ISSN 2224-087X. Електроніка та інформаційні технології. 2024. Випуск 27. С. 34–42
Electronics and information technologies. 2024. Issue 27. P. 34–42

УДК 004.9 DOI: https://doi.org/10.30970/eli.27.3

COMPARATIVE ANALYSIS THE PERFORMANCE OF CLIENT-SIDE AND SERVER-SIDE MACHINE LEARNING TECHNOLOGIES

I. Mysiuk, R. Shuvar

Department of System Design,
Ivan Franko National University of Lviv
50 Drahomanova St., UA-79005 Lviv, Ukraine
iruna.musyk8a@gmail.com

The performance analysis of client-side and server-side machine learning technologies is important for understanding the optimal way to model optimization. The study aims to analyze the training time of the model, taking into account parameters such as the number of likes, comments and shares according to the text of a post in social networks. Natural language processing (NLP) requires significant computing power, so it is important to determine whether it is more efficient to train models on client devices or on servers. TensorFlow for JavaScript can provide client-side computation, while Python can use server-side resources. The obtained results confirm that the models in web machine learning require optimization and are slower than in the server implementation, taking into account the training execution time. Therefore, the size of the data is important for effective machine learning of the model in client-side computing.

Keywords: machine learning, client-side computing, server-side computing, model training, text analysis, natural language processing, TensorFlow, learning time, social networks.

Overview

The relevance of using machine learning on the client and server side is constantly increasing in the modern information environment, where speed, efficiency and data protection are becoming key priorities. The latest trends in this area indicate the growing popularity of client-side machine learning using technologies such as TensorFlow for JavaScript and TensorFlow Lite to perform computations directly on user devices such as mobile phones and IoT devices. This allows you to provide personalized and fast solutions, reducing the dependence on server resources and ensuring greater data privacy. On the other hand, server-based machine learning remains popular in large enterprises and data centers where significant computing resources are available. The use of servers allows working with large volumes of data and complex models, ensuring high learning and processing speed.

Benefits of client-side machine learning include reduced load on servers, increased speed, and independence from the Internet. At the same time, its disadvantages are the limited computing capabilities of devices and the limited amount of data available for training.

Server-based machine learning has great power and scalability but requires significant computing resources and can cause data privacy issues. However, servers are still a necessary element for processing large amounts of data and performing complex calculations. In general, both approaches have their own unique advantages and disadvantages, and the choice between them depends on the specific needs, constraints, and requirements of the project. However, the

trend of increasing the value of client machine learning indicates the prospects of this direction of development in the future.

Machine learning (ML) in web applications is rapidly evolving, enhancing user experiences and extracting meaningful insights from vast data. TensorFlow.js allows developers to build and run ML models in JavaScript directly in the browser, enabling real-time processing, improved accessibility, and enhanced security. Description of basic flows building ML model and official documentation [1, 2] highlight these advantages. At the same time, in paper [3] emphasize the ease of deploying ML models in web environments using TensorFlow.js. Authors in [4] discusses broad ML applications, including content recommendation, sentiment analysis, and user behavior prediction, highlighting their real-world relevance. This means that despite ready-made implementations, there is a need for improved models for their specifications for each task separately.

In work [5] is showing how these techniques enhance content creation and management with reviewing ML approaches in web-based composition. It shows the practical aspects of this subject, which can be interesting for analytics and marketing.

The models of information collection, processing and analysis are based on the work done in previous works [7–9]. It is investigation of ML's role in analyzing news and social media posts, focusing on sentiment analysis to interpret public opinion and trends in [10] analyze consumer behavior using ML algorithms, helping businesses understand customers better and tailor their offerings.

User behavior analytics is another critical application, as it is described in [11] by combining ML with eye-tracking data to gain deeper insights into web user behaviors. Similarly, in [12, 13] survey clustering algorithms and ML models for user and entity behavior analytics, emphasizing the importance of segmenting users based on behavior. In articles [14, 15] highlight the use of ML techniques to predict user preferences, enhancing content and service personalization. In article [16] focus on using data analytics and ML to distinguish between malicious and legitimate users, enhancing web application security.

