tools and this has become a very active area of research. We could improve accuracy when applying QoE parametric models. For instance, the ITU-T Rec. P.1203 [5,17] presents an objective model for estimating QoE in video streaming services. However, one of the required input parameters is the number of stalling events

that have occurred and their duration. Unless there is a method to identify and measure stalling events in real-time, the application of the ITU-T Rec. P.1203 needs to be done a posteriori. Besides, the method should not be based on detection by active users, i.e., the user that is consuming the video. In addition, the ability to predict when these events are going to happen would allow the performing of corrective actions to improve the QoE, e.g., proactively lowering the video resolution or allocating more resources to specific microservices. Further, new prediction models should be studied and their outcomes evaluated to take full advantage of the deep learning potential [18–23].

Differing from other works dealing with stalling events by studying the behavior of the playback buffers or with access to this type of information, it is our initial hypothesis that the information contained in the network or transport communication layers (or a combination of both) could be good enough to either detect the existence of stalling events or to predict their occurrence (or both). One advantage of such an approach is that the solution would not be affected by the use of encryption techniques in upper communication layers or by the selected monitoring point in the network in case we use end-to-end transport layer information.

Consequently, the contribution of this paper is twofold:

1. We propose a method to classify IP packets from a video streaming flow as belonging (or not) to a stalling event.
2. We present a machine learning-based model to predict if the next packet will be a stalling event (or not); specifically, we use a Recursive Neural Network (RNN) using Long Short-Term Memory (LSTM).

To carry out this study, we employed a dataset obtained from an emulated Long-term Evolution (LTE) scenario that was validated in [5,24]; video streaming transmission and reception were executed in real devices, whereas the LTE network was simulated with the NS3 network simulator. Dynamic adaptive streaming over HTTP (DASH) was used. In the light of the results, both models are very promising in order to automate the detection and measurement of stalling events to facilitate the use of QoE monitoring. Although many players available on the market offer an API for these events and many others, it is important to realize that we are proposing obtaining these data from the network, without the need of accessing a client's player.

The rest of the document is organized as follows. Section 2 describes the research methodology, including data acquisition, imbalanced datasets, and data preprocessing. The classification model is introduced in Section 3, as well as its performance results. Then, we describe the proposed forecast model in Section 4, which also includes the performance evaluation. Finally, the paper ends summarizing the most important findings.

2. Methodology

2.1. Data Acquisition

The dataset was obtained in an LTE emulated deployment described and validated in [5,24] (see Figure 1). The scenario was composed of three devices: a real video server, an emulated LTE network, and a client. The video server was implemented with the Wowza Streaming Engine to serve live video streaming. The LTE network was simulated with the NS3 network simulator. Finally, a real client consumed the video streaming through his/her web browser. The video transmission was performed using DASH.

DASH or MPEG-DASH [25] is a standard for video transmission. It employs the TCP protocol at the transport layer and the HTTP framework to deliver the content at the application level [26], thus not being affected by Network Address Translations (NATs) or firewalls. One of its key features is that it is the client who chooses the bitrate that better adapts to the conditions. Briefly, the server divides the video content into smaller segments (usually from 2 s to 10 s). Then, it encodes them at different compression levels, i.e., providing different bitrates. All these segments are then available to the client. Multiple bit-rates and quality levels between segments allow meeting with the available

bandwidth, maximizing the use of the network in a dynamic way [7]. DASH uses TCP at the transport layer.

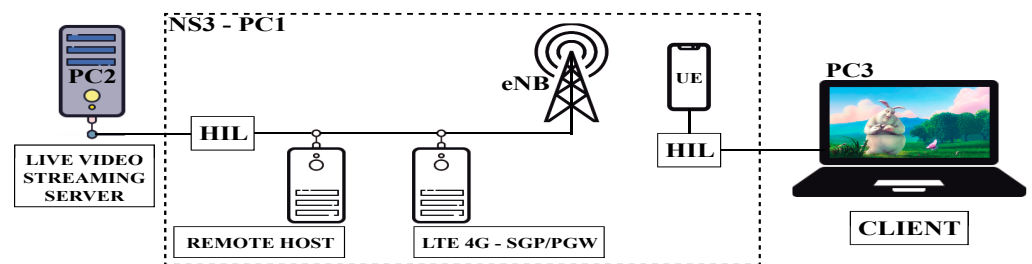


Figure 1. Testbench to obtain the dataset.

In all tests, an H.264 sequence of the “Big Buck Bunny” cartoon movie was sent, with a duration of 180 s and a resolution of 720p. Network delays varied from a minimum value of 0 to a maximum value of 150 ms in steps of 25 ms. Experiments were repeated 10 times to prevent unexpected singularities. Therefore, a total number of 70 captures were collected. On the client side, we employed the Wireshark tool [27] to collect both the received and sent packets from/to the video server. The total number of received packets was 1647770, of which 91,543 packets were identified by visual inspection as belonging to stalling events; thus, the imbalanced ratio was 1:18. Table 1 shows the characteristics of the emulated environment. From the collected data, we focused on the received data at the network and transport layers, corresponding to the TCP and IP protocols, respectively. Table 2 shows all the original features in our dataset.

Table 1. Characteristics of the Emulated Environment.

Feature	Description
Number of devices	3
Devices	1. PC2. A live video streaming server (Wowza Streaming Engine) 2. NS3-PC1. An emulated LTE Network with NS3 3. PC3- Video Client (Web browser and Wireshark)
Codec	H.264/MPEG-4 AVC
Audio encoding/channels	AAC/stereo
Resolution	720p (1280 × 720)
Coding bitrate (Kbps)	2628
Video Transport Protocol	DASH
Propagation Model	Nakagami (m = 5)
Simulations	10 for each configured delay
eNB operation power	100 dBm
Antenna Type	Cosine height 1.5 m
eNB noise	2 dB
UE noise	7 dB
UE operation power	26 dBm, height 2.5 m, distance 50 m

2.2. Imbalanced Datasets

It can be assumed that network performance is generally favorable and network impairments do not occur continuously. Under this assumption, a dataset obtained capturing network traffic will be usually imbalanced. For instance, stalling events could occur occasionally, thus becoming a minority class (not the natural state). However, the minority class would be the most important from a QoE perspective because is deeply linked with a behavior that needs to be avoided (or whose impact needs to be limited).

