BOOSTLET.JS: MEDICAL IMAGE PROCESSING PLUGINS FOR THE WEB VIA JAVASCRIPT INJECTION

Edward Gaibor[1] *Shruti Varade* [1] *Rohini Deshmukh*[1] *Tim Meyer* [2]
Mahsa Geshvadi[1] *SangHyuk Kim*[1] *Vidhya Sree Narayanappa*[1] *Daniel Haehn*[1]

[1] University of Massachusetts Boston, Boston, MA, United States
2 University of the Bundeswehr, Munich, Germany

ABSTRACT

Can web-based image processing and visualization tools easily integrate into existing websites without significant time and effort? Our Boostlet.js library addresses this challenge by providing an open-source, JavaScript-based web framework to enable additional image processing functionalities. Boostlet examples include kernel filtering, image captioning, data visualization, segmentation, and web-optimized machine-learning models. To achieve this, Boostlet.js uses a browser bookmark to inject a user-friendly plugin selection tool called *PowerBoost* into any host website. Boostlet provides on-site access to a standard API independent of any visualization framework for pixel data and scene manipulation. Boostlet is natively integrated into SliceDrop, a web-based, interactive viewer for medical imaging data. Boostlets provide a modular architecture and client-side processing capabilities to apply advanced image-processing techniques using consumer-level hardware. The code is open-source and available.

***Index Terms*—** Web-based image processing, JavaScript injection, Biomedical

1. INTRODUCTION

Medical image processing enables visualization and analysis of medical data. Current web-based visualization frameworks provide tools to view these images but have some limitations: non-extensible processing capabilities. Users of these public visualization applications cannot add custom algorithms to process the voxel data for filtering or segmentation directly on the framework. Implementing new image-processing functionalities in current frameworks will require users to use framework-specific API code. In most scenarios, the API code is inaccessible on the client side. For example, a researcher that needs to apply segmentation algorithms on public viewer datasets like OpenNeuro.org [1] would need to download the data, process the data (apply filters, segmentations, etc.) on a different tool, and re-visualize results. This workflow would need to be repeated for each image.

To address these limitations, we present Boostlet.js, which contributes with:

- A one-to-many client-side API that allows adding and integrating large language models (LLMs), kernel filtering, image captioning, and machine learning models via Javascript injection to any host website or framework such as Cornerstone2D [2], NiiVue [3], OpenSeaDragon [4], Xtk [5], and Papaya [6]. With a default fallback in case no compatible framework is detected.
- A native integration with medical imaging viewer application: "SliceDrop Reloaded." Researchers can visualize, modify, and create voxel data on a single platform.

The user only needs to drag and drop the *PowerBoost* as a bookmark into the web bookmarks bar. Once clicked on a compatible website, a menu with all available functionalities will be displayed through JavaScript injection, allowing the user to inject any functionality needed.

We then present the *PowerBoost* user interface in multiple platforms. Finally, we collect and discuss feedback from experts who develop web-based medical image processing algorithms and test the Boostlet library. All our developments and experiments are available as open science at: https://github.com/mpsych/boostlet.

2. RELATED WORK

Many frameworks exist for web-based visualization of biomedical images. They all represent significant contributions to the visualization community, and Boostlet.js can help improve the functionality gaps.

2.1. DICOM and Neuroimaging Visualization Tools

Cornerstone2D.js [2] is acclaimed for its Digital Imaging and Communications in Medicine (DICOM) image-rendering abilities and customizability, making it a staple in clinical settings. **Papaya.js** [6] is dedicated to DICOM and Neuroimaging Informatics Technology Initiative (Nifti) image rendering, providing visualization overlay capabilities and a suite of image control options. **NiiVue.js** [3] leverages WebGL's power for neuroimaging, providing high-performance interactive visualization through direct access to the graphical processing unit in the browser.

Gap: While these tools excel in visualization, they offer in-built image processing features that are non-cross-compatible with each other. Boostlet.js addresses this by extending its functionalities with advanced image processing capabilities that can be easily integrated into them simultaneously through our API.