Summarizing this sufficient list of studies, it can be assumed that a constant search for solutions for the problem of user behavior analysis is taking place. As a result, it is worth considering the possibilities for finding alternative ways and optimizing such software solutions (in particular, in web mode). As ML research and development progress, innovative applications will continue to enhance user experience and operational efficiency in web environments.

Performance evaluation methods and features of Tensorflow implementations

Regardless of the type of program implementation and, accordingly, the programming language (on the server or web), the following steps for program implementation can be formulated:

1. Downloads data from the "*uba.json*" file using an asynchronous fetch request.
2. Executes the *extractData* function for each object in the data array
3. Filters the data, removing objects where one of the fields (x or y) has a null value.
4. Displays the original data on a graph using the plot function.
5. Shuffles the data to prepare it for model training.
6. Converts input data into tensors and normalizes them.
7. Creates a TensorFlow model using two dense (fully connected) layers.
8. Compiles the model, specifying the losses and the optimizer.
9. Trains the model using the *trainModel* function.

10. Restores normalized data to raw data and displays it on a graph.

11. Shows the prediction results on a graph compared to the original data.

Regarding the significant difference between the Python and JavaScript versions of TensorFlow, the main difference is that TensorFlow.js is designed specifically for the web environment, so it has its own architecture and features tailored to work in the browser. In other respects, the functionality is similar, and both versions of the library allow you to conveniently work with ML models.

Features of using TensorFlow.js are the ability to perform ML directly in the web environment, without the need to install specialized libraries. This makes it easier and faster to develop web applications with intelligent functions such as pattern recognition or speech processing.

TensorFlow.js is commonly used in JavaScript to train models in the browser or on the server. We are used the performance.now() method to measure time, but it depends on the context of use.

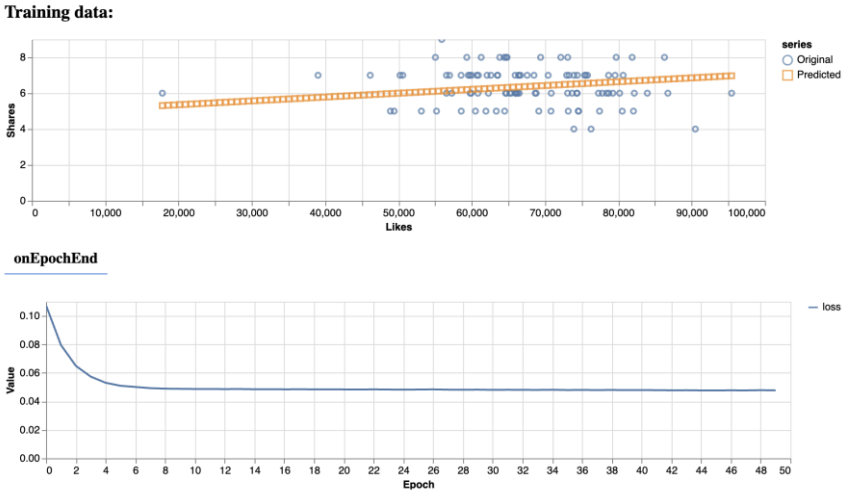

Fig. 2. The main components of a web page with visualization of the learning process with Tensorflow.js

Returning to the input data that was used to train the model, it includes the numerical values of the number of likes, shares, comments and the text of the post on the Facebook social network collected from newspaper news in the same way as it is described in [8, 9]. The "uba.json" file is an array of values that has the following structure:

{"NumberOfLikes": 795, "NumberOfComments": 58, "NumberOfShares": 7, "TextInPost": "Taquería El Califa de León, a taco stand in Mexico City, earned a Michelin star this month. It was “incredible,” said Arturo Rivera Martínez, who has worked there for 20 years, because “in the end, it’s a taquería and a very simple taco” that earned it." }

On Fig. 2 shows two graphs, which allows you to analyze all interactions in ML model preparation. To visualize the process of ML on the web page, it is used:

- *tfvis.render.scatterplot* which is a function from the TensorFlow.js Visualization library (TF.js Vis) that is designed to create a scatter plot for data visualization. This feature allows you to display data quickly and easily as points on a graph, where each point is represented by two or more values (such as coordinates).

