On the Role of Font Formats in Building Efficient Web Applications

Benedikt Dornauer[1,2][0000−0002−7713−4686], Wolfgang Vigl[1][0009−0007−4938−2782], and Michael Felderer[1,2,3][0000−0003−3818−4442]

[1] University of Innsbruck, 6020 Innsbruck, Austria
{benedikt.dornauer,wolfgang.vigl}@uibk.ac.at
[2] University of Cologne, 50923 Cologne, Germany
[3] German Aerospace Center (DLR), Institute for Software Technology, 51147 Cologne, Germany
michael.felderer@dlr.de

Abstract. The success of a web application is closely linked to its performance, which positively impacts user satisfaction and contributes to energy-saving efforts. Among the various optimization techniques, one specific subject focuses on improving the utilization of web fonts. This study investigates the impact of different font formats on client-side resource consumption, such as CPU, memory, load time, and energy. In a controlled experiment, we evaluate performance metrics using the four font formats: OTF, TTF, WOFF, and WOFF2. The results of the study show that there are significant differences between all pair-wise format comparisons regarding all performance metrics. Overall, WOFF2 performs best, except in terms of memory allocation. Through the study and examination of literature, this research contributes (1) an overview of methodologies to enhance web performance through font utilization, (2) a specific exploration of the four prevalent font formats in an experimental setup, and (3) practical recommendations for scientific professionals and practitioners.

Keywords: front-end development · web font formats · web typography · performance evaluation · energy-efficiency

1 Introduction

To ensure the success of websites and achieve optimal user satisfaction, it is crucial to consider usability and various other design criteria [29, 32]. According to Took [38], users' interaction with websites with a higher-than-normal user experience was significantly associated with improved web performance. The importance of performance was also stated by Google's March 2016 data showing that more than half of mobile site visits are abandoned if the site takes longer than three seconds to load [1]. Besides that, improving performance also contributes to energy-saving efforts, reducing the time and resources required for rendering and displaying content [34]. Consequently, comprehensive research is in progress

to enhance the efficiency of web applications with different approaches, including specific considerations related to the utilization of fonts.

In an article from 2022, Google experts Hempenius and Pollard [17] explore the potential performance enhancements of different web page font settings. They identify potential bottlenecks for web performance and provide insights into mitigation opportunities. One of their mentioned bottlenecks was the improper choice of suitable **font formats**.

Usually, fonts are utilized in computer systems and other text presentation systems to represent glyphs visually. Most computers have various fonts available preinstalled for creating documents and graphics. This availability and flexibility of fonts are the culmination of over three decades of gradual advancements in computer font science [43]. Nowadays, fonts also play a pivotal role in impactful web designs, as font features convey feelings and reactions that text alone cannot achieve. For this reason, fonts are often chosen that match the corporate design [37].

Unfortunately, these specific font styles are often not preinstalled on the devices by default [44]. In order to solve this requirement (using a non-standard font), web developers use a `@font`-face declaration in Cascading Style Sheet (CSS)-file to declare the new fonts. The declaration also includes a URL to the online font-file resource (e.g., Google Fonts), a file holding information about the specific custom font [30].

Among the various font file formats, the predominant options encompass the system font formats, TrueType Font (TTF) and OpenType Font (OTF), as well as the web font formats, Web Open Font Format (WOFF) and its second generation WOFF2. Most of web browsers widely support TTF, OTF, and WOFF, whereas WOFF2 is comparatively supported only by newer browser versions. TTF and OTF hold particular significance due to their extensive availability, serving as standard font formats developed by Adobe, Microsoft, and Apple. While TTF and OTF are font formats designed for system fonts, WOFF and the newer version WOFF2 are web fonts optimized for loading from a web server. WOFF is a container format that embeds TrueType or OpenType fonts and compresses them. The second version of WOFF can show a significantly reduced file size, according to Buhler et al. [5]. Following the transmission of web fonts to the client, the browser undertakes data decompression to facilitate font loading and display. Although the diminished file size contributes to decreased transfer time, this advantage is anticipated to be accompanied by heightened CPU and memory utilization [21, 31].

This study investigates how the choice of font format in web applications affects the client´s device performance. Therefore, we conducted a benchmark experiment to investigate the effects during the loading of web content of **different font formats (independent variable)** on the **clients' performance-related metrics (dependent variable)**. Therefore, the research question is:

[RQ] How do different web font formats compare in terms of their impact on performance improvement in web applications?

In order to evaluate the performance-related metrics, the following null hypothesis are defined:

H_1 The font format does not influence the required **Document Loading Time**.
H_2 The font format does not affect the **Processor Cycles**.
H_3 The font format does not impact the **Allocated Amount of Memory**.
H_4 The font format overall does not influences the **Energy Consumption**.

After providing the necessary context and motivation for this experiment, the subsequent sections of this paper are organized as follows. Section 2 provides background information on font optimization techniques and related work, including grey literature. Section 3 describes the conducted methodology. Subsequently, in Section 4, the implications of these findings are then discussed in 5 and followed by Section 6, which addresses study limitations and potential Threats to Validity. Finally, Section 7 concludes the overall study.