2.2. High-Resolution Image Visualization Tools

OpenSeadragon [4] excels in displaying high-resolution images, essential for visualizing microscope data.

Gap: Although OpenSeadragon provides excellent visualization capabilities, it lacks advanced image processing functions without installing extra plugins. Since the user can't install plugins on a public website, Boostlet.js fills this gap by providing a range of processing plugins that add-on functionalities like ROI selection, filters, and more.

2.3. Machine Learning and Advanced Processing Tools

Brainchop.org [7] represents a significant advancement in processing large-scale neuroimaging data using machine learning, offering advanced neural network models applied to brain imaging datasets on the client side.

Gap: Brainchop.org focuses on machine learning models for neuroimaging but does not provide a unified framework for integrating various processing tools across different web-based platforms. Boostlet.js complements this by providing an accessible framework that facilitates the application of such machine-learning techniques throughout multiple websites. This enables researchers to perform complex analyses with improved efficiency and ease without high-end equipment. Together, Boostlet.js and platforms like Brainchop.org enhance neuroimaging capabilities and ensure compatibility across different frameworks.

2.4. 3D Volume Visualization Tools

XTK [5] is a general-purpose toolkit for 3D visualization in the browser.

Gap: While XTK is versatile for 3D visualization, Boostlet.js compliments this framework by providing unified capabilities compatible with all other supported frameworks.

3. BOOSTLET.JS FRAMEWORK OVERVIEW AND EXAMPLES

The Boostlet.js library is designed to provide user-friendly image processing independent of a visualization framework. The library enables the development of processing modules as plugins. These plugins are called *Boostlets*. With a unified API, these plugins can access default functionalities such as pixel data access, filtering, segmenting, and data display. This allows developers to wrap their algorithm once and support a variety of visualization toolkits.

We provide a range of Boostlets as examples across several categories. For deep learning applications, we offer a Melanoma Predictor [8] that runs optimized deep learning models for melanoma detection on the web, and Segment Anything (SAM) [9] which enables users to segment areas of interest in any supported visualization framework. Our filtering capabilities include the Sobel Filter [10] for edge detection and Trako [11] for fiber track decompression in medical images. We also integrate Large Language Model (LLM) functionalities through Image Captioning [12] powered by Huggingface and WebLLM Chat for model interactions. We provide visualization tools for data analysis using Plotly.js to generate histograms and other data visualizations.

4. USAGE AND USER INTERFACE

One notable application of Boostlet is SliceDrop [13], a web-based medical imaging viewer that renders *PowerBoost* as a native integration. Researchers can drag and drop their medical imaging data directly into SliceDrop [13] and instantly access PowerBoost's functionality without additional setup.

This integration model extends to dataset platforms. For instance, researchers browsing datasets on OpenNeuro.org can similarly utilize Boostlet's capabilities through *PowerBoost* injection. In both scenarios, users can leverage tools like Meta's Segment Anything algorithm (SAM) [9] with ONNX.js [14] to segment regions of interest. After loading data in either platform, researchers can activate the Segment Anything Boostlet and draw bounding boxes around structures of interest, such as the corpus callosum. The segmentation is performed in real-time, with results immediately visible to the viewer. An example of this workflow can be seen in fig. 1, where PowerBoost has been activated within SliceDrop [13] to process voxel data with SAM on the client.

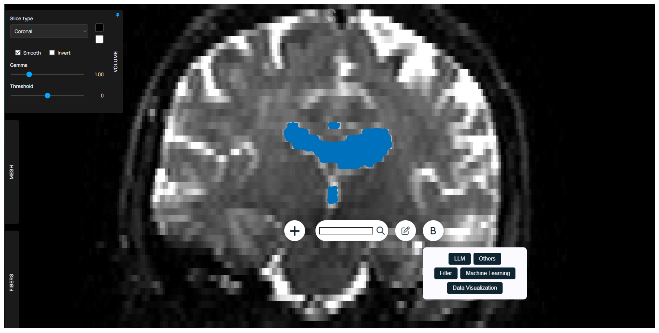

Fig. 1: The Segment Anything boostlet executed in SliceDrop [13] allows segmenting a region of interest in blue within a slice of MRI voxel data via box selection.