- *tfvis.show.fitCallbacks* which is a function from the TensorFlow.js Visualization (TF.js Vis) library that allows you to display graphs of metrics while training a model using TensorFlow.js. This function creates an object with callbacks that automatically update the metric visualizations (in this case loss) during each epoch of model training.

Batch size is the number of training examples utilized in one iteration of the model's training process. The batch size parameter indicates the number of training samples the model processes before updating its internal parameters (weights). Decreasing the batch size and increasing the number of epochs did not significantly affect the loss values for larger model sizes, so Table 1 shows a batch size of ***10*** and a number of epochs of ***50***.

In this case, a batch size of ***10*** means that the model will look at ***10*** training examples at a time, calculate the error for those ***10*** examples, and then adjust the model parameters accordingly. ***50*** epochs provide enough opportunities for the model to learn and adjust its parameters based on the training data. This value is chosen based on prior experiments or benchmarks showing that this number of epochs typically results in good performance without overfitting. More epochs generally mean longer training time. The model will go through the dataset multiple times, learning and adjusting the weights in each pass.

To measure the training time of a model with different amounts of data, different values of parameters batch size and epochs, similar methods are used in Python and JavaScript, but taking into account the peculiarities of each language. In Python, you can use the TensorFlow library, which has the tf.keras module, which provides a fit() method to train the model. To measure training time, you can use the time or timeit module, as shown in the previous example. The batch size and epochs parameters are passed directly to the fit() method.

Table. 1. The results of the learning loss value for different model sizes are obtained

Model size	Loss
100	0.046
1000	0.030
2000	0.023
5000	0.022

There is a slight difference in the code implementation in the JavaScript and Python programming languages, as shown in Fig. 3 and Fig. 4.

```
async function trainModel(model, inputs, labels, surface) {
  const batchSize = 10;
  const epochs = 50;
  const callbacks = tfvis.show.fitCallbacks(surface, ['loss'], {callbacks:['onEpochEnd']});

  const startTime = performance.now();
  await model.fit(inputs, labels, {
    batchSize, epochs, shuffle: true, callbacks: callbacks
  });
  const endTime = performance.now();

  const trainingTime = (endTime - startTime) / 1000;
  console.log(`Training Time: ${trainingTime.toFixed(2)} seconds`);
}
```

Fig. 3. Asynchronous Function to Train the Model in JavaScript

In described data processing pipeline, we utilize functions like "*extract_data*" and "*remove_errors*" to meticulously handle JSON data, ensuring accurate extraction while filtering out records with missing values. Leveraging Matplotlib, the "*plot_data*" function facilitates the creation of intuitive scatter plots, offering comparative insights between original and predicted datasets. Orchestrating these operations, the "*run_tf*" function serves as the backbone, seamlessly managing data loading, normalization, model creation, training as it is shown in Fig.4, prediction, and visualization to enable informed decision-making and actionable insights.

```
batch_size = 10
epochs = 50
start_time = time.time()
history = model.fit(norm_inputs, norm_labels, epochs=epochs, batch_size=batch_size, shuffle=True)
end_time = time.time()

training_time = end_time - start_time
print(f"Training Time: {training_time:.2f} seconds")
```

Fig. 4. Part of code to train model in Python

In the same way as for JavaScript in Fig. 2, results are visualized in the software implementation of server calculations in Python in Fig. 5. The difference is only in the tools and libraries, which have a certain style and minor customization features.