Table 2. Extracted Features for each Packet in the Dataset.

Feature	Description
Tcp seq num	TCP sequence number
Tcp win siz	TCP window size
Tcp fast ret	Fast retransmitted packet
Tcp ack dup	Packet with ACK duplicated
Tcp prev NC	Previous packet is lost
Tcp pkt ooo	TCP Packet out-of-order
Pkt len	IP packet length
Ip len	IP header length
Tcp hea len	TCP header length
Tcp flag ack	TCP ACK flag
Tcp flag urg	TCP URG flag
Tcp flag psh	TCP PUSH flag
Tcp flag rst	TCP RESET flag
Tcp flag syn	TCP SYN flag
Tcp flag fin	TCP FIN flag

In this work, we deal with an imbalance dataset obtained from [5,24]. Despite being imbalanced, our goal is to use it in order to propose an efficient mechanism able to classify a packet as belonging (or not) to a stalling event and to predict if the next packet will belong (or not) to a stalling event. Whereas most of the regular learning algorithms generalize well into balanced datasets, imbalanced datasets need to be addressed differently [28,29], otherwise the underrepresented minority class would be weaker than the majority class.

In [30], Sun et al. presented a review of the imbalanced classification problems in different application domains and reported several solutions. Further, the work done by He et al. [31], Krawczyk [32], and Vluymans [33] explained how to deal with imbalanced data and data skewness focusing on computational efficiency and adaptive methods. In this sense, there is a common agreement that preprocessing is required in this situation. Most preprocessing methods are available in Scikit-Learn [34], a simple Open-Source tool that also provides predictive analysis using additional libraries such as NumPy, SciPy, and matplotlib. The first preprocessing step is to transform the original dataset by reducing the number of features. It can be assumed that very similar features are redundant and only increase the dimension and complexity of the dataset. Consequently, some of them could be ignored. Some techniques employed for feature selection are correlation, Recursive Feature Elimination (RFE), Principal Feature Analysis (PFA), Independent Component Analysis (ICA), and Fisher's Linear Discriminate Analysis (LDA) [35,36]. These methods follow a similar process: a subset of features is created and one feature is excluded from the group at each iteration. In contrast, Principal Component Analysis (PCA) and Forward Selection Component Analysis (FSCA) reduce the number of features by approximation, reconstructing the majority of the original information [37].

2.3. Processing Data

For feature selection, we applied PFA. PFA computes the n -dimensional correlation matrix, showing the correlation coefficients between two features. From the calculated correlation matrix, the method obtains a series of V_i vectors that represent the i -th projection of the feature vector X for a lower subset of features. The higher the correlation, the higher the absolute value weight vector V_i . Once the subspace dimension has been set, 1D for our case, PFA clusters highly correlated features together using K-Means. For each cluster, it chooses the feature that best represents the set containing less redundant and more dispersed information. This allows us to maintain a good representation of the original data set with a complexity similar to PCA [22].

After preprocessing the dataset and using cross-validation techniques, the most relevant features in the appearance of stalling events belong to the transport communication layer, specifically the TCP SYN (Transport Control Protocol SYNchronization) flag, the

TCP sequence number, and the TCP window size, as shown in Figure 2. It is intuitive to think that a video streaming transmission protocol over reliable transport such as DASH will have an effect on the transport layer, so the feature selection process corroborates this approach.

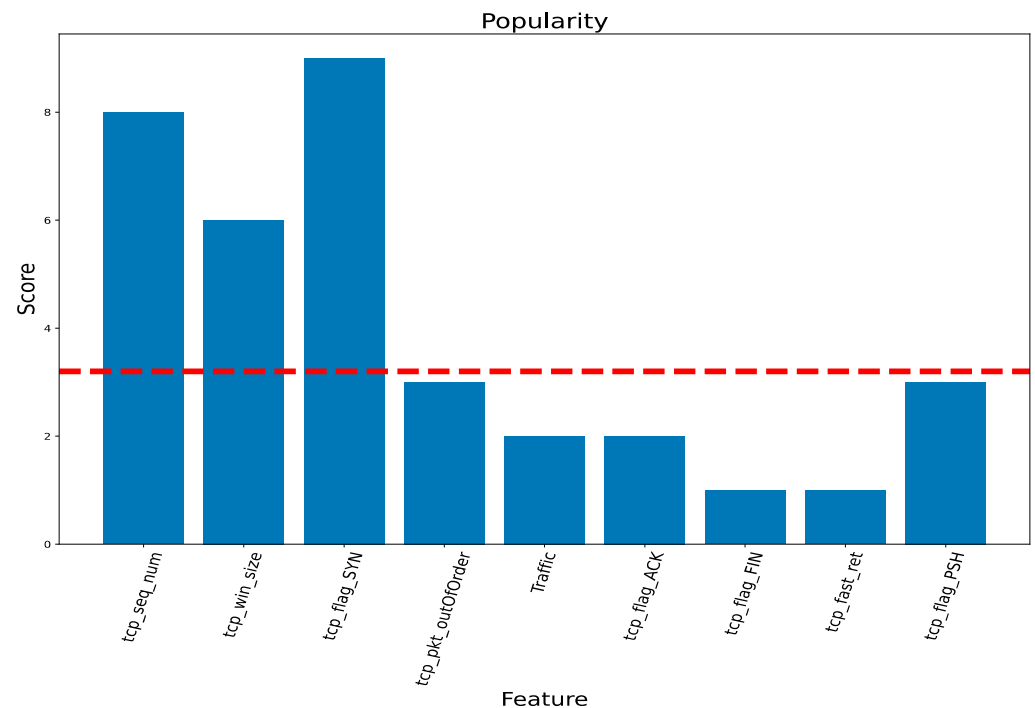


Figure 2. Distribution of top features.