PowerBoost fig. 2 is the interface that bridges the framework and the user. The main features are a search bar for quick Boostlet recommendations and a dropdown menu for browsing various categories like Data Visualization, Filters, LLMs, and Machine Learning. The live code editor mode also offers a comprehensive panel for executing various API functions, including framework detection and HTTP POST requests, enabling users to create and test Boostlets in real time.

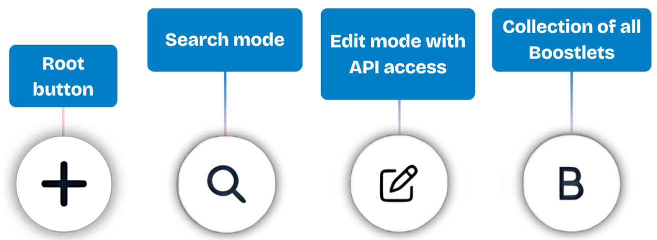

Fig. 2: PowerBoost is a versatile plugin that equips any frontend environment with a range of Boostlets and a code editor for API access. It is easily activated by dragging and dropping the bookmark from our main website into the bookmarks bar. It is also natively supported on the latest SliceDrop [13] viewer. After activation, PowerBoost injects code into the host website, revealing a floating icon. Clicking this icon expands a user interface with sections for Search, Edit, and Boostlets examples.

5. DEVELOPMENT AND INTEGRATION

To enhance accessibility for advanced image manipulation, we offer user-friendly functions that integrate various frameworks discussed in section 2.

The *Boostlet* superclass offers pixel data access functions during Boostlet execution. For example, *select_box()* lets the user select a rectangular region of interest (ROI) for manual segmentation or region-based analysis to then incorporate in processing (i.e., SAM). If a framework does not support a specific mode of interaction (such as OpenSeaDragon.js), the integrated BoxCraft.js [15] library implements custom widgets. Another interactive function example is *set_mask()*, which applies a mask to the image or canvas, which can be used to highlight, extract, or contrast different types of pixel data.

5.1. Framework Integration

Boostlets require a compatible visualization framework to access pixel data and modify the scene with native capabilities (i.e., ROI). If no compatible framework is present, Boostlets will access the **2D canvas fallback mechanism** for baseline image processing. This approach ensures that Boostlets' core functionalities remain operational, providing a consistent user experience across various web environments. The canvas element is the working area where the image data is rendered.

5.2. Development of Functionalities

To develop functionalities like segmentation and filtering, users can utilize the convenience functions provided by our API to interact with the image data from the host website. The high-level view of this process includes:

1. **Initialization**: The Boostlet library automatically detects the active visualization framework on the host website or triggers the fallback.
2. **Image Data Retrieval**: Using the *get_image()* function, users can fetch pixel data from the current image displayed on the website.
3. **Image Processing**: The *filter()* function, as an example, enables users to apply pixel transformations, such as convolution with a specified kernel.
4. **Displaying Results**: The *set_image()* function updates the canvas with the processed image data, reflecting the applied transformations in real time.

6. IMPLEMENTATION AND TESTING

6.1. Application: SliceDrop Reloaded

Boostlet is offered as default functionality on SliceDrop [13], which was first released in 2012 as a web-based viewer for medical imaging data, supporting visualization of volumes, meshes, and fiber tracts. The original version, built with X Toolkit (XTK) [5], received the Mozilla Dev Derby Award and was featured in VisWeek 2012.

SliceDrop Reloaded rebuilds this platform using the NiiVue [3] framework and includes PowerBoost as a native feature. Users can still drag and drop their neuroimaging data (volumes, meshes, and fiber bundles) directly into their web browser but now have access to all Boostlet functionalities by default. For instance, researchers working with brain MRI data can load their volumes or meshes, apply segmentation algorithms through PowerBoost, and immediately visualize the results in 2D and 3D views. Those working with fiber tracking data can load their tractography results and use Boostlets for filtering and analysis, all within the same interface. SliceDrop reloaded provides a solution for researchers who need to visualize and process their medical imaging data directly in the web browser.