2 Related Work

Prior studies on web font optimization have followed two distinct methods. The first perspective focuses on identifying improved visual representations that enhance task performance, promote better text comprehension, and ultimately increase user satisfaction (*2.1 Aesthetic Optimization*). The second viewpoint revolves around optimizing computational performance, such as by reducing the load time, which is the perspective that is considered mainly in this paper (*2.2 Performance Optimization*).

2.1 Aesthetic Optimization

Ling et al. [24] have presented the results of two experiments in which the influence of font and line length on several task performance and subjective measures was investigated. The authors showed that the effect of line length was significant on performance, but the effect of font type had only an insignificant small impact.

Similarly, Bhatia et al. [4] have conducted a study to investigate the effects of font size, italics, and color count on three web usability dimensions: effectiveness, efficiency, and satisfaction. While the effectiveness and efficiency of the participants were measured via tasks, satisfaction was determined using a survey instrument. The study showed that font size and number of colors had no significant effect on any variable. However, using italics had a statistically significant effect on performance but not on efficiency and satisfaction.

In 2016, Rello et al. [33] examined the font size and line spacing in more detail regarding objective and subjective legibility and showed a continuous improvement in both up to a size of 18 points. From 22pt, there was again a decrease in subjective legibility. The effects of line spacing on objective legibility were insignificant, but participants indicated that their subjective legibility was impaired at extreme values (0.8 and 1.8). The authors summarized that increasing the font size is an efficient way to improve legibility.

2.2 Performance Optimization

Improving a web application's energy efficiency, related to many other performance metrics, requires a deep understanding of various optimization techniques. Therefore, for instance, Wagner [41] collated several performance improvement techniques that were also addressed in research. Such as optimizing CSS as well as JS content (e.g., [6]), tackling the problem of media-related optimization (e.g., [42]), considered different transmission protocols (e.g., [15]), covering design-effective aspects (e.g., [19, 22]) and also mentioned the optimization of web fonts considering mainly the application layer of the Open Systems Interconnection model (OSI). Riet et al. [40] further advanced these techniques by incorporating some of them into a replicable performance engineering plan consisting of 13 interventions for the desktop and mobile web. Appropriating those to a sample case study showed significant performance improvement opportunities. One of those was "Intervention 10: Preload Fonts", also known as lazy-loading. Besides the papers mentioned above, we also reviewed some gray literature that described other font-related techniques, as follows:

Font Subsetting: is a technique for reducing the size of a font file. Here, only the characters needed in a font file are selected, and the rest is discarded. One often-used example is subsetting a font by language, e.g., to provide a font with only Latin characters for English-language pages. Using this technique, the loading time of the fonts can be improved by more than 200% [41]. The widely used Google Web Fonts API is able to automatically create a subset for many font families by providing an additional attribute. There is also the possibility to create a subset specifically for custom purposes by modifying the file [2]. Last but not least, the CSS *unicode-range* property of a `@font-face` Definition specifies the characters for which, if any, the font is to be loaded [41].

Font Hosting: There are two ways to load fonts: Self-Hosting or Third-Party Hosting. While self-hosting stores the files on your own web server, third-party hosting uses a font service such as Google Fonts [26]. While using Third-Party-Hosting is generally considered easier, it requires additional communication with an external resource, resulting in a decreased loading speed and a dependency on the service provider [23].

Font Loading: Another option is using `<link rel="preload">` so that the font is not loaded when it is encountered in the external stylesheet but already when this tag appears. Another variant is to use the Font Loading API in the JavaScript code. This way, the process can be followed precisely, and user-defined steps can be initiated [14].

Font Rendering: The *font-display* CSS attribute determines what happens until the external font file is loaded: Should the browser wait until it is loaded or

render the text in a fallback font? Besides these two options, there are others to choose from. The default behavior varies from browser to browser. The attribute affects the *Largest Contentful Paint*, the *First Contentful Paint*, and the stability of the layout [17].

General Optimizations: Furthermore, the use of general optimization strategies is often suggested, such as enabling client-side caching [14] or enabling server-side compression using algorithms such as GZIP or Brotli. The latter should only be used for TTF and OTF formats, as WOFF and WOFF2 already use built-in compression [20].

Many studies have been conducted, focusing on several aspects of performance improvement or improving the visual perception of fonts. However, we have not identified any scientific literature yet examining the performance of font formats in terms of several efficiency criteria. Also, the grey literature in this field has no benchmarking tests available. The present study attempts to fill this gap.

3 Experimental Methodology

In order to assess the influence of various font formats on the performance of web applications and address the research questions as well as the hypothesis, we have outlined the following proposed experimental framework, simplified in Fig. 1.

3.1 Environmental Setup

The web client (Windows 11; AMD Ryzen 5 5500, 32GB) and web server (Ubuntu 22; i7-6700HQ, 16GB) are hosted on two different machines inside the same Wireless Local Area Network (WLAN). The web server uses NGINX to handle the web requests and deploy the web application containing the different web formats.