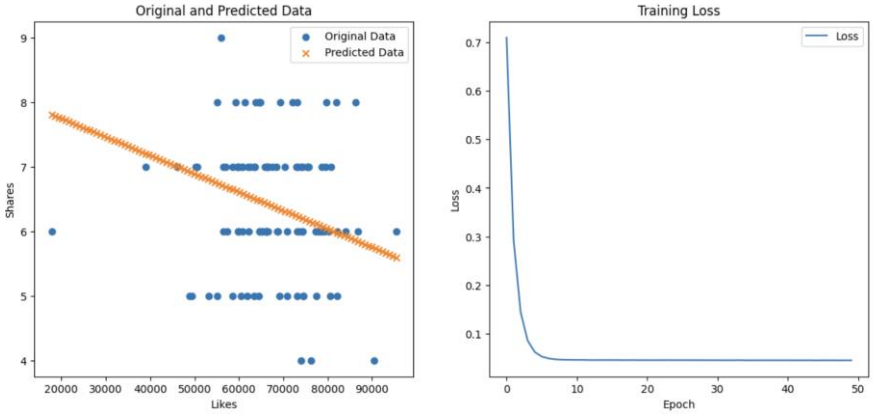

Fig. 5. The main components of a web page with visualization of the learning process with Tensorflow in Python

Summarizing the stage of software implementation, we can conclude that there is no particularly fundamental difference during development. In this context, it is important not to take into account the time of visualization of the results on the chart, because this time may depend directly on the implementation of the chart.

Comparative analysis of the efficiency of using Tensorflow on the client and server side

After implementing the training of the model, we move on to evaluating the numerical and functional indicators of such parameters. In particular, the efficiency in terms of the value of the model training time is considered in accordance with the size of the model. A

comparative analysis of the efficiency of using TensorFlow on the client and server side, along with the training time of the model is shown in Table 2.

Table. 2. Training time for client-based and server-based classification using Tensorflow

Aspect	**Client-side TensorFlow (JavaScript)**	**Server-side TensorFlow (Python)**
Training Time	Typically, shorter due to client-side resources, but may be limited by client hardware capabilities.	Generally faster, especially for large-scale models and datasets, due to access to powerful server hardware and potential for parallelization.
Hardware Requirements	Relies on client hardware, potentially limited by computational capabilities.	Access to powerful server hardware, including GPUs or TPUs, for accelerated training.
Latency	Low latency for local execution, suitable for real-time or interactive applications.	Higher latency for server requests, but suitable for batch processing or offline tasks.
Accessibility	Easily accessible to users with web browsers, no need for server infrastructure.	Requires server infrastructure for deployment and access.
Scalability	Limited scalability due to client-side constraints and potential for resource exhaustion.	Highly scalable, with potential for parallelization and distributed computing.
Privacy and Security	Data remains on the client side, potentially enhancing privacy and security.	Data transmission to the server introduces potential privacy and security concerns.

Table 2 provides a comparative overview of various aspects related to the efficiency of using TensorFlow on the client and server side, including training time, hardware requirements, accessibility, scalability, latency, and privacy/security considerations. The choice between client-side and server-side TensorFlow depends on specific requirements, such as performance goals, available resources, and privacy concerns.

Table. 3. Training time for client-based and server-based classification using Tensorflow

Model size	**Client-based classification training time using Tensoflow, s**	**Server-based classification training time using Tensoflow, s**
100	2.17	2.47
1000	12.23	7.59
2000	23.90	15.49
5000	58.08	41.34

The results may be determined by the features of the device, namely its computing capabilities. All calculations were performed on HP Pavilion (Intel Core i5-1335U, RAM 16 Gb).

Small Dataset (e.g., <100 samples): Training using client-side TensorFlow (JavaScript) is generally faster due to the utilization of client-side resources and the potential for parallelization. On the other hand, server-side TensorFlow (Python) achieves faster training times owing to access to powerful server hardware.

Medium Dataset (e.g., 100 - 1000 samples): The performance between client-side and server-side TensorFlow is comparable, but client-side training may be limited by the computational capabilities of client hardware. Server-side TensorFlow typically achieves faster training times due to optimized server-side processing and potential parallelization.

Large Dataset (e.g., >1000 samples): Training time on the client-side TensorFlow tends to be longer due to potential resource limitations on client devices. Conversely, server-side TensorFlow generally achieves faster training times, especially with distributed computing and GPU/TPU acceleration.