2.4. Learning Metrics

If a raw imbalanced dataset is trained, the result may achieve high accuracy, but the model has a “naive” result. The accuracy is not appropriate for imbalanced problems because it is biased to the majority class [19]. For binary classification problems, as in this work, the confusion matrix consists of four components (Table 3): True Positive (TP), when predicted and real values are both positive; True Negative (TN), when predicted and real values are negative; False Positive (FP), when the prediction is positive and real data are negative, and False Negative (FN), when the prediction is negative and real values are positive. Our models will try to maximize TP and TN and minimize FP and FN.

Table 3. Confusion Matrix.

	Predicted Positive	Predicted Negative
Real Positive	True Positive (TP)	False Negative (FN)
Real Negative	False Positive (FP)	True Negative (TN)

In addition, new metrics can be derived from the confusion matrix to evaluate model performance using imbalance datasets. Traditionally, the learning process has adopted accuracy as the metric to fit and improve the performance of the model. Accuracy is usually defined as the number of correct predictions made by the model over all types of predictions. The use of Accuracy in imbalanced problems is discouraged because it is not very precise. As an alternative, Precision, Recall, F1, and AUC are recommended. Precision measures the exactness as the number of positive samples tagged without errors. Therefore, it is calculated as the ratio between samples correctly assigned to the positive class and all samples classified as positive (1). Recall quantifies the completeness. It is assessed as the

fraction of the total amount of relevant instances that were actually retrieved (2), i.e., how many positive samples were identified among all real positive samples. The F1-Score (F1) is a simpler metric that joins Precision and Recall as a harmonic mean as shown in (3). F1 is very useful when we look for an equilibrium between both metrics and we have a large number of negative samples. Consequently, it is possible to represent the robustness of the classifier in a single metric, quantifying the correct number of non-lost instances. Finally, the Area Under the Curve (AUC) evaluates the model according to the distinction among classes (4); the higher the value, the greater the separation of the classes. The range for all previous metrics is [0, 1] [38].

$$\text{Precision} = \frac{TP}{TP + FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

$$F1 = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}} \quad (3)$$

$$AUC = \frac{TP(TN + FP) + TN(TP + FN)}{2(TP + FN + TN + FP)} \quad (4)$$

3. Threshold-Based Model to Classify Stalling Events

In this section, we describe and evaluate the proposed model to classify packets as belonging (or not) to stalling events.

3.1. Model Description

Instead of applying a machine learning method for classification, we will address this goal by observing only the behavior of the TCP SYN flag in the multimedia stream. The TCP SYN flag was identified previously as the most relevant feature in the feature selection process. Therefore, we propose a sliding window-based, simple (yet efficient) algorithm (see Algorithm 1).

Let us assume an initial window size t , where t is much smaller than the total number of captured packets in the dataset T ($t \ll T$). Please note that in the paper, a packet is equivalent to a captured IP packet with its corresponding header and payload. Packets will be numbered using the sub index i , from $i = 1$ to $i = T$. Then, we define a vector L of size N , where N (5) represents how many consecutive windows with different packets are contained in the dataset. L is used to store the number of packets with the TCP SYN flag active within the same sliding window. From the stored values in L , a threshold will be defined so that if this threshold is exceeded, then packet i will be classified as belonging to a stalling event.

Let us give an example. Let us assume a sliding window with a size equal to three packets ($t = 3$) and a dataset with a size of six packets ($T = 6$). Then, applying (5), vector L would have four elements ($N = 4$).

$$N = T - t + 1 \quad (5)$$

That is, there are four consecutive windows containing different packets through the dataset (see Figure 3). The first sliding window w_1 covers packet $i = 1$ to packet $i = 3$ (both included). If the sum of TCP SYN flags in these packets is 3, then $L[0] = 3$. For the second iteration, the sliding window w_2 covers packet $i = 2$ to packet $i = 4$ (both included). If the sum of TCP SYN flags in these packets is 2, $L[1] = 2$, and so on, until the last sliding window w_4 (e.g., $L = [3, 2, 2, 1]$). Once vector L has been computed, the average value (6) and the standard deviation (7) of all its elements are obtained. We define the threshold th as shown in (8). In our example, th would be equal to 2.71 (the average value is equal to 2 and the standard deviation is equal to 0.71). In order to classify a packet i as belonging to a stalling event (or not), we observe the positions in vector L impacted by this packet. For instance, the first packet ($i = 1$) appears only in the first window w_1 , so we only observe

$L[0]$. If $L[0] > th$ then packet i belongs to a stalling event. In the example, $3 > 2.71$ so the first packet is assumed to belong to a stalling event. For the second packet ($i = 2$), we know that this packet appears in the first two sliding windows w_1 and w_2 , so we need to compute the average value of $L[0]$ and $L[1]$; in this case, 2.5. Then again, if $avg(L[0], L[1]) > th$ this packet would belong to a stalling event; in this case, $2.5 < th$, so the second packet ($i = 2$) does not belong to a stalling event.

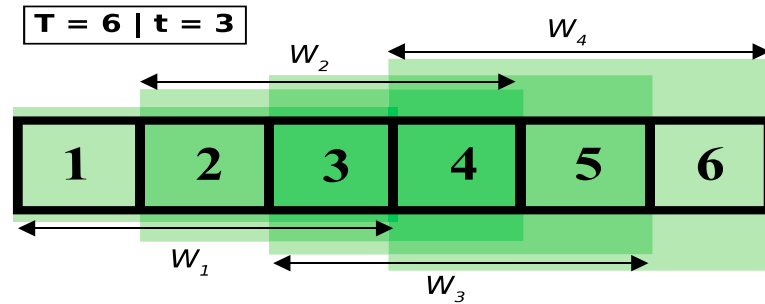


Figure 3. Sliding window example.