6.2. Build and Testing

For Boostlet.js developers, we use a build system managed with Parcel, a fast, zero-configuration web app bundler [16] to compile the project into minified Javascript.

The testing process for Boostlet is designed in a multi-step pipeline (fig. 3) and focuses on validating the integrity of the image processing functionalities within the used framework during the development process fig. 3.

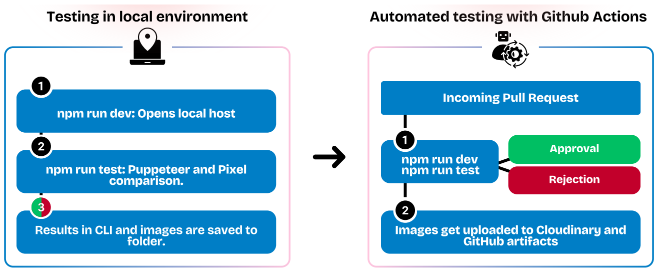
Testing in local environment
1
npm run dev: Opens local host
2
npm run test: Puppeteer and Pixel comparison.
3
Results in CLI and images are saved to folder.
Automated testing with Github Actions
Incoming Pull Request
1
npm run dev
npm run test
Approval
Rejection
2
Images get uploaded to Cloudinary and GitHub artifacts

Fig. 3: Pipeline for automated local and online testing.

7. DISCUSSION AND FUTURE WORK

We asked a group of medical imaging and visualization professionals from different institutions to provide feedback and evaluation by completing an 8-question survey (table 1). We received three anonymous responses; the results are in fig. 4.

No.	Question
1	I found Boostlet.js easy to integrate with existing web-based frameworks.
2	The process of adding Boostlet.js to my browser as a bookmark is straightforward.
3	The PowerBoost user interface is easy to navigate.
4	The client-side processing capabilities of Boostlet.js are satisfactory.
5	The API for pixel data manipulation provided by Boostlet.js is user-friendly.
6	Boostlet.js makes it easy to share or develop image processing plugins with the medical imaging community.
7	I am likely to recommend Boostlet.js to other members of the medical imaging community.
8	Please provide any additional comments or suggestions for improving Boostlet.

Table 1: List of questions given to an audience of three medical imaging and visualization professionals regarding their experience with Boostlet.

According to the survey results, Boostlet has a strong user experience and functionality foundation, as seen in responses to questions 1 through 7 (table 1, fig. 4). However, for question 8, respondents suggested that Boostlet could improve in making its processes more intuitive, such as allowing the drawing of multiple bounding boxes without refreshing and displaying helpful messages about ongoing processes. Another suggestion was to expand the model library to include a broader range of medical imaging tasks, which is now the current objective. Sharing Boostlet.js with the medical imaging and visualization community could help to achieve this goal.

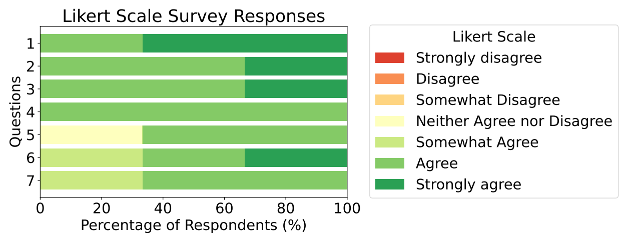

Fig. 4: Results of the Likert scale survey show that all users either stay neutral, agree, or strongly agree with the questions prompted in table 1. *Results for question 8 are not plotted since it is a short-answer feedback question.

7.1. Limitations and Future Work

Boostlet only supports a 2D canvas fallback when the framework is not detected as compatible section 5.1. However, some frameworks use WebGL or WebGPU. Thus, current efforts focus on a WebGL/WebGPU fallback mode allowing pixel manipulation.

Early experiments confirm that JavaScript-injected code can wrap around the WebGL context to monitor all rendering commands sent to the graphical processing unit (GPU). This code can then extract pixel data from the frame buffers before data is rendered. These pixels can then be processed using JavaScript code or in a fragment shader as part of a Boostlet plugin.