Client-side TensorFlow exhibits limited scalability due to constraints inherent to client-side execution and the potential for resource exhaustion. In contrast, server-side TensorFlow is highly scalable, offering potential for parallelization and distributed computing.

Server-side TensorFlow benefits from access to powerful server hardware, including GPUs or TPUs, for accelerated training. In contrast client-side TensorFlow relies on the computational capabilities of client hardware, which may be limited.

Client-side TensorFlow has limited optimization options due to its execution environment. Conversely, server-side TensorFlow offers a wide range of optimization techniques, including distributed training and hardware acceleration.

Conclusion

In this work, a study of the learning time on the client and server side for Tensorflow is performed. The training time for client-based and server-based classification using TensorFlow can vary depending on several factors, including the complexity of the model, the size of the dataset, the hardware resources available, and the optimization techniques applied.

For client-based classification using TensorFlow.js (running in the browser), the training time might generally be shorter compared to server-based classification due to the utilization of client-side resources. However, it's essential to consider that client-side training might be limited by the computational capabilities of the user's device, potentially leading to longer training times for more complex models or larger datasets.

On the other hand, server-based classification using TensorFlow in Python (running on a server) can benefit from more powerful hardware resources leading to potentially faster training times, especially for large-scale models and datasets. Additionally, server-based training allows for parallelization and distributed computing, which can further optimize training time for demanding tasks. The obtained results confirm that the approach with calculations on the client side is quite effective on a small amount of data, but is less accurate on a large amount of data.

Overall, while client-based classification might offer more flexibility and accessibility, server-based classification generally provides higher performance and scalability, especially for computationally intensive tasks. The choice between client-based and server-based classification should be based on the specific requirements and constraints of the application, including the available hardware resources, latency considerations, and privacy concerns.

REFERENCES

[1] Building a simple text classification neural network in TensorFlow.js - Medium [Online]. URL: https://medium.com/@GeorgePerry/finding-intent-to-buy-from-instagram-comments-with-tensorflow-js-3f764c132be7

[2] TensorFlow for JavaScript - TensorFlow [Online]. URL: https://www.tensorflow.org/js

[3] *S. Kletz, M. Bertini, and M. Lux*. 2021. Open source column: Deep learning in the browser: TensorFlow JS. SIGMultimedia Rec. 11, 1, Article 4 (March 2019), doi: 10.1145/3458462.3458466

[4] *Sarker, I.H*. Machine Learning: Algorithms, Real-World Applications and Research Directions. SN COMPUT. SCI. 2, 160 (2021), doi: 10.1007/s42979-021-00592-x

[5] *Y. J. Ekie et al*. 2021. Web Based Composition using Machine Learning Approaches: A Literature Review. In Proceedings of the 4th International Conference on Networking, Information Systems & Security (NISS '21). Association for Computing Machinery, New York, NY, USA, Article 48, 1–7, doi: 10.1145/3454127.3457623

[6] *M. R. M. V., Rodriguez C., Navarro Depaz, C., Concha, U.R., Pandey B., S. Kharat, R., Marappan R.* Machine Learning Based Recommendation System for Web-Search Learning. Telecom 2023, 4, 118-134. https://doi.org/10.3390/telecom4010008

[7] *A. Verma, C. Kapoor, A. Sharma and B. Mishra*, Web Application Implementation with Machine Learning, 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2021, pp. 423-428, doi: 10.1109/ICIEM51511.2021.9445368

[8] *Mysiuk I., Mysiuk R., Shuvar R*. Collecting and analyzing news from newspaper posts in Facebook using machine learning. Stuc. intelekt. 2023. Vol. 28, No. 1, P. 147-154, doi: 10.15407/jai2023.01.147

[9] *I. Mysiuk, R. Mysiuk, R. Shuvar, V. Yuzevych*. Methods of analytics of big data of popular electronic newspapers on Facebook. Electronics and information technologies 2022. Vol. 19., P. 66–74, doi: 10.30970/eli.19.6

