Reducing consumed Data Volume in Bandwidth Measurements via a Machine Learning Approach

Christian Maier*, Peter Dorfinger*, Jia Lei Du*, Sven Gschweitl†, Johannes Lusak†

*Salzburg Research Forschungsgesellschaft mbH, Salzburg, Austria
{christian.maier, peter.dorfinger, jia.du}@salzburgresearch.at

†alladin-IT GmbH, Vienna, Austria
{sg, jl}@alladin.at

***Abstract*—Measurements that determine the available download and upload bandwidth of an end-user Internet connection (so-called speed tests) are typically performed by maximizing the utilization of the connection for a fixed predefined time interval. Especially in broadband connections, such tests consume a huge amount of data volume during their execution. As a result, only a few tests can be performed per month on mobile connections with limited data volumes, since otherwise a significant portion of the volume is used for tests or additional costs are incurred. To reduce the required average data volume of these tests, we present a novel approach with a dynamic test duration based on a machine learning model. We train this model via a supervised learning process, using the recorded data of real speed tests executed in cellular 4G networks. The subsequent evaluation of the resulting method suggests that the amount of saved data volume is significant, while the deviation of the determined bandwidth (compared to a usual test with a fixed duration) is negligible.**

***Index Terms*—Bandwidth Measurements; Machine Learning; Cellular Networks.**

I. Introduction

The concept of bandwidth, defined as the amount of data that a link or path can deliver per unit of time, is central to data communication. It often directly relates to the performance of applications and hence has a strong impact on the end-user Quality of Experience (QoE). This bandwidth is influenced by various factors, such as the used technology, the present infrastructure and environment or the number of concurrent users. In the case of Internet connections, it is usually limited by a bound specified in the contract concluded between the customer and the Internet service provider. In particular mobile Internet connections provide an actual available bandwidth which varies greatly and often does not reach this bound. Especially if the performance of applications suffers noticeable on this gap between the promised and actual bandwidth, users are interested in determining the latter. The methods to achieve this can roughly be divided into two different categories, depending on whether they actually measure or estimate this quantity. Obviously, the effort for estimations is lower, whereas a measurement offers (in general) higher accuracy.

This research was partly funded by the Austrian Research Promotion Agency (FFG).

There are various tools for available bandwidth estimations, such as Pathload, Yaz and Delphi. All of them are (more or less) carried out by producing some small amount of data transfer between the client and the server and by estimating the unused bandwidth from the occurring behavior.

In this paper, we will deal with measurements of the available bandwidth of an end-user Internet connection. These are usually done via a specified speed test, which determines in an accurate way the maximum available download or upload data rate (as well as some other parameters) of the used connection. Examples are the RTR Multithreaded Broadband Test and IPerf. Such tests, which are among the most frequently performed Internet measurements, fully utilize the connection for a fixed time interval. This results in a not inconsiderable amount of consumed data (of up to 250 MB in broadband connections), as well as a noticeable influence on the performance of other applications. The former is in particular a huge problem for customers which have contracts with limited data amount per month. One of the determining factors for the amount of consumed data is the duration of the speed test, which is usually a pre-defined constant. Since this constant is in principle arbitrarily selected, the consumed data volume can be reduced by simply choosing a shorter duration. This will, as we will see, change the result of the test in many cases significantly, in particular if there is a huge dynamic in the available data rate of the connection. The latter behaviour occurs especially in mobile connections.

In practice, there are commonly accepted values for the duration of a speed test which lead to meaningful results. We will not deal with the (difficult) question how to determine such a duration. We just want to mention that the result of a performed speed test should only be used to obtain information about the bandwidth in the respective measurement interval. This is, however, in contrast with the common expectation of an end user on the result of a speed test. The point of view in this paper is the following: If one accepts that a speed test with a fixed predefined duration gives a meaningful value, is it possible to specify a new test (using the same method as the original one) with a fewer duration, which gives a result that is not significantly different than the result from the original test. As argued above, a principally shorter duration will not satisfy the latter requirement. Hence we propose a dynamic

test duration.