The web application consists of a single web page containing several headings together with textual paragraphs and a footer. In total, three different fonts are used, as this is a common practice for designing interfaces [37]. The following combination was selected:

- *Raleway Extrabold v3.000* for the headings,
- *SourceSans Regular v3.052* for the paragraphs, and
- *Montserrat Semibold v3.100* for the footer.

Each of the fonts is provided in each of the considered font formats: *TrueType Font (TTF)*, *OpenType Font (OTF)*,*Web Open Font Format Version 1.0 (WOFF)*, and *Web Open Font Format Version 2.0 (WOFF2)*. While all of the required formats of Montserrat are downloaded directly from the official Github

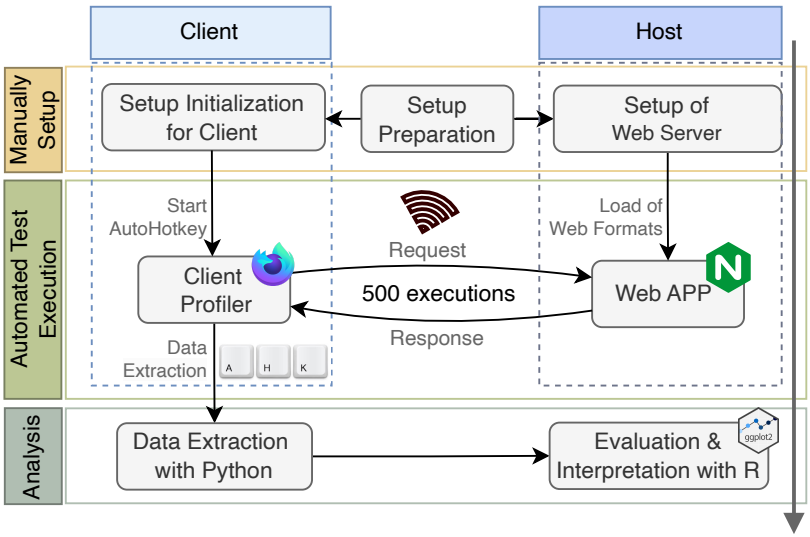


Fig. 1: Overview of experimental setup.

repository [39], the repositories of Raleway [25] and SourceSans [18] did only provide TTF and OTF versions. For obtaining the WOFF and WOFF2 formats, which contain the same information as their counterparts, two prevalent NPM conversion tools *ttf2woff version 3.0.0* [35] and *ttf2woff2 version 5.0.0* [12] are used to convert from TTF to WOFF as well to WOFF2. Both packages claim over $150K$ weekly installations.

Performance measurements are subject to fluctuations on the client device due to other processes running on the same client machine. These processes have different resource consumption at different times. To mitigate this specific threat, some common practices [9] were considered. First, the client device is restarted before each bunch of repeated trials to minimize the influence of processes running and ensure the trials are independent. After the reboot, all heavy background processes are shut down, including collaboration tools, synchronization software, antivirus software that might perform arbitrary security scans, and background browser processes. These measures are intended to put the computer into a "low idle energy fluctuations" state before profiling is started.

3.2 Automated Test Execution

A total of 500 individual trials were conducted for each font format, wherein the resource consumption was meticulously profiled. To measure the performance of the web application, a commonly used tool was applied, the *Firefox Profiler* [11]. Specifically, the developmental version *Firefox Nightly* (version 115.0a1) is used, as memory usage analysis is only available in the respective Firefox Profiler. In this way, we were able to analyze and measure various performance metrics of the

entire browser process or specific threads. It provides insights into CPU, memory, as well as energy consumption. This study did not consider other browsers, as they lack support for the aforementioned performance metrics.

The selected profiler takes samples at a desired interval and writes the result to a buffer, overwriting old values when complete. The sampling rate in this experiment was set to 0.5 milliseconds, with a buffer size of 2 GiB. In addition, the following manual settings are applied to all experimental runs in order the retrieve the specific metrics: any additional threads have been disabled, Browser Cache deactivated, only CPU Utilization has been enabled in the "Features" section, and the experimental features "Process CPU Utilization" and "energy Use" have been switched on.

The repeated trials are conducted automatically using AutoHotkey [13]. The respective script launches Firefox Nightly, starts the Firefox Profiler, navigates to the web application, stops the Profiler, saves the result as JSON, and shuts down Firefox before the next run starts.

Some actions have been performed to minimize skewed results in the runs. Causes of such distortions include, for example, the Profiler recording the resource consumption of other processes and tasks or the runs influencing each other. For this reason, the following delays are built into the AutoHotkey script: an 8-second delay after Firefox Nightly is started and an 8-second delay after it is closed.

Each batch of runs, where each font is provided with the same format, is automatically profiled 500 times, and the results are used for evaluation.