Taking into account that any packet will appear in t sliding windows, except the first and last $(t-1)$ packets, packet i (where $t \leq i \leq T - t + 2$) belongs to the sliding windows from $w_{(i-t+1)}$ to w_i . Consequently, to classify packet i , the positions $L[i-t]$ to $L[i-1]$ will be evaluated (9).

$$avg(L) = \sum_{n=1}^N \frac{L[n]}{N} \tag{6}$$

$$std(L) = \sqrt{\frac{1}{N} \sum_{n=1}^N (L[n] - avg(L))^2} \tag{7}$$

$$th = avg(L) + std(L) \tag{8}$$

$$Frame\ i \begin{cases} \text{if } i < t & \text{eval. from } L[0] \text{ to } L[i-1] \\ \text{if } t \leq i \leq T-t+2 & \text{eval. from } L[i-t] \text{ to } L[i-1] \\ \text{if } i > T-t+1 & \text{eval. from } L[i-t+1] \text{ to } L[t-1] \end{cases} \tag{9}$$

Algorithm 1 Threshold-based classification model

1. **procedure** DATASETSPLITTING (data(ex, cap))
2. *experiment* = 150
3. **while** *experiment* ≥ 0 **do**
4. *capture* = 10
5. **while** *capture* ≥ 0 **do**
6. **if** *capture* > 2 **then**
7. *train* = data(*experiment*, *capture*)
8. **else**
9. *test* = data(*experiment*, *capture*)
10. **end if**
11. *capture* = *capture*−1
12. **end while**
13. *experiment* = *experiment*−25
14. **end while**
15. **end procedure**

Algorithm 1 Cont.

```

16. procedure GETTHRESHOLDVALUE(train, t)
17.   dataX = []
18.   T = len(train)
19.   while  $i \leq T - t + 1$  do
20.     a = train[i: (i + t)]
21.     dataX = a.append(a)
22.     i = i + 1
23.   end while
24.   dataTrain = sum(dataX, 1)
25.   th = mean(dataTrain) + std(dataTrain)
26.   return th
27. end procedure
28. procedure SLIDINGWINDOWS(test, t)
29.   dataY = []
30.   T = len(test)
31.   while  $i \leq T - t + 1$  do
32.     a = test[i: (i + t)]
33.     dataY = a.append(a)
34.     i = i + 1
35.   end while
36.   dataTest = sum(dataY, 1)
37.   return dataTest
38. end procedure
39. procedure CHECKINGTHRESHOLD(th, a)
40.   if a.mean()  $\geq$  th then
41.     st_event.append(1)
42.   else
43.     st_event.append(0)
44.   end if
45. end procedure
46. procedure STALLINGDETECTION(dataTest, th, t)
47.   st event = []
48.   T = len(dataTest)
49.   while  $i \leq T + t$  do
50.     if  $i < t$  then
51.       a = dataTest[0: i]
52.       CheckingThreshold(th, a)
53.     else if  $i > T$  then
54.       a = dataTest[i-t:]
55.       CheckingThreshold(th, a)
56.     else
57.       a = dataTest[i-t: i]
58.       CheckingThreshold(th, a)
59.     end if
60.     i = i + 1
61.   end while
62. end procedure

```

3.2. Results

We compared the frame classification obtained with this algorithm with the stalling events registered via experimentation in [5]. It can be observed that the stalling events in the test-bed undergo a short time-delay compared to the classification results obtained with the algorithm (see Figure 4). This fact can be explained because of the buffering effect. As shown in Figure 4, the model detected the stalling event before the moment it was registered as such by the user (please take into account that real measurements were done

by the final user who consumes the video streaming). From our point of view, this can be seen as an advantage, since proactive actions could be applied in the service/system in order to avoid the stalling event before it is experienced by the final user.

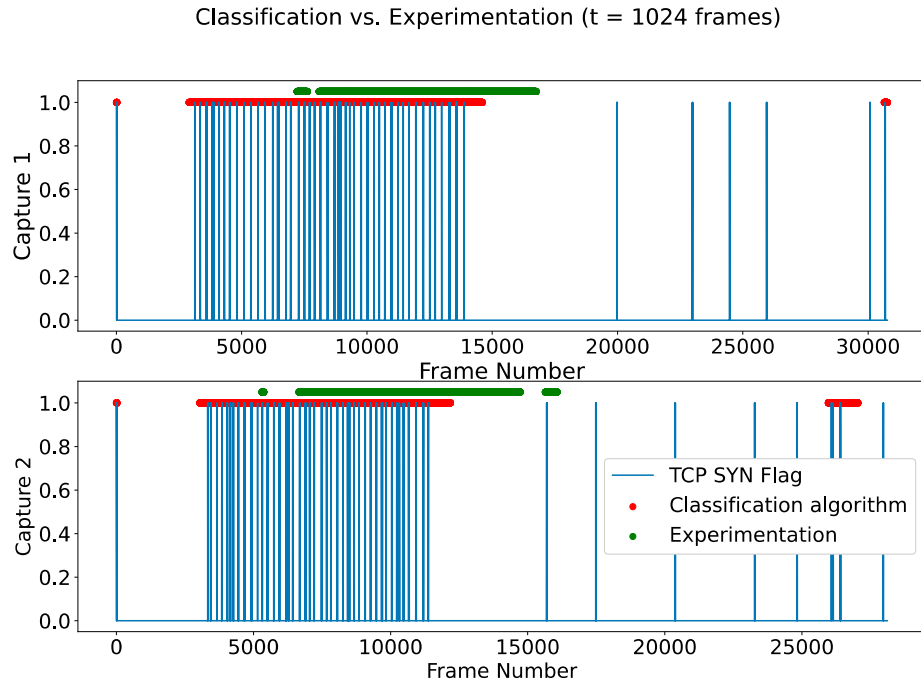


Figure 4. Example of packet classification vs. real occurrences using the threshold-based classification model.

In addition, we tested this process using different window sizes, from $t = 128$ to $t = 1024$ packets and observed the AUC score. The best performance was obtained for larger window sizes (1024 packets). For small values of t , overfitting occurred due to insufficient data. This effect was clearly avoided by increasing t because the amount of false positives was highly reduced (see Figure 5).