To determine this dynamic duration which satisfies the requirement mentioned above, we will use a machine learning model. The use of such techniques in network measurements has increased significantly in recent times. This is on the one hand due to the high dimensionality of network data and on the other hand due to the striking success of machine learning (and especially deep learning) in fields like image and signal processing. There are various different machine learning models. They all have been evaluated for their usefulness in performing different tasks. In our approach, we will use a simple feed forward artificial neural net.

The rest of the paper is organized as follows: Sec. **??** gives an overview on related work, dealing both with bandwidth measurements and estimations, as well as with the application of machine learning techniques to network measurement problems. In Sec. II we describe the necessary details of a single execution of the speed test we are investigating. This also serves to fix some notation for the rest of the exposition. Sec. III analyses the amount of saved data volume and the deviation of of the test result that a general shortening of a stat-of-the art speedtest would cause.

recalls the properties and issues of the usual implementation of this test by choosing a fixed nominal duration.

In Sec. IV we propose to improve this state-of-the-art approach via a dynamic duration for the test, which is determined by a neural net. The necessary training process, as well as some other related topics, are explained in detail. Sec. V evaluates this method by comparing it to the original one and Sec. **??** concludes the paper and gives an outlook on future work.

II. Methodology of Bandwidth Measurements

This section provides a brief description of a test method that determines the available download or upload bandwidth of an end-user Internet connection. In order to keep the exposition simple, we will restrict to the download case. The upload case works (with some minor modifications) in the same way. Besides, we only describe those parts of the test which are relevant for the further discussion.

Such a test has a nominal duration t and a fixed number n of used TCP connections. The nominal duration could either be a predefined constant (which is the case in state-of-the-art implementations), or could be determined dynamically for each execution of the test. We do not specify this for now. Before the actual test begins, a pretest is performed. During this pretest, the client (executing the test) opens n parallel TCP connections to a single test server, which are then used for the transmission of data chunks of growing size. This ensures that the connections are in an active state and also determines some further necessary test parameters. Afterwards, the actual test starts with the server continuously sending data streams, one via each TCP connection, to the client. These data streams consist of chunks of randomly generated data with high entropy. The size of the data chunks varies with a specific distribution, whose parameters are among those mentioned above. For each $k \in \{1, \dots, n\}$, the client records the arrival time $t_k^{(j)}$ of the last bit of the j-th data chunk which was sent on TCP connection k, as well as the total amount $b_k^{(j)}$ of data received on this connection up to this moment. The chunks and the arisen data amount of the pretest are not considered in this records.

After time t, the test server stops sending further chunks on all connections. The client waits until the last sent data chunks are completely transmitted. Let m_k denote the total number of data chunks which were sent on connection k during the actual test execution. The sequence

$$(t_k^{(1)}, b_k^{(1)}), (t_k^{(2)}, b_k^{(2)}), \dots, (t_k^{(m_k)}, b_k^{(m_k)}) \tag{1}$$

describes the course of the test on TCP connection k. To determine the test result from the n recorded courses, let $t^* = min\{t_1^{(m_1)}, \dots, t_n^{(m_n)}\}$ denote the time when the first transmission was finished. Then an approximation b_k for the amount of data received over TCP connection k from the beginning of the test until time t^* is given as

$$b_k = b_k^{(m_k-1)} + \frac{b_k^{(m_k)} - b_k^{(m_k-1)}}{t_k^{(m_k)} - t_k^{(m_k-1)}}(t^* - t_k^{(m_k-1)}) \tag{2}$$

and $R = (b_1 + \dots + b_n)/t^*$ is defined to be the bandwidth determined by the measurement method. The total data amount consumed during the test execution is given as $V = \sum_{k=1}^{n} b_k^{(m_k)}$. This value depends on the nominal test duration and on the characteristics of the connection to be measured.