3.3 Data Evaluation Process

The data provided by one run of the Firefox Profiler includes performance metrics, events, and other actions that happened during the recorded time frame. The Profiling was started before the request and stopped after it finished. While each metric covers the entire recorded period, only a fraction of that period is necessary for examining performance. To focus on the font acquisition process, including decoding, conversion, and display, the data series of each performance metric is trimmed accordingly. The starting point is determined by extracting the time in milliseconds when the "DOMContentLoaded" event occurs. This event indicates that the HTML file has been downloaded, parsed, and external resources are being fetched [27]. Similarly, the end time is determined by the millisecond timestamp when the *Load Event* is triggered. At this point, all resources, including fonts, have been successfully loaded and rendered [28].

The profile of one run provides many measurement values, specifically yielding information about the performance. Some of these metrics are expressed as related to a particular thread. This allows a targeted analysis of that specific thread in the browser process, which particularly processes the request to the test web application, and further excludes any values of other threads. However, other measurements, such as the energy consumption are provided per core and in total. When evaluating the performance, the metrics given in Table 1 are extracted using a Python script and considered in the evaluation.

Table 1: Performance-related metrics overview.

Metric	Description	Unit
Load Time	The time in Milliseconds between the events *DOMContentLoaded* and *Load*.	Milliseconds **[ms]**
CPU Cycles	The Profiler's JSON output provides a data series for each thread, indicating the number of processor cycles required between each sampling point. For each run, the sum of this data series between the two events considered is extracted.	CPU Cycles **[Count]**
Average memory allocation changes	This metric provides the number of relative changes in the allocated memory. The size of the allocated memory at a given sampling point in time can be calculated by the cumulative sum of the changes. However, since an investigation of the performance is primarily concerned with the development of memory usage, absolute values are not required and the 10% trimmed mean of the series of relative values is sufficient for analysis.	MegaByte **[MB]**
Energy Consumption	The energy consumption provided by an implementation of RAPL (Running Average Power Limit) is used [16] and is specified in picowatt-hours by the Firefox Profiler. The sum of all data series values within the two events is considered.	Milli-Watt-hours **[mWh]**

Although a sampling rate of 0.5 milliseconds was set, the distances between the individual samples are not always uniform. For this reason, the values are first converted to *per milliseconds* to allow for comparability. Subsequently, for each data series provided, only the values that lie within the desired observation period, i.e. between the two events, are considered.

4 Results

The analysis presented in the following intends to cover all aspects to provide a holistic view. In all resource consumption evaluations, the required time between the events *DOMContentLoaded* and *Load* is considered.

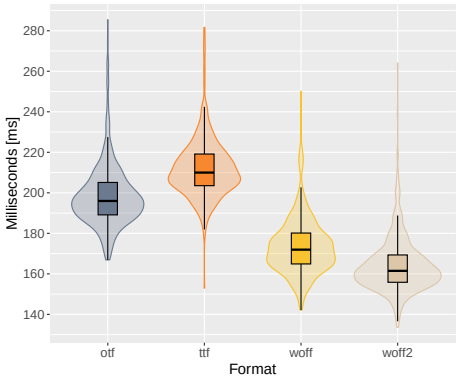
4.1 Load Times

First, a look at the differences in the length of the time frame between each of the font formats is provided. The length of this specifies how much time has been spent after the HTML has been parsed to further download, parse, and render external resources [27].

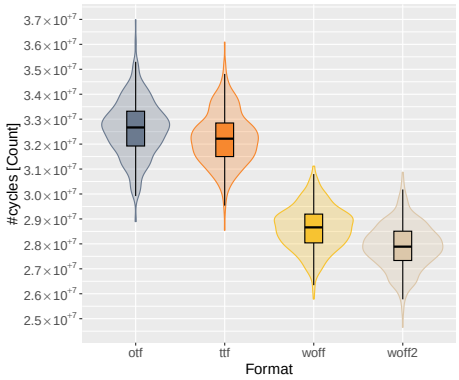
As illustrated in Fig. 2b, there are clear differences in the individual font formats considering the web page speed. TrueType Fonts had the lowest performance in terms of the speed at which the fonts are loaded, parsed, and rendered, with a median of 209 ms. OpenType fonts already had a slightly shorter length between the two window events considered, with a median of 195 ms. Fonts provided in the Web Open Font Formats were significantly faster in the analysis,

Format	Load Time [ms]	Processor Cycles [Count]
otf	196.0 (189.1, 205.1)	32667944 (31923896, 33319052)
ttf	210.0 (203.5, 219.1)	32217562 (31500902, 32846721)
woff	172.0 (164.9, 180.1)	28661862 (28039596, 29193426)
woff2	161.5 (155.8, 169.3)	27888552 (27333156, 28507697)
Format	Memory Allocations [Mb]	Watt-hours [Wh]
otf	2264.5 (1953.7, 2634.3)	0.00313 (0.00299, 0.00326)
ttf	2727.5 (2306.4, 3152.3)	0.00327 (0.00315, 0.00342)
woff	6235.8 (5157.0, 7540.5)	0.00272 (0.00257, 0.00288)
woff2	5746.6 (4877.8, 6779.1)	0.00259 (0.00247, 0.00273)