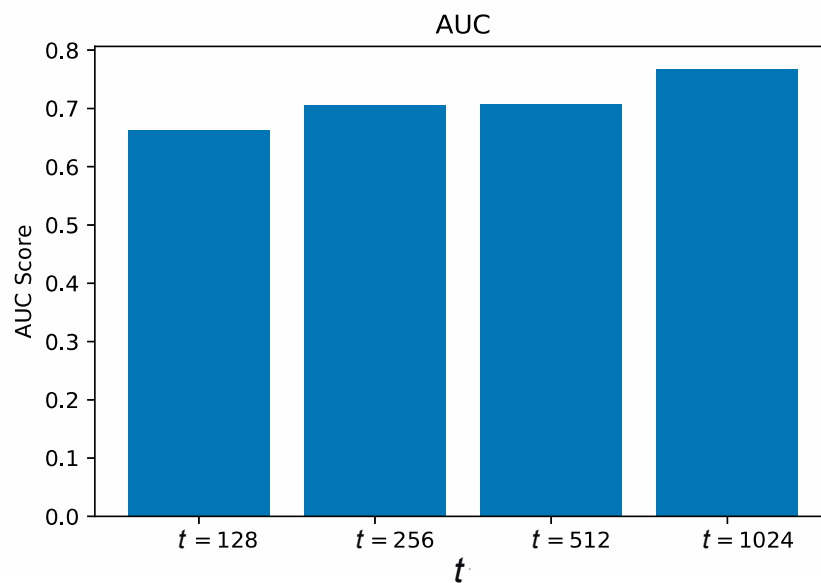


Figure 5. Performance evaluation of the threshold-based classification model for different windows sizes (t).

4. Forecasting Model to Predict Stalling Events

In this section, we describe and evaluate the proposed forecasting model for stalling events. The forecasting model aims to predict the behavior of future packets. Two classes are defined to solve the binary prediction challenge: 1 for stalling event and 0 for non-stalling (normal operation).

4.1. Model Description

One of the most popular approaches for forecasting is the use of artificial Neural Networks (NN), considered a deep learning technique. Despite the advances in NN, their use with imbalanced datasets is not completely mature. Johnson and Khoshgoftaar's survey [39] described the implementation details of different case studies, highlighting their strengths and weaknesses in dealing with imbalanced data in aspects related to complexity and the architecture of the data, performance evaluation, or ease of use. Of particular interest for our study is the RNN using LSTM. LSTM was introduced by Hochreiter and Schmidhuber in [18] to overcome and explore gradient problems and has gained popularity in predictive analysis due to its capacity to predict time series data [19–21].

Consequently, our proposal is based on RNN since the state of the memory influences the decision making process. In particular, it will use LSTM, solving the vanish gradient problem in the RNN with back propagation through time. As in the classification problem, we suggest the use of a sliding window of size t to make decisions. In this case, the most recent t packets will be used to predict if the next packet will belong to a stalling event or not. The selected t value for this specific case is 1024 packets due to the overfitting effect observed previously for smaller values. We implemented the Attention enhanced Bidirectional Long Short—Term Memory (ABi-LSTM) model [38] with blocks of long-term memory and back and forward propagation (see Algorithm 2). These memory blocks use gates, inputs, and outputs to manage the state and obtain the prediction of the future packet. Finally, the forget gate is responsible for updating the memory. Observe that the proposed model is based on the classification of sequences using two LSTMs in the input sequence as depicted in Figure 6. The first adds the sequence as such, and the second an inverted copy of the sequence, providing an additional context giving faster and deeper learning [20,40,41].

Algorithm 2 ABi-LSTM prediction model

procedure DEFINITIONANDTRAINING

(trainX[k] trainY[k], testX[k], testY[k])

1. model = Sequential();
 2. **Bidirectional(LSTM(activation = 'relu'));**
 3. **Dropout(0.3);**
 4. Dense(activation = 'sigmoid');
 5. model.compile(loss = binary_crossentropy, metrics = [accuracy,F1,Recall,Precision], opt = SGD(lr = 0.0001))
 6. model.fit(trainX[k],trainY[k])
 7. **return model.evaluate(testX[k],testY[k])**
 8. **end procedure**
 9. **procedure CV ABi-LSTM(data, K)**
 10. Partition randomly the data in K subsets (DT_1, \dots, DT_k)
 11. **for** k = 1 to K:
 12. Construct the k th training set with K—1 subsets
 13. Construct the k th testing set with 1 subset, different to training set.
 14. result[k] = DEFINITIONANDTRAINING (trainX[k], trainY[k], testX[k], testY[k])
 15. **end for**
 16. Represent results for Loss Function, Accuracy, Precision, Recall and F1 Score for each fold.
-

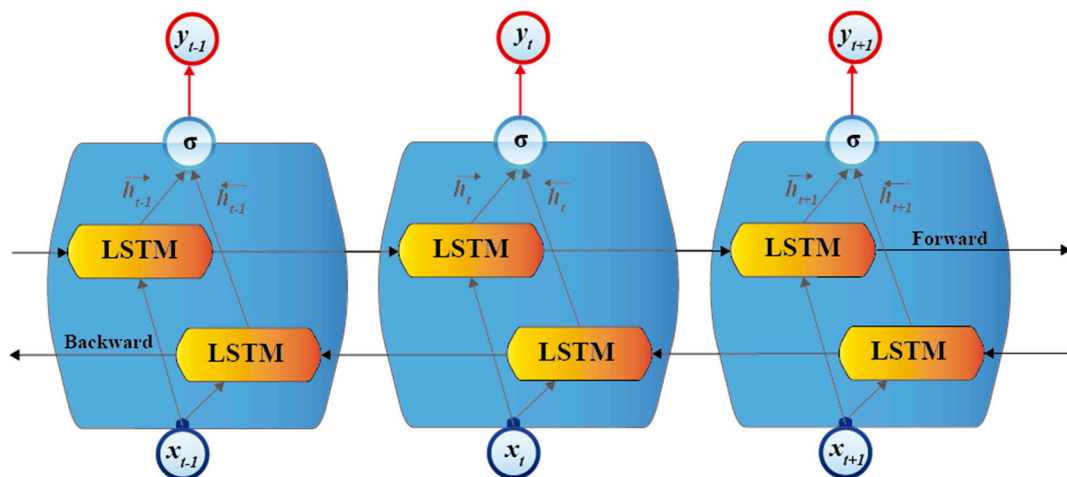


Figure 6. ABI-LSTM prediction model.