The test records can also be used to calculate the result R_τ and the consumed data amount V_τ that a test execution with a shorter nominal duration τ would have yielded. This is done by removing all tuples $(t_k^{(j)}, b_k^{(j)})$ in the sequence (1) that contain information about data chunks sent after time τ and a subsequent repetition of the above calculation. The absolute percentage deviation of R_τ from R is then given as $d_\tau = 100 \cdot |1 - R_\tau/R|$. By replacing R with V and R_τ with V_τ in the right-hand-side of this formula, one obtains the percentage s_τ of saved date volume that this reduction of the nominal duration from t to τ would cause. These two quantities (d_τ and s_τ) will serve as metrics for evaluating the behaviour of test shortenings.

Fig. 1 show the courses for all connections of an execution of the test performed in reality in a 4G cellular network with a nominal duration of $t = 7s$ and $n = 3$ parallel TCP connections. This test consumed a total amount of about 120 MB. A test execution with a shorter nominal duration of e.g. $\tau = 4s$ would have consumed $s_\tau = 43\%$ less data volume. The deviation that this shortening would cause is $d_\tau = 20\%$. This significant deviation may not be acceptable for providers implementing such tests or for customers performing them. In general, the percentage deviation d_τ will be small if the courses on all threads are linear, and otherwise huge.

III. Fixed Test Duration

It is state-of-the-art to implement a speed test using the method specified in the last section by choosing a constant

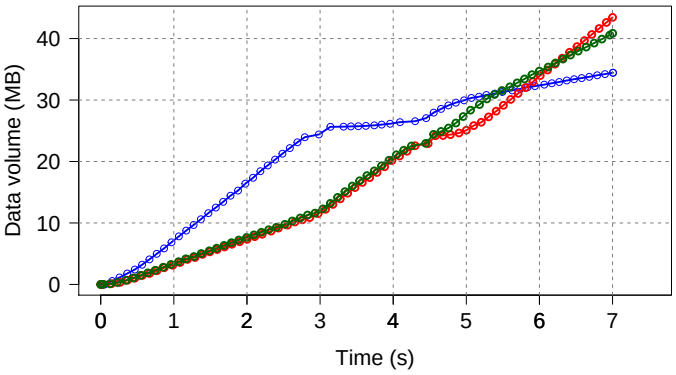


Fig. 1. The courses on all TCP connections of a (download) speedtest performed in reality in a 4G cellular network.

predefined nominal test duration $t = T$. To reduce the amount of consumed data volume of this test, the simplest approach is to generally shorten the test duration from T to another fixed constant τ (with $\tau < T$). As the example at the end of the last section shows, this shortening must not be too large, since otherwise this would often result in a significant deviation in the resulting bandwidth. This arises the question of there if any value $\tau < T$ such that on the one hand the average consumed date volume of the test is reduced significantly, and on the other hand the test result stays more or less the same.

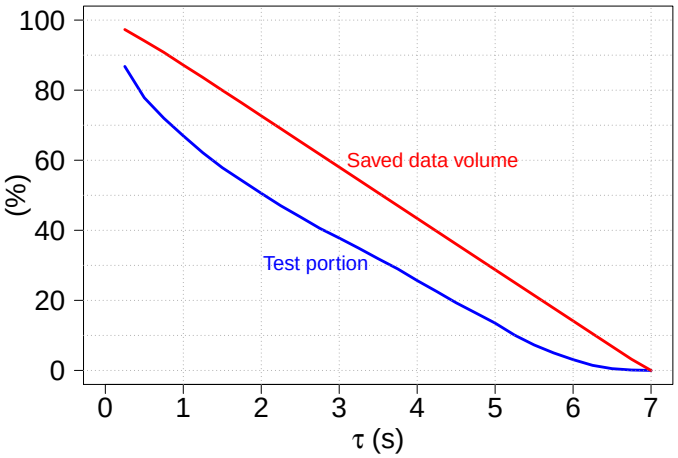
To get an answer on this question, one can consider a database which contains as entries the recoreded courses on all TCP connections of real executions of this speedtest. For any $\tau < T$, one can than calculate for each test in this database the values d_τ and s_τ and from this the average of them over all tests in the database. When doing this it is necessary to differentiate between downlaod and upload tests since the behavior of them may be different.