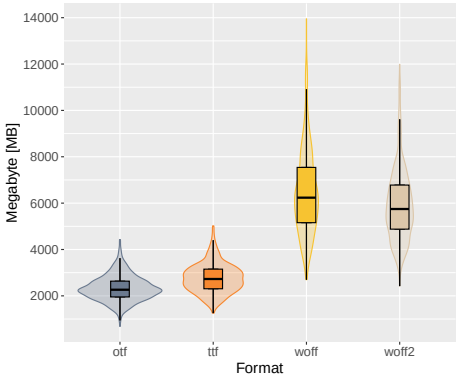
(a) Values given as: median(1^{st} quartile, 3^{rd} quartile)



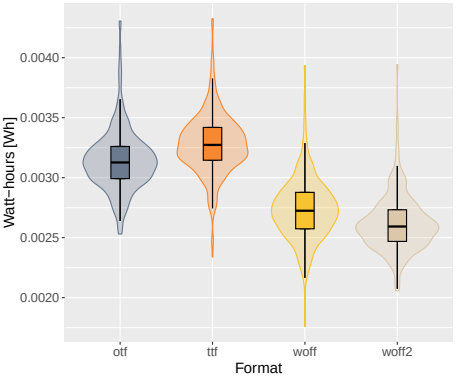
(b) Load Time



(c) Processor Cycles



(d) Average memory allocation changes



(e) Energy Consumption

Fig. 2: Comparison of performance metrics (b)-(d) across the four **web font formats**.

with version one yielding a median of 171 ms and its extension WOFF2 showing the best performance with a median of 161 ms.

Also evident in Figures 2b is the skewness of the distributions. The skewness of each format is positive, indicating a right-skewed distribution ranging from TrueType with a skewness of 1.11 to WOFF2 with a value of 1.95. These values suggest that more values on the upper end show a longer loading time than the median. Similarly, all four distributions show a positive kurtosis, ranging from 7.00 (TrueType) to 11.2 (WOFF), indicating leptokurtic distributions. This signalizes the required time between both events to contain more extreme outliers than a normal distribution, especially at the upper end.

The results of the one-way ANOVA revealed that there was a statistically significant difference in the document loading time between at least two formats ($F(3, 1996) = 1138, p < 2.2e^{-16}$). Thus, the null hypothesis H_0 can be rejected. A Tukey HSD test shows a significant difference ($p_{adj} = 0$) between all pairwise format comparisons. With η^2 value (and partial η^2 value) equals 0.631, the effect size of format on load time is considered as high [7].

4.2 Processor Cycles

As a metric of resource consumption, the results of the processor cycles are presented. An overview of the sum of required processor cycles is shown in Fig. 2c.

As illustrated in 2c, variations depend on the font format. In terms of the required processor cycles, using WOFF2 results in the lowest CPU consumption, with a mean of 27.89 million cycles. The second-best performance was achieved by using the WOFF font format, with a median of 28.66 million cycles. This is followed by TrueType fonts (32.21 M) and OpenType fonts (32.67 M). With a correlation coefficient of 0.736, the correlation between the length of the time frame and the required processor cycles is high.

Considering the skewness of the processor cycles concerning the format, very symmetrical distributions are observed, as the values range from -0.064 (WOFF) to 0.177 (TrueType). Furthermore, the kurtosis of the distributions are similar to those of a normal distribution, as all values are within the range of 2.97 (WOFF) and 3.45 (OpenType).

The null hypothesis associated with H_2 can be rejected, stating that the format does not influence the processor utilization measured in the number of cycles required. The results revealed that there is a statistically significant difference in the processor cycles between at least two formats ($F(3, 1996) = 2903, p < 2.2e^{-16}$). The Tukey HSD test indicates that there are significant differences between all pairs ($p_{adj} = 0 < 0.01$). According to the calculated η^2 value of 0.813, the effect size of the format on the processor cycles required is considered high [7].

4.3 Memory Allocation

The profiler results regarding the required memory allocations are presented. An overview of the relative changes to the allocated memory is shown in Fig. 2d. It is noted that the values provided are the 10% trimmed means of the relative memory allocation changes. The distribution of the metric differs depending on the format. While OpenType fonts and TrueType fonts show few additional memory allocations with medians of 2264 and 2728 allocated bytes, respectively, the web fonts have significantly more memory allocations with medians of 6236 (WOFF) and 5747 (WOFF2) allocated bytes.

The allocation of memory is symmetric for all formats with skewnesses in the range of 0.493 (TrueType) to 0.874 (WOFF2). Similarly, the kurtosis of the distributions is similar to those of a normal distribution, ranging from 3.43 (TrueType) to 4.19 (WOFF2).

Furthermore, a clear difference in the spread between system fonts and web fonts is illustrated. While system fonts have a standard deviation of 545 (OpenType) and 630 (TrueType) bytes, a greater dispersion is shown for WOFF and WOFF2, whose average deviation from the mean varies by 1846 and 1540, respectively.