The training process updates the weight coefficient w of the model using Stochastic Gradient Descent (SGD) (10) and finds the optimal weights to minimize the loss function. Another factor is the learning rate (η) that comes into play managing the learning speed and reducing the training overfitting [38]. In order to reduce overfitting, this algorithm applies regularization techniques by penalizing the network weights on the final objective function. The regulation factor (0.3) has been carefully selected to obtain a balanced classification using the Dropout technique; this implementation is the most powerful in deep learning. Its purpose is to drop units during forward propagation or backward propagation so that the network does not adapt to a specific set of features.

The proposed model has an entrance layer, an exit layer and a hidden layer; the connections between the different layers are bidirectional. Each layer has different connection units; I units for the input layer, H units for the hidden layer, and K units for the output layer. The computational complexity of a bidirectional LSTM is given by $O(2n)$, where n is the number of weights [18]. For our case, n can be expressed as shown in (11), with $I = 1024$, $H = 1024$, and $K = 1$. For each fold, the average training time is 2650 s, with 60 s for evaluation. For the fitting, we use 25 epochs and the *batch_size* value is 1024. Table 4 shows the characteristics of the equipment used to run the model.

$$w \rightarrow w + \Delta w; \quad \Delta w = -\eta \nabla J(w) \quad (10)$$

$$n = 4IH + 4H^2 + 3H + HK \quad (11)$$

Table 4. Details of the Server used for Simulation.

Feature	Description
Description	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20 GHz
CPU(s)	60
Thread(s) per core	2
Core(s) per socket	10
Core(s) per socket	2
UE operation power	26 dBm, height 2.5 m, distance 50 m

4.2. Results

In order to obtain the results for this model, we use k -Fold Cross Validation as the statistical method to estimate the ability of the model raised on different subsets of data within a single dataset. The procedure has a single parameter called k that refers to the number of groups into which a given data sample will be divided. For the evaluation of each group, the group is taken as a testing dataset and the rest as a training dataset; the

model is fitted with the training dataset and evaluated with the testing dataset. The result of each fold is saved and finally, the results from the different folds evaluated are compared. Generally, k -Fold Cross Validation results in a less biased or less optimistic estimate of the ability of the model than other methods. In this study, a value of $k = 10$ will be used; thus, a 10-fold cross-validation will be performed.

After the training and validation, the proposed ABi-LSTM model is able to predict the next packet state (stalling event or not) with an excellent performance. In the learning process, the η value is small (0.0001) and the learning process is quite slow. As depicted in Figure 7, almost all the folds in which the data set is divided achieve notable results in terms of Precision, Recall, and F1. For instance, the model only fails classifying 2936 out of 27,091 packets of one capture, which is an error rate of 10.83%. Table 5 includes the performance results in terms of the confusion matrix, Recall, Precision, and F1 score for this example.

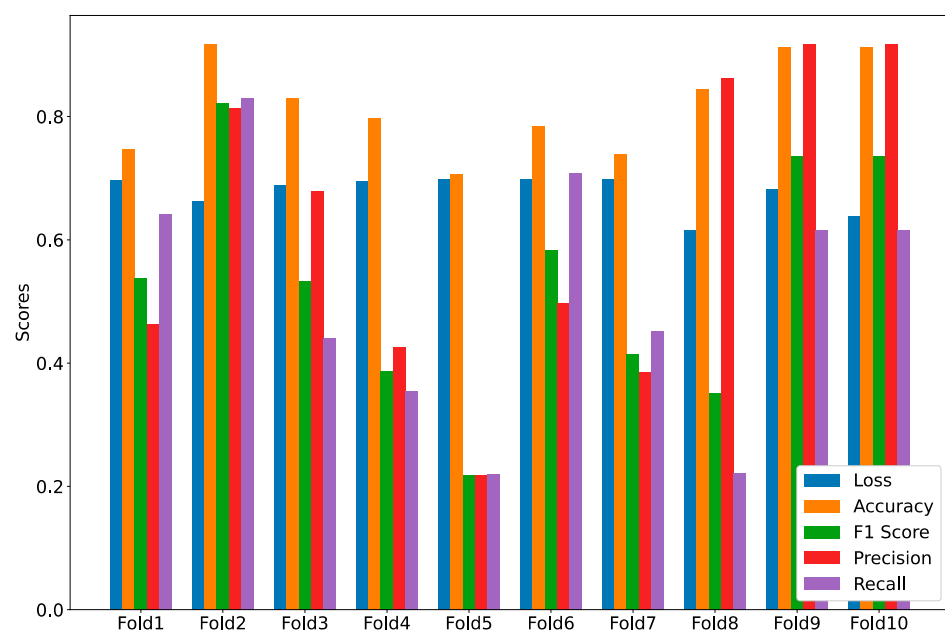


Figure 7. Results of the 10-fold cross validation using the proposed ABi-LSTM model to predict the occurrence of stalling events.

Table 5. Example of the Model Performance for one Data Capture.

Metric	Value
TP	6467
FP	821
FN	2115
TN	17688
Precision	$\frac{TP}{TP+FN} = \frac{17688}{17688+821} = 0.956$
Recall	$\frac{TP}{TP+FP} = \frac{17688}{17688+2115} = 0.893$
F1 Score	$\frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}} = \frac{2 \cdot (0.956 \cdot 0.893)}{0.956 + 0.893} = 0.923$

Finally, we compare from a qualitative perspective our two proposals with those closer to the related literature. The collected information is shown in Table 6. It is important to note that among the methods that do not use information from the buffers, our proposal is the simplest (yet efficient) in terms of number of features and the only one using LSTM as the technique for deep learning.

Table 6. Qualitative comparison of our proposal with other methods from the related literature.