All the tests were performed on mobile Android or iOS devices in 4G cellular networks. The test method is the one described in Sec. II, with a slight (non-relevant) difference on iOS devices. For an evaluation, one chooses a fixed p such that one considers the deviation as significant if $d(\tau) > p$. For each τ we than get the test portion $f(\tau)$ of those tests where the deviation is significant and a function $g(\tau)$ of average percentage saved data volume. Both functions tend to zero if τ tends to T

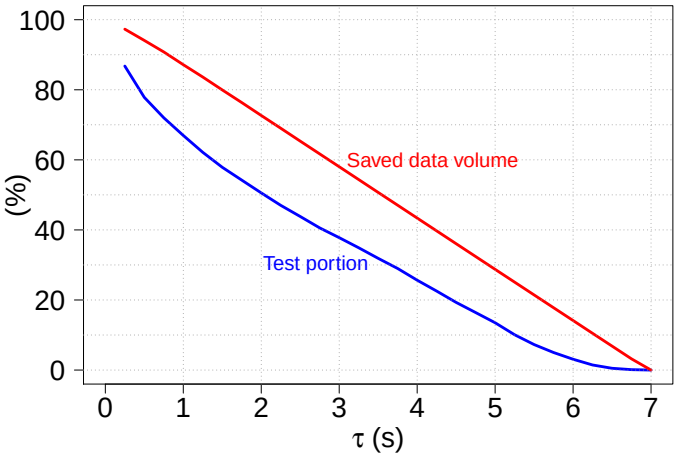
We did this with a database containing 37935 download tests and 37333 upload tests with a value $p = 10\%$ (this seems to be a plausible value). All the tests The resulting function are shown in Fig. ?. As one can see, there is no τ which satisfies the required properties. E. g. at $\tau = 4$, the amount of saved data volume is 43%, but also a significant deviation in 25% of the tests (i.e. in every 4th tests).

IV. Dynamic test duration

Lets consider a speed test (measuring either download or upload throughput) as presented in detail in Sec. II with a fixed nominal duration T. The data rate determined by this test (on a single execution) is denoted with R. Our goal is to specify a



(a) Download



(b) Upload

Fig. 2. The amount of saved data volume and the portion of tests where the deviation between the result of a test with duration τ from the result of the test with duration $T = 7s$ would be significant.

new test with a dynamic nominal duration t (satisfying $t \leq T$) which, on the one hand, gives a result R' that only slightly deviates from the data rate R and, on the other hand, reduces the average consumed data amount extensively. To keep the exposition simple, the set of possible values for t is chosen to consist, besides T, just of one more element, denoted τ (which evidently satisfies $\tau < T$). This means that the new test either stops when the last data chunk, which was sent before time τ, arrives (we will henceforth call this a short test) or the test proceeds as the original one. The decision, which value t takes (that is, which of the just mentioned cases occurs), is made on the basis of a calculation performed by a trained artificial neural net. To be more specific, this neural net (which is a simple multi-layer perceptron) has two output values, which are real numbers, denoted y_1 and y_2. It is trained in a way such that a high value of y_1 is an indicator that a test execution with nominal duration τ would yield a result that does not deviate from R considerably. On the other hand, a high value of y_2 indicates the opposite, i.e. that there is a huge difference between R and the data rate that a short test would determine.

Hence the natural decision rule (determining t) using these two output values is the following:

If $y_1 \geq y_2 + \lambda$, then $t = \tau$ (with $\lambda \in \mathbb{R}$).
Otherwise, $t = T$.

Here we introduced a parameter λ, which controls how carefully the rule decides for a short test: The larger λ is, the higher the indicator y_1 must be (compared to y_2) in order to make such a decision.