Again, a one-way ANOVA test is performed. The H_3 associated null hypothesis states that the format has no effect on the memory allocations required. The results revealed that there was a statistically significant difference in memory consumption between at least two formats ($F(3, 1996) = 1381, p < 2.2e^{-16}$). Further analysis with Tukey HSD shows a significant difference between all pairwise groups. Similarly to the prior resource metrics, the effect sizes are considered as high according to an η^2 value of 0.678.

4.4 Energy Consumption

Finally, the energy consumption of each of the formats is analyzed by specifying the watts per hour consumed in the considered time frame, i.e between the two events. A violin plot in Fig. 2e is provided.

By only looking at the required watts per hour, web fonts can show a better performance related to energy consumption. WOFF2 formats required the least energy with a median of 2.59 Milliwatts per hour (mWh), followed by the older version WOFF with 2.72 mWh and OTF with 3.13 mWh. The lowest performance in terms of energy consumption was achieved by using TrueType fonts, with a median of 3.27 mWh.

To test H_4, whether the font format affects the energy consumption while the document is loaded, a one-way ANOVA is conducted. The results of this test revealed that there was a statistically significant difference in energy consumption between at least two formats ($F(3, 812) = 1381.p < 2.2e^{-16}$). The Tukey HSD test shows a significant difference between all pairwise font formats. The effect size is strong according to the η^2 value of 0.55. Therefore, the associated null hypothesis is rejected, stating that the font format does not influence energy consumption.

5 Discussion

Fonts play a crucial role in design and typography. In the competitive market of web applications, they need to have a distinctive appeal and value proposition. Consequently, it has become customary to employ custom-designed fonts in one of the four prevalent formats: TTF, OTF, WOFF, and WOFF2. However, incorporating these font formats can introduce performance challenges that potentially compromise user satisfaction, particularly in regions with limited data bandwidth, such as developing countries. Thus, the selection of an appropriate format assumes paramount importance. Through the analysis, we can reject the null hypotheses associated with H_1 to H_4, as compelling evidence has emerged highlighting a significant impact of the font format on all four dependent variables.

H1: The results confirm the recommendations mentioned by the gray literature [17, 36] about the usage of the font formats. The web fonts are able to show a faster loading time in contrast to the system fonts. Usage of the newest Web Open Font Format version 2 resulted in the best loading time, followed by WOFF, OpenType and TTF.
H2: The process utilization results were contrary to the expectations since a higher CPU utilization was expected for web fonts due to the necessary decompression. The possible reason for these contradictory results is assumed to be the longer network connection, which leads to a longer observation period for system formats with respect to the processor cycles (idle time). The longer network connection may require additional processing cycles, compensating for decompression's necessary disadvantage.
H3: An examination of the performance of the different formats concerning the allocated memory resulted in the conclusion that system fonts (TrueType and OpenType) require significantly less memory than web fonts. Although the observation period is longer for system fonts, OpenType fonts were able to minimize the required allocated memory, with TrueType fonts following closely with 1.2 times more allocations. Using WOFF2 resulted in 2.53 times more memory allocations on average; for WOFF the factor increases to 2.75. To prioritise memory consumption, it is advisable to specify fonts with fallback formats in this order.
H4: The last performance metric examined is the energy consumption required to load the web page with respect to the font formats used. The results show that the usage of font formats with fallback formats in the following order can reduce energy consumption: WOFF2, WOFF, OpenType, and TrueType. Web developers with a high prioritization of energy efficiency are recommended to use this order.

To sum up, we suggest the following key takeaways to both researchers and practitioners:

1. TTF and OTF fonts have been established for a considerable period, making them widely adopted and offering an extensive range of styles.
2. Overall, WOFF2 is the favored font format due to its superior performance, as predicted by gray literature (e.g., [17, 17, 36]). Hence, a wrapper like `ttf2woff2` seems to be the appropriate and recommended method.
3. It must be noted that WOFF2 might face limited support in specific browsers. As a result, it is advisable to consider using WOFF as a fallback option to ensure broader compatibility.

6 Threats to Validity

Threats to Validity refer to factors that may undermine the reliability and generalizability of this study. We split those into four types based on Cook and Campbell [8] that we want to discuss further:

Internal Validity: In order to mitigate this risk, we took into consideration the requirements for reliable benchmarking from Beyer et al. [3]. Concretely this means we decided to completely automate the test execution using AutoHotkey, which also allows independent replication and verification of the experiment. Similarly, we have used common practices in order to keep the performance measurement stable using, for instance, a timeout window between the runs, a specific configuration to avoid external influences, and conducted 500 test runs per web font format to recognize potential outliers.

External Validity: As there exist many font styles, we combined three font styles as this is a typical design principle [37]. Nonetheless, while our combination encompasses a variety of font styles, it is worth considering the inclusion of further styles as a potential avenue for future research. Moreover, to retrieve the targeted performance metrics, our study was constrained to a concrete experimental setup (described in Section 3). This limitation adversely impacts the generalizability of our findings, given the multitude of alternative browsers and client devices available in the market. However, it is noteworthy that our results confirm prevailing assumptions documented in the grey literature, lending support to their relevance across different experimental configurations.