Author	Network	Transp. Layer	Video Streaming	ML Model	Buffer Check	Number of Features
Martinez-Caro & Cano	LTE	TCP	DASH (HTTP)	LSTM & Time Series	No	1
Casey & Muntean [7]	WiFi & Cellular	TCP	DASH (HTTP)	-	Yes	-
Seufert et al. [4]	-	TCP & QUIC	HAS (HTTP)	Weka	No	+300
Tao et al. [3]	Wireless	-	-	DNN	Yes	4
Bampis et al. [11]	-	TCP	HAS (HTTP)	SVR, KNN, NB	No	30
Gutterman et al. [13]	WiFi	TCP	DASH (HTTP)	DNN	Yes	3
Krishnamoorthi et al. [14]	-	TCP	HAS (HTTP)	DT, SVN & Threshold based	Yes	-
Mazhar et al. [15]	3G	QUIC	HAS (HTTP)	C4.5 Adaboost	No	5
Abar et al. [16]	-	TCP	DASH (HTTP)	Threshold-based RF, DNN, KNN & DT	Yes	-
Wassermann et al. [12]	LTE	TCP	HAS (HTTP)	RF, DT, SVM; KNN & NB	Yes	30

5. Conclusions

Competition among video streaming platforms is fierce. Consequently, the price/quality binomial can be a key factor to guarantee users' subscriptions. In terms of quality monitoring, numerous QoE models have been proposed in the related literature, and one common factor among all of them is the number and duration of stalling events. Nevertheless, how to carry out this task in a non- (or minimal) invasive way is still an open issue. In this paper, we have proposed a classification algorithm and a prediction model able to detect and measure the duration of stalling events using only information from the transport and network layers. By doing so, we provide a valuable tool for QoS/QoE service monitoring and the development of new adaptive schemes for video streaming. Stalling events are one of the most influential parameters of users' quality perception. Devising an agnostic method—from the user's device perspective—to detect and foresee stalling events, will allow the application of more powerful countermeasures to recover previous quality levels. In addition, new streaming protocols could benefit from such information, in a similar way to the DASH adaptive mechanism. The fact of being agnostic, that is, to operate without extracting information from the player but only from the network, would be extremely valuable because it could be more easily deployed. Our proposed models have been trained and tested using a dataset containing a DASH-based video stream. Even though our proposal has limitations to be executed in real time, it can serve as a basis to advance towards real-time solutions. The results are very promising, achieving values close to 1 in Recall, Precision, and the F1 score. As future work, we plan to extend these models by testing multiple simultaneous flows, different streaming protocols, and novel neural network mechanisms.

Author Contributions: M.-D.C. conceived and designed the experiments. J.-M.M.-C. contributed with the state of the art, selection of tools, and performed the experiments. Finally, J.-M.M.-C. together with M.-D.C. analyzed the results and wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the AEI/FEDER, EU project grant TEC2016-76465-C2-1-R (AIM) and project SPID202000 × 116746SV0 (AriSe2: FINE).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Statista The Statistics Portal IoT: Number of Connected Devices Worldwide 2012–2025. Available online: <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/> (accessed on 8 March 2021).
2. The Economist Netflix, Disney and the Battle to Control Eyeballs. Who Will Win the Media Wars? Available online: <https://www.economist.com/leaders/2019/11/14/who-will-win-the-media-wars> (accessed on 8 March 2021).
3. Tao, X.; Duan, Y.; Xu, M.; Meng, Z.; Lu, J. Learning QoE of Mobile Video Transmission with Deep Neural Network: A Data-Driven Approach. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1337–1348. [CrossRef]
4. Seufert, M.; Casas, P.; Wehner, N.; Gang, L.; Li, K. Features that Matter: Feature Selection for On-line Stalling Prediction in Encrypted Video Streaming. In Proceedings of the INFOCOM 2019-IEEE Conference on Computer Communications Workshop (INFOCOM WKSHP), Paris, France, 29 April–2 May 2019; pp. 688–695. [CrossRef]