[10] *V. Shrirame, J. Sabade, H. Soneta, M. Vijayalakshmi*, Consumer Behavior Analytics using Machine Learning Algorithms, 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2020, pp. 1-6, doi: 10.1109/CONECCT50063.2020.9198562

[11] *Castilla, D., Del Tejo Catalá, O., Pons, P. et al*. Improving the understanding of web user behaviors through machine learning analysis of eye-tracking data. User Model User-Adap Inter 34, 293–322 (2024), doi: 10.1007/s11257-023-09373-y

[12] *A. Pierpaolo, M. Antonio, M. Alessio.* A comprehensive investigation of clustering algorithms for User and Entity Behavior Analytics. Frontiers in Big Data, Vol.7, 2024 doi: 10.3389/fdata.2024.1375818

[13] *Martín A. G., A. Fernández-Isabel, I. Martín de Diego, and M. Beltrán*, A survey for user behavior analysis based on machine learning techniques: current models and applications, Applied Intelligence, vol. 51, no. 8, pp. 6029–6055, Jan. 2021, doi: 10.1007/s10489-020-02160-x.

[14] *M. Callara, P. Wira*, User Behavior Analysis with Machine Learning Techniques in Cloud Computing Architectures, 2018 International Conference on Applied Smart Systems (ICASS), Nov. 2018, doi: 10.1109/icass.2018.8651961

[15] *J. Moon, Y. Kim, S. Rho*, User Behavior Analytics with Machine Learning for Household Electricity Demand Forecasting, 2022 International Conference on Platform Technology and Service (PlatCon), Aug. 2022, doi: 10.1109/platcon55845.2022.9932037

[16] *R. Ranjan, S. S. Kumar*, User behaviour analysis using data analytics and machine learning to predict malicious user versus legitimate user, High-Confidence Computing, vol. 2, no. 1, p. 100034, Mar. 2022, doi: 10.1016/j.hcc.2021.100034

ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОДУКТИВНОСТІ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ НА СТОРОНІ КЛІЄНТА ТА СЕРВЕРА

І. Мисюк, Р. Шувар

Кафедра системного проектування,
Львівський національний університет імені Івана Франка,
вул.Драгоманова, 50, м.Львів, 79005, Україна
iruna.musyk8a@gmail.com

Аналіз продуктивності технологій машинного навчання на стороні клієнта та сервера важливий для розуміння оптимального способу оптимізації моделі. Дослідження має на меті проаналізувати час навчання моделі з урахуванням таких параметрів, як кількість вподобань, коментарів і поширень відповідно до тексту публікації в соціальних мережах. Обробка природної мови (NLP) вимагає значної обчислювальної потужності, тому важливо визначити, чи ефективніше навчати моделі на клієнтських пристроях чи на серверах. TensorFlow для JavaScript може забезпечити обчислення на стороні клієнта, тоді як Python може використовувати ресурси на стороні сервера. Отримані результати підтверджують, що моделі в веб додатках з машинним навчанням потребують оптимізації та є повільнішими, ніж у серверній реалізації, враховуючи час виконання навчання. Тому розмір даних важливий для ефективного машинного навчання моделі в обчисленнях на стороні клієнта.

Використання на клієнтській стороні, зокрема за допомогою TensorFlow для JavaScript, може виконувати навчання моделей у проте навчання проводити краще на окремих обчислюваних ресурсах. Використання серверних ресурсів, які базуються на Python, також має свої переваги, особливо в обробці великих обсягів даних або складних моделей, проте деяка втрата часу може спостерігатися під час обміну даних (запит-відповідь).

Важливою частиною цього дослідження є візуалізація результатів в режимі реального часу. В перспективі дослідження можна розширити на повноцінних конвеєр зчитування, опрацювання та аналізу даних з реалізацію систем обробки природної мови у широкому спектрі застосувань, від соціальних мереж до підприємств.

Ключові слова: машинне навчання, клієнтська сторона обчислення, серверна сторона обчислення, аналіз тексту, обробка природної мови, TensorFlow, час навчання, соціальні мережі.

The article was received by the editorial office on 18.06.2024.
Accepted for publication on 01.07.2024.