Construct Validity: In terms of the construct design and poor operationalization, we carefully outlined the design process. We selected the performance metric in a way that direct impacts to hardware effects were identifiable. Specifically, CPU cycles, allocated and deallocated memory, and the loading time. For a general metric, we selected the energy consumption as an overall is expected to provide a good overview of how efficient the website is in providing the expected outcome.

Statistical Conclusion Validity: This threat is addressed by using statistical hypothesis tests, namely *one-way ANOVA* and *Tukey HSD*, such that the conclusions drawn from this study are founded on common data analysis practices. Furthermore, the results are checked against grey literature, which was selected by specific inclusion criteria. Furthermore, we have fully disclosed all findings and test materials in Zenodo [10] obtained by our experiments.

7 Conclusion and Future Work

Loaded via CSS, font formats give web developers the opportunity to individualize their visual representation, with the drawback of performance downturns. Hence, the selection of an appropriate font format is essential. This study aims to provide insights into the prevailing font formats, TTF, OTF, WOFF, as well as its second generation. Our benchmarking shows that WOFF2 surpasses all other types, albeit with a higher memory allocation. Thus, we conclude that practitioners should employ WOFF2 and consider converting other formats to WOFF2 when feasible. Other optimization options mentioned in chapter 2, such as font subsetting or the hosting method used, also influence the web application's performance and could be part of further elaboration.

8 Acknowledgment

This work has been supported by and done in the scope of the ITEA3-SmartDelta project, which has been funded by the Austrian Research Promotion Agency (FFG, Grant No. 890417).

References

1. An, D.: Find out how you stack up to new industry benchmarks for mobile page speed. Think with Google-Mobile, Data & Measurement p. 24 (2017)
2. Barashkov, A.: Advanced web font optimization techniques - Pixel Point (2022), https://pixelpoint.io/blog/advanced-web-font-optimization-techniques/
3. Beyer, D., Löwe, S., Wendler, P.: Reliable benchmarking: Requirements and solutions. International Journal on Software Tools for Technology Transfer **21**(1), 1–29 (Feb 2019). https://doi.org/10.1007/s10009-017-0469-y
4. Bhatia, S.K., Samal, A., Rajan, N., Kiviniemi, M.T.: Effect of font size, italics, and colour count on web usability. International journal of computational vision and robotics **2**(2), 10.1504/IJCVR.2011.042271 (Apr 2011). https://doi.org/10.1504/IJCVR.2011.042271
5. Bühler, P., Schlaich, P., Sinner, D., Bühler, P., Schlaich, P., Sinner, D.: Schrifttechnologie. Typografie: Schrifttechnologie-Typografische Gestaltung-Lesbarkeit pp. 73–84 (2017)
6. Cao, B., Shi, M., Li, C.: The solution of web font-end performance optimization. In: 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI). pp. 1–5. IEEE, Shanghai (Oct 2017). https://doi.org/10.1109/CISP-BMEI.2017.8302083

7. Cohen, J.: Statistical Power Analysis for the Behavioral Sciences. Lawrence Erlbaum Associates (1988)
8. Cook, T.D., Campbell, D.T.: Quasi-Experimentation: Design & Analysis Issues for Field Settings. Houghton Mifflin, Boston (1979)
9. Dornauer, B., Felderer, M.: Energy-Saving Strategies for Mobile Web Apps and their Measurement: Results from a Decade of Research. In: 2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft). pp. 75–86. IEEE, Melbourne, Australia (May 2023). https://doi.org/10.1109/MOBILSoft59058.2023.00017
10. Dornauer, B., Vigl, W., Felderer, M.: On the Role of Font Formats in Building Efficient Web Applications - Replication Package (Aug 2023). https://doi.org/10.5281/ZENODO.8247573
11. Firefox, M.: Firefox profiler (2023), https://profiler.firefox.com/
12. Froidure, N.: Ttf2woff2 - npm (2022), https://www.npmjs.com/package/ttf2woff2
13. Gray, S., Mallet, C.: AutoHotkey, https://www.autohotkey.com/
14. Grigorik, I.: Optimize WebFont loading and rendering (2020), https://web.dev/optimize-webfont-loading/
15. Gupta, P., M, I.O.P.: A Survey of Application Layer Protocols for Internet of Things. In: 2021 International Conference on Communication Information and Computing Technology (ICCICT). pp. 1–6. IEEE, Mumbai, India (Jun 2021). https://doi.org/10.1109/ICCICT50803.2021.9510140
16. Hackenberg, D., Ilsche, T., Schone, R., Molka, D., Schmidt, M., Nagel, W.E.: Power measurement techniques on standard compute nodes: A quantitative comparison. In: IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2013). pp. 194–204. IEEE Computer Society (Apr 2013). https://doi.org/10.1109/ISPASS.2013.6557170
17. Hempenius, K., Pollard, B.: Best practices for fonts. Optimize web fonts for Core Web Vitals. (2022), https://web.dev/font-best-practices/
18. Hunt, P.: Adobe-fonts/source-sans: Sans serif font family for user interface environments (2023), https://github.com/adobe-fonts/source-sans
19. Johnston, N., Vincent, D., Minnen, D., Covell, M., Singh, S., Chinen, T., Jin Hwang, S., Shor, J., Toderici, G.: Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4385–4393 (Jun 2018). https://doi.org/10.1109/CVPR.2018.00461
20. Kaleev, N.: 8 font loading strategies to improve your core web vitals (2022) (2023), https://nitropack.io/blog/post/font-loading-optimization
21. Kew, J., van Blokland, E., Leming, T.: WOFF file format 1.0. W3C recommendation, W3C (Dec 2012)
22. Li, Z.: Cross-Layer Optimization for Video Delivery on Wireless Networks. Doctor, Princeton University (2023), https://dataspace.princeton.edu/handle/88435/dsp01bk128f14w
23. Liew, Z.: The best font loading strategies and how to execute them — CSS-Tricks - CSS-Tricks (2021), https://css-tricks.com/the-best-font-loading-strategies-and-how-to-execute-them/#loading-fonts-with-self-hosted-fonts
24. Ling, J., van Schaik, P.: The influence of font type and line length on visual search and information retrieval in web pages. International Journal of Human-Computer Studies **64**(5), 395–404 (2006). https://doi.org/10.1016/j.ijhcs.2005.08.015
25. McInerney, M.: Impallari/Raleway: Raleway fonts (2016), https://github.com/impallari/Raleway