The calculation of the neural net and the subsequent decision for $t = \tau$ or $t = T$ is done at the aforementioned time of arrival of the last chunk sent before time τ. As input, the neural net obtains the course of the data rate from the beginning of the test up to this moment. Since the calculation has to be done on mobile end devices, one needs to keep in mind the required computing effort, which depends on the actual structure of the used neural net. To train this neural net, the obvious approach is via a supervised learning process using the recorded data of real executions of the original speed test (with fixed nominal duration T) on mobile devices in cellular 4G networks. These records are also used to determine the parameter λ, which completes the specification of the proposed method. Next, we will describe all the things that have been mentioned now in more detail.

A. Neural Net Inputs

As already mentioned, the neural net gets as input the recorded data rate in the time interval $[0, \tau]$. With the specification of the test method in Sec. II, this course is given for each thread $k \in \{1, \ldots, n\}$ as a sequence of fractions

$$\frac{b_k^{(1)}}{t_k^{(1)}}, \frac{b_k^{(2)}}{t_k^{(2)}}, \ldots, \frac{b_k^{(m_k)}}{t_k^{(m_k)}}$$

with $t_k^{(r_m-1)} < \tau$ and $t_k^{(m_k)} \geq \tau$. Usually, the length of this sequence varies from one test to another (and also from one TCP connection to another). Since the number of actual input values has to be constant, it is necessary to preprocess this course in order to make it accessible to the neural net. To achieve this goal, one chooses a resolution $\delta > 0$ in a way such that τ is an integer multiple of δ. The inputs delivered to the neural net are then the approximations $x_k^{(i)}$ of the data rate on connection k in the time intervals $[(i-1)\delta, i\delta]$ (with $i = 1, \ldots, \tau/\delta$), which are calculated from the fractions $b_k^{(j)}/t_k^{(j)}$ in the obvious way (i.e. by a linear approximation, see Fig. 3). This results in $n\tau/\delta$ input values.

B. Structure of the Neural Net

The considered neural net is a feed-forward neural net with one input layer, several hidden layers and one output layer (i.e. a multi-layer perceptron). The discussion of the preceding subsection leads to $n\tau/\delta$ input nodes. As activation functions we selected ELU (exponential linear unit) functions for all neurons in the hidden layers, which are given as

$$\varphi(x) = \begin{cases} e^x - 1, & x \leq 0 \\ x, & x > 0 \end{cases}$$

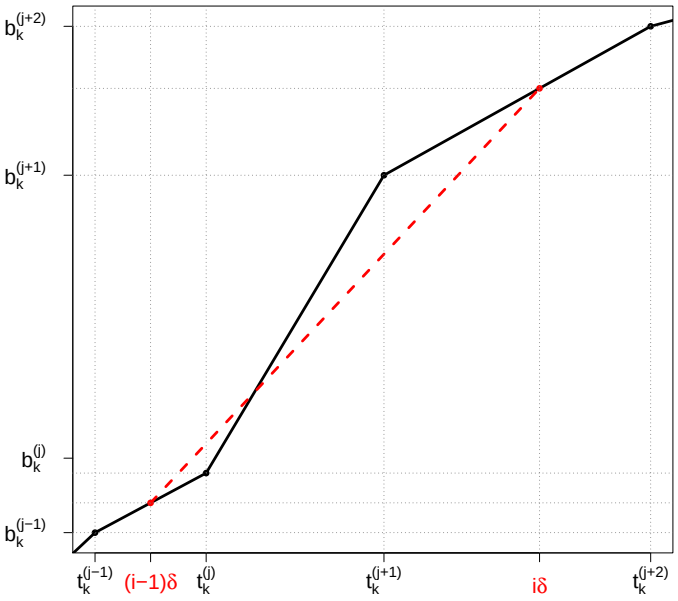


Fig. 3. The input value $x_k^{(i)}$ is the slope of the dashed line.

For the two nodes at the output layer (which provide the values y_1 and y_2) we used linear activation functions, which is a common choice for classification tasks.

