© 2026 Arabian International Conference on Science,Technology and Innovation. All rights reserved.

AI-DRIVEN APPROACHES TO REDUCING WEB PAGE LOAD TIME

Nigorakhon Khamidjonova Botir qizi
Master student, Sharda University Uzbekistan
+998 93 2521844
shamsuddinovanigoraxon@gmail.com
https://doi.org/10.5281/zenodo.24142715

Annotation. Load time is one of the main factors that determine whether a visitor stays on a website, and conventional optimization relies on fixed rules written in advance by a developer. This thesis considers how artificial intelligence (AI) and machine learning (ML) can make load time optimization adaptive. Five directions are examined: predictive prefetching, intelligent media optimization, adaptive loading, AI-assisted code optimization and intelligent performance monitoring. A four-stage cyclic model that combines these directions is proposed, together with a plan for evaluating it by Core Web Vitals metrics.

Key Words. Artificial intelligence, machine learning, web performance, page load time, frontend development, Core Web Vitals, predictive prefetching, image optimization, adaptive loading, performance monitoring.

ПОДХОДЫ НА ОСНОВЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА К СОКРАЩЕНИЮ ВРЕМЕНИ ЗАГРУЗКИ ВЕБ-СТРАНИЦ

Аннотация. Время загрузки является одним из главных факторов, определяющих, останется ли посетитель на сайте, а традиционная оптимизация опирается на фиксированные правила, заранее заданные разработчиком. В работе рассматривается, как искусственный интеллект (ИИ) и машинное обучение (МО) позволяют сделать оптимизацию времени загрузки адаптивной. Изучены пять направлений: предиктивная предзагрузка, интеллектуальная оптимизация медиафайлов, адаптивная загрузка, оптимизация кода с помощью ИИ и интеллектуальный мониторинг производительности. Предложена четырёхэтапная циклическая модель, объединяющая эти направления, и план её оценки по показателям Core Web Vitals.

Ключевые слова. Искусственный интеллект, машинное обучение, производительность веб-сайтов, время загрузки страницы, фронтенд-разработка, Core Web Vitals, предиктивная предзагрузка, оптимизация изображений, адаптивная загрузка, мониторинг производительности.

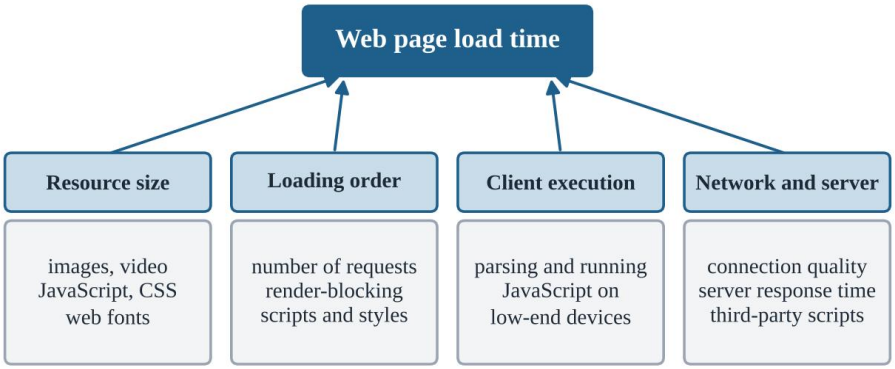
INTRODUCTION

Modern websites are no longer simple documents. A typical page loads large JavaScript bundles, high-resolution images, fonts and a number of third-party scripts, while a growing share of visitors open it on a mid-range smartphone over a mobile network. In this situation load, time directly affects business results: according to Google's research, 53% of mobile visits are abandoned if a page takes longer than three seconds do to load [3]. Since 2020, the quality of loading has also been measured by the Core Web Vitals metrics, which Google uses as one of its search ranking signals [1].

The usual set of optimization techniques includes magnification, compression, caching, content delivery networks and lazy loading. They are effective, but each of them is a static rule: the developer decides once what to load and when, and this decision is the same for every visitor. Such rules ignore how people actually move through the site, what device they use and how good their connection is. Artificial intelligence makes it possible to replace part of these fixed rules with decisions that are learned from real usage data. The aim of this thesis is to systematize AI-driven approaches to reducing load time and to propose a model that unites them in one process.

© 2026 Arabian International Conference on Science,Technology and Innovation. All rights reserved.