26. Morey, R.: A guide to web font optimization (2022), https://wp-rocket.me/blog/guide-web-font-optimization/
27. mozilla: Window: DOMContentLoaded event - web APIs MDN (Apr 2023), https://developer.mozilla.org/en-US/docs/Web/API/Window/DOMContentLoaded_event
28. mozilla: Window: Load event - web APIs MDN (Apr 2023), https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event
29. Nielsen, J.: Designing Web Usability: The Practice of Simplicity. New Riders Publishing, USA (1999)
30. Olsson, M., Olsson, M.: Font. CSS3 Quick Syntax Reference: A Pocket Guide to the Cascading Style Sheets Language pp. 67–70 (2019)
31. Ouyang, J., Luo, H., Wang, Z., Tian, J., Liu, C., Sheng, K.: FPGA implementation of GZIP compression and decompression for IDC services. In: 2010 International Conference on Field-Programmable Technology. pp. 265–268 (2010). https://doi.org/10.1109/FPT.2010.5681489
32. Pearrow, M.: Web Site Usability Handbook with Cdrom. Charles River Media, Inc., USA (2000)
33. Rello, L., Pielot, M., Marcos, M.C.: Make it big! The effect of font size and line spacing on online readability. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. pp. 3637–3648. CHI '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2858036.2858204
34. Rodriguez Fernandez, M., Zalama Casanova, E., Gonzalez Alonso, I.: Review of display technologies focusing on power consumption. Sustainability **7**(8), 10854–10875 (2015)
35. Semykin, V.: Ttf2woff2 - npm (2021), https://www.npmjs.com/package/ttf2woff2
36. Stein, B.: The 2022 Web Almanac: Fonts. Tech. Rep. 5, HTTP Archive (Sep 2022), https://almanac.httparchive.org/en/2022/fonts
37. Tidwell, J., Brewer, C., Valencia, A.: Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Media, Sebastopol, CA, third edition edn. (2020)
38. Took, R.: Putting design into practice: Formal specification and the user interface. In: Formal Methods in Human-Computer Interaction, pp. 63–96. Cambridge University Press, USA (1990)
39. Ulanovsky, J.: Montserrat/Montserrat-SemiBold.otf at master · JulietaUla/Montserrat · GitHub (2021), https://github.com/JulietaUla/Montserrat/blob/master/fonts/otf/Montserrat-SemiBold.otf
40. Van Riet, J., Malavolta, I., Ghaleb, T.A.: Optimize along the way: An industrial case study on web performance. Journal of Systems and Software **198**, 111593 (Apr 2023). https://doi.org/10.1016/j.jss.2022.111593
41. Wagner, J.L., Marcotte, E.: Web Performance in Action: Building Fast Web Pages. Manning Publications Co, Shelter Island, NY (2017)
42. Willis, M., Hanna, J., Encinas, E., Auger, J.: Low Power Web: Legacy Design and the Path to Sustainable Net Futures. In: Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems. pp. 1–14. CHI EA '20, Association for Computing Machinery, New York, NY, USA (Apr 2020). https://doi.org/10.1145/3334480.3381829
43. Wright, T.: History and technology of computer fonts. IEEE Annals of the History of Computing **20**(2), 30–34 (1998). https://doi.org/10.1109/85.667294
44. Zhao, N., Cao, Y., Lau, R.W.: Modeling fonts in context: Font prediction on web designs. In: Computer Graphics Forum. vol. 37, pp. 385–395. Wiley Online Library (2018)