C. Training Process

To train a neural net such that its output values are indicators for or against a short test duration, one can use a huge database containing the recorded courses of real speed test executions with fixed nominal duration T. For a supervised learning process, it is necessary to label this data. This is done in the following way: Using the procedure described in II and III one calculates for each record in the database the test result R and the result R_τ, which a test with nominal duration τ would yield. The absolute percentage deviation $d = 100 \cdot |R_\tau/R - 1|$ from R_τ compared to R splits the data into two classes: Class A if d is smaller than p%, and class B otherwise. Here p is a selected, fixed constant. After the data are labeled in this way, the neural net is trained to determine from the inputs $x_k^{(i)}$ the class which it belongs to. That is, the neural net should learn to determine from the course of the data rate in the first τ seconds of the test execution, if the deviation from R_τ to R is above or below p%. Note that R_τ can (approximately) be calculated from the values $x_k^{(i)}$. The accuracy which the neural net reaches in its task will show how much information the values $x_k^{(i)}$ contain in order to determine R.

D. Determination of the Control Parameter

The test data can also be used to determine the control parameter λ for the proposed speed test. This can be done in the following way: One selects a small percentage ρ such that a significant deviation of the result in less than ρ percent of the tests is acceptable (for customer and the provider of the test).

E. Computing Effort

The computing effort of the neural net to decide whether a test belongs to class A or class B consists essentially of three matrix multiplications, some additions and the multiple application of the activation functions. Especially with optimized frameworks (like TensorFlow Lite), this can be done on mobile end-devices in real time. On the other hand, the training process need much more computational performance. However, this does not matter, since the training needs only be done once and can be done on a powerful machine.

F. Summary of the Method

To recapitulate, the proposed test with dynamic duration starts as the usual (fixed duration) test. After the last data chunk which was sent before time τ arrived completely at the client, the courses of the data rate on each TCP connection from the beginning of the test up to this moment is sampled with resolution δ. This sampling process yields the inputs for the neural net, which was trained before to decide on the basis of these values, whether the test with nominal duration τ gives a result which is not significantly different from the result, that a test with nominal duration T would produce. A possibly low confidence in the result of the neural network can be counteracted by selecting a high parameter λ. After the decision for or against a short test is made, the test either stops or the server continuous with sending data chunks until time T.

V. Evaluation

To evaluate the method presented in Sec. IV we consider two databases which contain the recorded courses of real executions of a speed test implementation with a fixed nominal duration of $T = 7s$ and a nominal number of $n = 3$ parallel TCP connections. The separation of the data into these two databases, as well as the number of test records in each of them, has historical reasons without any value. All the tests were performed on mobile Android or iOS devices in 4G cellular networks. The test method is the one described in Sec. II, with a slight (non-relevant) difference on iOS devices. The data further splits into download and upload bandwidth tests. Since one suspects that there is a difference in the behaviour of the tests in these two categories, they must be handled separately. The number of tests in each database per category is show in Tab. I.

TABLE I
Number of test records used for the evaluation

	Download	Upload
Database I	37935	37861
Database II	401601	400790

The general basic principle of our evaluation is the following: From the course of a single test execution, it is not only possible to calculate the result R from the original test, but also the result R_τ that a test with a shorter duration τ would have yielded to. This is done just by ignoring the information coming from all data chunks which were sent after time τ and determining R_τ by the same procedure which was specified in Sec. II to calculate R. Now the percentage deviation of R_τ from R is given as $d_\tau = 100 \cdot (R_\tau/R - 1)$. We will say that this deviation is significant, if $|d_\tau| > 10\%$. The selected boundary of 10% is in principle arbitrary chosen, but seems as a quite natural choice. In addition, we can also calculate the amount V of consumed data of a single original test, as well as the amount V_τ of data that a test with duration τ would have consumed.