Factors of load time and its measurement. The factors that slow a page down can be divided into four groups (Picture 1): the size of resources, the order in which they are loaded, the cost of executing code on the client, and the conditions of the network and server. Each group requires its own optimization methods, and in practice a slow page usually suffers from several of them at once.



Picture 1. *Main groups of factors affecting web page load time.*

To assess the result objectively, three Core Web Vitals metrics are used (Table 1). The thresholds are evaluated at the 75th percentile of real visits, so a page is considered good only when at least three quarters of its users have a good experience [2].

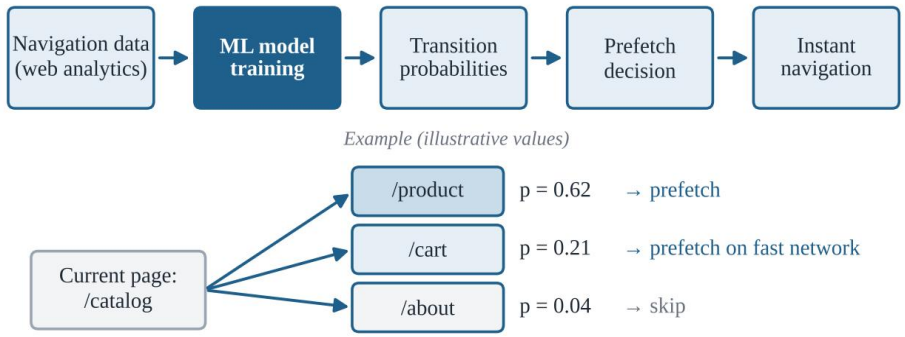
Table 1. *Core Web Vitals metrics and their thresholds.*

Metric	What it measures	Good	Needs improvement	Poor
Largest Contentful Paint (LCP)	Loading of the main content	≤ 2.5 s	2.5–4.0 s	> 4.0 s
Interaction to Next Paint (INP)	Response to user actions	≤ 200 ms	200–500 ms	> 500 ms
Cumulative Layout Shift (CLS)	Visual stability	≤ 0.1	0.1–0.25	> 0.25

AI-driven approaches. The analysis of existing tools and publications allows five directions to be distinguished in which AI is applied to reduce load time.

Predictive prefetching. Prefetching loads the resources of the next page before the user clicks a link. If it is applied to every link, it wastes traffic and competes with the current page for bandwidth. An ML model solves this by estimating, from the history of navigation recorded by web analytics, the probability of each transition from the current page, so that only the most probable targets are fetched in advance (Picture 2). The open-source project Guess.js implements this idea and is integrated with Angular, Next.js, Nuxt.js and Gatsby. Its authors refer to an experiment in which ML-based prefetching improved page load times by about 30% [4].

© 2026 Arabian International Conference on Science,Technology and Innovation. All rights reserved.



Picture 2. *Scheme of predictive prefetching based on machine learning.*

Intelligent media optimization. Images make up the largest part of the weight of most pages, and the largest image is often the LCP element. AI-based services analyze the content of every image and choose its format (for example, WebP or AVIF), degree of compression and dimensions individually, instead of applying one quality setting to the whole site. Computer vision models also find the semantically important region of an image, which allows it to be cropped correctly for small screens.

Adaptive loading. The same page is opened on a powerful computer and on an inexpensive phone. Signals available in the browser, such as the effective connection type, the amount of device memory and the number of processor cores, can be used by a classifier to assign the visitor to a capability class. Depending on this class the site delivers lighter images, postpones non-essential scripts or switches off heavy animations, which shortens loading for weak devices without reducing quality for the rest.

AI-assisted code optimization. Excess JavaScript delays both loading and interactivity. Large language model assistants help a developer to find unused code, to split a bundle into parts that are loaded on demand and to replace heavy libraries with lighter ones. In a controlled experiment, developers who used an AI pair programmer completed a programming task 55.8% faster than the control group [6]; this suggests that routine performance refactoring, which is often postponed for lack of time, becomes cheaper with such tools.

Intelligent performance monitoring. Performance tends to degrade unnoticed after releases or changes in third-party scripts. Anomaly detection models applied to real user monitoring data reveal a deviation of LCP or INP soon after it appears and indicate its probable source, such as a particular page, script or device type.