5. Bermudez, H.F.; Martinez-Caro, J.M.; Sanchez-Iborra, R.; Arciniegas, J.L.; Cano, M.D. Live video-streaming evaluation using the ITU-T P.1203 QoE model in LTE networks. *Comput. Netw.* **2019**, *165*, 106967. [CrossRef]
6. Ghadiyaram, D.; Pan, J.; Bovik, A.C. A Subjective and Objective Study of Stalling Events in Mobile Streaming Videos. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *29*, 183–197. [CrossRef]
7. Casey, T.; Muntean, G.M. Reducing stalling events during DASH video playback in heterogeneous multi-network wireless environments. *IEEE Int. Symp. Broadband Multimed. Syst. Broadcast. BMSB* **2017**, 1–6. [CrossRef]
8. Tsolkas, D.; Liotou, E.; Passas, N.; Merakos, L. A survey on parametric QoE estimation for popular services. *J. Netw. Comput. Appl.* **2017**, *77*, 1–17. [CrossRef]
9. Liotou, E.; Tsolkas, D.; Passas, N.; Merakos, L. Quality of experience management in mobile cellular networks: Key issues and design challenges. *IEEE Commun. Mag.* **2015**, *53*, 145–153. [CrossRef]
10. ITU Telecommunication Standardization Sector ITU-T Rec P 1203 Models and Tools for Quality Assessment of Streamed Media. Available online: <https://www.itu.int/rec/T-REC-P.1203/en> (accessed on 8 March 2021).
11. Bampis, C.G.; Bovik, A.C. Feature-based prediction of streaming video QoE: Distortions, stalling and memory. *Signal Process. Image Commun.* **2018**, *68*, 218–228. [CrossRef]
12. Wassermann, S.; Casas, P.; Seufert, M.; Wamser, F. On the Analysis of YouTube QoE in Cellular Networks through in-Smartphone Measurements. In Proceedings of the 12th IFIP Wireless and Mobile Networking Conference: WMNC 2019, Paris, France, 11–16 September 2019; pp. 71–78. [CrossRef]
13. Gutterman, C.; Guo, K.; Arora, S.; Wang, X.; Wu, L.; Katz-Bassett, E.; Zussman, G. Requet: Real-time QoE detection for encrypted YouTube traffic. In Proceedings of the 10th ACM Multimedia Systems Conference (MMSys 2019), Amherst, MA, USA, 18–21 June 2019; Volume 1, pp. 48–59. [CrossRef]
14. Krishnamoorthi, V.; Carlsson, N.; Halepovic, E.; Petajan, E. BUFFEST: Predicting bufer conditions and real-time requirements of HTTP(S) adaptive streaming clients. In Proceedings of the 8th ACM Multimedia Systems Conference (MMSys 2017), New York, NY, USA, 20–23 June 2017; pp. 76–87. [CrossRef]
15. Mazhar, M.H.; Shafiq, Z. Real-time Video Quality of Experience Monitoring for HTTPS and QUIC. In Proceedings of the IEEE INFOCOM 2018, Honolulu, HI, USA, 15–19 April 2018; pp. 1331–1339. [CrossRef]
16. Abar, T.; Ben Letaifa, A.; El Asmi, S. Heterogeneous Multiuser QoE Enhancement Over DASH in SDN Networks. *Wirel. Pers. Commun.* **2020**, *114*, 2975–3001. [CrossRef]
17. ITU-T Parametric Bitstream-Based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services over Reliable Transport. Available online: <https://www.itu.int/rec/T-REC-P.1203/en> (accessed on 8 March 2021).
18. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
19. García, V.; Sánchez, J.S.; Mollineda, R.A. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Syst.* **2012**, *25*, 13–21. [CrossRef]
20. Cao, Y.; Yang, F.; Tang, Q.; Lu, X. An attention enhanced bidirectional LSTM for early forest fire smoke recognition. *IEEE Access* **2019**, *7*, 154732–154742. [CrossRef]
21. Sokolov, A.N.; Alabugin, S.K.; Pyatnitsky, I.A. Traffic modeling by recurrent neural networks for intrusion detection in industrial control systems. In Proceedings of the 2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, The Russian Federation, 25–29 March 2019; pp. 1–5. [CrossRef]
22. Luo, C.; Zhang, N.; Wang, X. Time series prediction based on intuitionistic fuzzy cognitive map. *Soft Comput.* **2020**, *24*, 6835–6850. [CrossRef]
23. Wang, H.; Luo, C.; Wang, X. Synchronization and identification of nonlinear systems by using a novel self-evolving interval type-2 fuzzy LSTM-neural network. *Eng. Appl. Artif. Intell.* **2019**, *81*, 79–93. [CrossRef]
24. Bermudez, H.F.; Sanchez-Iborra, R.; Arciniegas, J.L.; Campo, W.Y.; Cano, M.D. Statistical validation of an LTE emulation tool using live video streaming over reliable transport protocols. *Telecommun. Syst.* **2019**, *71*, 491–504. [CrossRef]
25. ISO/IEC ISO/IEC 23009-1:2019 “Information technology—Dynamic Adaptive Streaming over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats.”. Available online: <https://www.iso.org/standard/79329.html> (accessed on 8 March 2021).
26. Juluri, P.; Tamarapalli, V.; Medhi, D. Measurement of Quality of Experience of Video-on-Demand Services: A Survey. *IEEE Commun. Surv. Tutorials* **2016**, *18*, 401–418. [CrossRef]
27. Wireshark Foundation Wireshark. Available online: <https://www.wireshark.org/> (accessed on 8 March 2021).
28. Chartre, F.; Rivera, A.J.; del Jesus, M.J.; Herrera, F. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing* **2015**, *163*, 3–16. [CrossRef]
29. Batista, G.E.A.P.A.; Prati, R.C.; Monard, M.C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explor. Newsl.* **2004**, *6*, 20. [CrossRef]
30. Sun, Y.; Wong, A.K.C.; Kamel, M.S. Classification of imbalanced data: A review. *Int. J. Pattern Recognit. Artif. Intell.* **2009**, *23*, 687–719. [CrossRef]
31. He, H.; Garcia, E.A. Learning from Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1285. [CrossRef]
32. Krawczyk, B. Learning from imbalanced data: Open challenges and future directions. *Prog. Artif. Intell.* **2016**, *5*, 221–232. [CrossRef]

33. Vluymans, S. *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning Using Fuzzy and Rough Set Methods*; Studies in Computational Intelligence; Springer International Publishing: Cham, Switzerland, 2019; Volume 807, ISBN 978-3-030-04662-0.
34. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
35. Lu, Y.; Cohen, I.; Zhou, X.S.; Tian, Q. Feature selection using principal feature analysis. In Proceedings of the 15th ACM International Conference on Multimedia, Seattle, WA, USA, 24–29 September 2007; pp. 301–304. [[CrossRef](#)]
36. Vavra, J.; Hromada, M. Comparative study of feature selection techniques respecting novelty detection in the industrial control system environment. *Ann. DAAAM Proc. Int. DAAAM Symp.* **2018**, *29*, 1084–1091. [[CrossRef](#)]
37. Puggini, L.; McLoone, S. Feature Selection for Anomaly Detection Using Optical Emission Spectroscopy. *IFAC-PapersOnLine* **2016**, *49*, 132–137. [[CrossRef](#)]
38. Raschka, S.; Mirjalili, V. *Python Machine Learning*, 2nd ed.; Pohlmann, F., Nelson, C., Sangwan, M., Rai, B., Shetty, N., Editing, S., Eds.; Packt Publishing Ltd.: Birmingham, UK, 2017; ISBN 978-1-78712-593-3.
39. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*, 27. [[CrossRef](#)]
40. Dai, S.; Li, L.; Li, Z. Modeling Vehicle Interactions via Modified LSTM Models for Trajectory Prediction. *IEEE Access* **2019**, *7*, 38287–38296. [[CrossRef](#)]
41. Wang, M.; Wang, Z.; Lu, J.; Lin, J.; Wang, Z. E-LSTM: An Efficient Hardware Architecture for Long Short-Term Memory. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2019**, *9*, 280–291. [[CrossRef](#)]