In order to be able to interpret the performance of the test with dynamic duration, it is useful to investigate the percentage deviation in the result and the amount of saved data volume that a general shortening of the duration T would cause. This is done by calculating for each duration τ (with $0 < \tau < T$) the portion of tests in database I with significant deviation between R_τ and R (i.e. with $|d_\tau| > 10\%$) and the total portion of saved data volume. The results are shown in Fig. 2. For example, a general shortening of the nominal download test duration from 7 to 4 seconds would result in a 43.95% reduction in the amount of consumed data volume, but also in a significant deviation of the test result in 25.93% of the tests. For upload tests, the corresponding values are 42.67% (saved data volume) and 22.00% (test portion).

For the test with dynamic duration, we selected a duration of $\tau = 4s$ for a short test and a resolution of $\delta = 0.25s$ for the sampling of the courses. This leads to $n\tau/\delta = 48$ input values for the neural net. The number of hidden layers was set to 2, with 32 neurons in the first and 16 neurons in the second hidden layer. The training process of this machine learning model was done with the test records in database II. Due to the reason already mentioned above, a neural net has to be trained separately for the download and the upload case. As it is usual practice, the data was further divided into actual training data and test data (to check if no overfitting behavior occurs) in a $9:1$ ratio. For the labeling of the data as type A and type B we selected a bound of 5%. The neural nets were implemented in Python using the TensorFlow Framework. Weights and Biases were initialized randomly. Optimization was done with the Adam Optimization Algorithm. The number of training epochs was set to 1000 with a batch size of 50. The neural nets reached a maximal accuracy of about 76% in the classification of download tests and about 78% in the upload case. The accuracy of just 75% is not surprising, since the 4sec data definitely does not fully determine the test length. After that, we selected the necessary parameter λ in a way such that the percentage of tests with significant deviation between the result of the test with dynamic duration and the result of the test with fixed duration is exactly 1%. This can be done, since this portion tends to 0 with ascending λ. The resulting values are $\lambda = 0.9$ for the download test process and $\lambda = 1.5$ for the upload process.

The subsequent evaluation was again done with the tests in database I, using the same principle as above: We calculated the test portion with significant deviation in the result of the

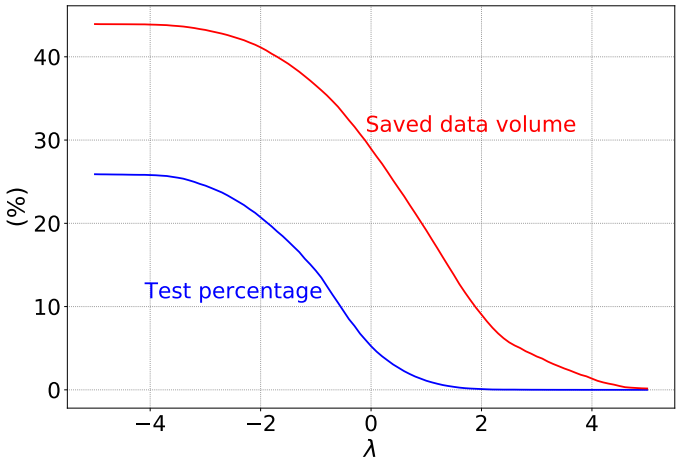


Fig. 4. The amount of saved data volume and the percentage of those tests where the deviation between the result from the test with dynamic duration and the test with fixed duration is more than 10% (download).

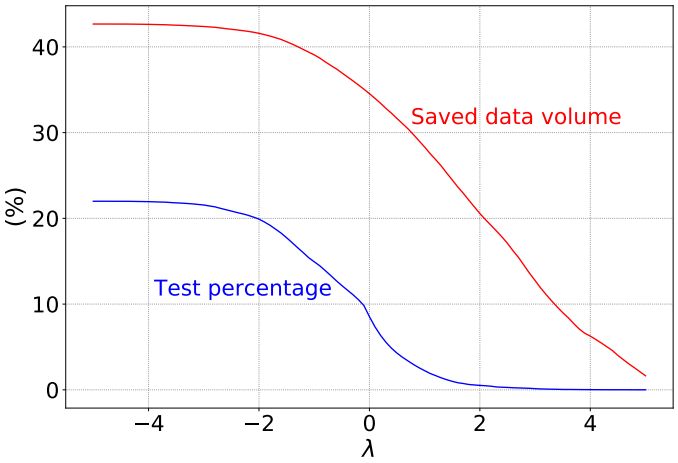


Fig. 5. The amount of saved data volume and the percentage of those tests where the deviation between the result from the test with dynamic duration and the test with fixed duration is more than 10% (upload).

test wit dynamic duration and the test with fixed duration, as well as the amount of reduced data volume. Tab. III shows the resulting values.