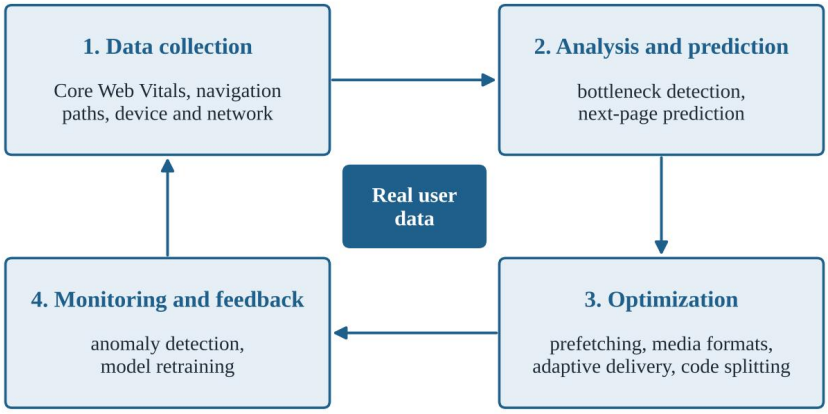
The relation between these approaches, the problems they address and the metrics they influence is summarized in Table 2.

© 2026 Arabian International Conference on Science,Technology and Innovation. All rights reserved.

Table 2. *AI-driven approaches and the metrics, they are aimed at (compiled by the author)*

Approach	AI technique	Problem addressed	Main metric
Predictive prefetching	Probabilistic navigation models	Slow transitions between pages	LCP of next page
Media optimization	Computer vision, learned compression	Heavy images and video	LCP, page weight
Adaptive loading	Classification of device and network	Weak devices, slow connections	LCP, INP
Code optimization	Large language model assistants	Excess and unused JavaScript	INP, page weight
Performance monitoring	Anomaly detection	Unnoticed regressions	All metrics

Applied separately, the listed approaches give a local effect. It is proposed to combine them into one cycle of four stages (Picture 3). At the first stage real user data are collected: Core Web Vitals values, navigation paths, and device and network characteristics. At the second stage ML models find bottlenecks, predict the next page and classify the visitor's conditions. At the third stage the decisions are applied automatically: prefetching, choice of media variants, adaptive delivery and recommendations on code splitting. At the fourth stage the results are measured, anomalies are detected and the models are retrained on fresh data. Thus optimization stops being a one-time task and becomes a continuous process that adjusts itself to changes in the site and its audience.



Picture 3. *Cyclic model of AI-based web page load time optimization (proposed by the author).*

Evaluation plan. The model is to be tested on two versions of one web application: a baseline optimized by traditional methods and a version with the AI-based techniques. Laboratory measurements will be made with Lighthouse on several device and network profiles, and field data will be collected by real user monitoring. The compared indicators are LCP, INP, CLS, total page weight, the number of requests and the developer's time spent on optimization.

Limitations. Predictive models need a sufficient volume of navigation data, which small sites may lack. A wrong prediction leads to unnecessary downloads, so prefetching should be restricted for users

© 2026 Arabian International Conference on Science,Technology and Innovation. All rights reserved.

with limited data plans. A model that runs in the browser must be lightweight, otherwise it will itself slow the page down. Code suggested by AI assistants requires review by the developer, and the collection of behavioural data must comply with privacy legislation.

Conclusion

Rule-based optimization treats all visitors identically, whereas the real conditions of loading differ from user to user. The AI-driven approaches considered in this thesis remove this limitation in different parts of the loading process: prediction of navigation shortens transitions between pages, content-aware media processing reduces page weight, classification of devices and networks adapts delivery, language model assistants lower the cost of code optimization, and anomaly detection protects the achieved results. The proposed cyclic model links these approaches through shared real user data and can serve as a basis for integrating AI into the frontend development workflow. Its experimental verification by Core Web Vitals metrics is the subject of the author's further research.

REFERENCES

1. Walton P. Web Vitals. – web.dev, Google, 2020. – URL: https://web.dev/articles/vitals
2. McQuade B. How the Core Web Vitals metrics thresholds were defined. – web.dev, Google, 2020. – URL: https://web.dev/articles/defining-core-web-vitals-thresholds
3. The need for mobile speed: How mobile latency impacts publisher revenue. – Think with Google, 2016. – URL: https://www.thinkwithgoogle.com
4. Guess.js: Libraries and tools for enabling machine learning driven user experiences on the web. – URL: https://github.com/guess-js/guess