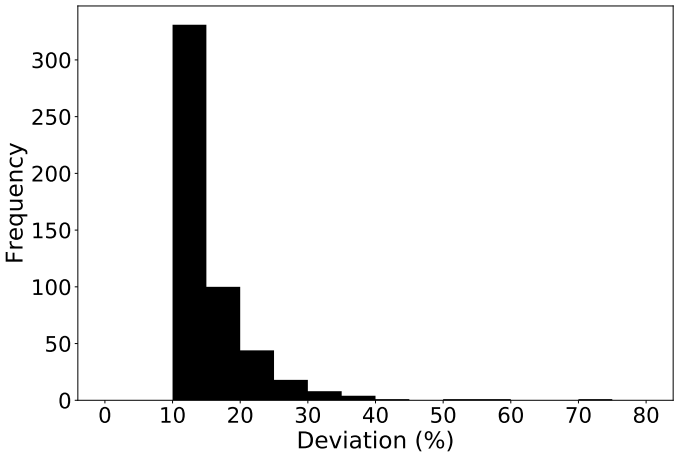
TABLE II
EVALUATION OF THE PROPOSED METHOD

Quantity	Download	Upload
Number of evaluation data	37935	37861
Number of short measurements	11851	13162
Tests with significant deviation Portion	510 1.34%	375 0.99%
Consumed data volume of the original method	$1453GB$	$586GB$
Consumed data volume of the proposed method	$1160GB$	$443GB$
Portion of reduced data volume	20.18%	24.49%

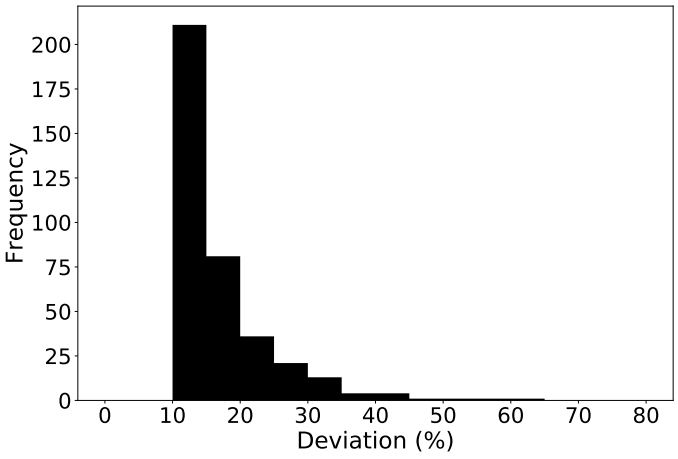
Fig. 6 shows a histogram of the percentage deviations of those tests where this value is above 10%. As one can observe, most of them are just short above 10%. From the values in Tab. III it can be seen that a percentage of 31.24% of short tests results in a saving of 45.92% in relation to the total possible saving (i.e. if all tests would last only 4 seconds). This shows that a short measurement is possible, especially for tests with a high data volume. A more detailed investigation of this fact provided the following results: The average required data volume of all tests downloadis 38.31 MB, whereas the average required data volume of all tests where a short measurement is possible is 56.76. MB.

TABLE III
AVERAGE CONSUMED DATA VOLUME OF THE ORIGINAL SPEED TEST

	Download	Upload
All Tests	38.31 MB	15.49 MB
Tests which can be shortened	56.76 MB	26.21 MB



(a) Download



(b) Upload

Fig. 6. Histogram of the deviations which are beyond 10%.