7.2. Security Concerns

There is the potential for Cross-Site Scripting (XSS) attacks if the host website does not follow best practices for request headers [17]. Additionally, JavaScript injection can be used to hijack WebGL contexts and intercept WebGL commands to modify 2D and 3D content using frame buffer extraction or texture mapping [18]. Content alteration can be forced without access to the server. This risk is inherent to any JavaScript injection process and is not unique to Boostlet.js. To mitigate this, host websites must implement robust security measures, such as setting appropriate Content Security Policy (CSP) headers, to prevent malicious exploits.

8. CONCLUSION

Boostlet.js enhances web-based visualization frameworks with medical image processing capabilities by injecting JavaScript code. This allows the development of a processing plugin (Boostlet) that works with a range of supported visualization frameworks without further tailoring. Boostlets are installable as bookmarklets or available within the bundled PowerBoost user interface.

9. ACKNOWLEDGEMENTS

The authors wish to thank Jenna Kim, Dhruv Shah, Adnan Ali, Sai Bharadwaj, Likhith Charugundla, Gopi Krishna, Neha Rapolu, Roshan Vemula and our anonymous evaluators for additional testing and improvements. This work was partly supported by grants from the Alfred P. Sloan Foundation, Oracle for undergraduate research, and the TRIO McNair program.

10. REFERENCES

[1] Christopher J Markiewicz, Krzysztof J Gorgolewski, Franklin Feingold, Ross Blair, Yaroslav O Halchenko, Eric Miller, Nell Hardcastle, Joe Wexler, Oscar Esteban, Mathias Goncalves, et al., "Openneuro: An open resource for sharing of neuroimaging data," *BioRxiv*, pp. 2021–06, 2021.

[2] Erik Ziegler, Trinity Urban, Danny Brown, James Petts, Steve D Pieper, Rob Lewis, Chris Hafey, and Gordon J Harris, "Open Health Imaging Foundation Viewer: An Extensible Open-Source Framework for Building Web-Based Imaging Applications to Support Cancer Research," .

[3] "WebGL2 based medical image viewer. Supports over 30 formats of volumes and meshes.," `https://github.com/niivue/niivue`.

[4] "An open-source, web-based viewer for zoomable images, implemented in pure JavaScript.," `https://github.com/openseadragon/openseadragon`.

[5] Daniel Haehn, Nicolas Rannou, Banu Ahtam, P. Ellen Grant, and Rudolph Pienaar, "Neuroimaging in the browser using the x toolkit," *Frontiers in Neuroinformatics*, 2012.

[6] Rii-Mango, "Rii-mango/papaya: A pure javascript medical research image viewer.," `https://github.com/rii-mango/Papaya`, 2012.

[7] Mohamed Masoud, Pratyush Reddy, Farfalla Hu, and Sergey Plis, "Brainchop: Next generation web-based neuroimaging application," 2023.

[8] SangHyuk Kim, Edward Gaibor, and Daniel Haehn, "Web-based melanoma detection," 2024.

[9] Yuhao Huang, Xin Yang, Lian Liu, Han Zhou, Ao Chang, Xinrui Zhou, Rusi Chen, Junxuan Yu, Jiongquan Chen, Chaoyu Chen, et al., "Segment anything model for medical images?," *Medical Image Analysis*, p. 103061, 2023.

[10] Chris Rorden, Roger Newman-Norlund, Chris Drake, Daniel R Glen, Julius Fridriksson, Taylor Hanayik, and Paul A Taylor, "Improving 3d edge detection for visual inspection of mri coregistration and alignment," *bioRxiv*, pp. 2022–09, 2022.

[11] Daniel Haehn, Loraine Franke, Fan Zhang, Suheyla Cetin Karayumak, Steve Pieper, Lauren O'Donnell, and Yogesh Rathi, "Trako: Efficient transmission of tractography data for visualization," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, Cham, 2020.

[12] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," 2022.

[13] Daniel Haehn, Nicolas Rannou, P. Ellen Grant, and Rudolph Pienaar, "Slice:drop – collaborative medical imaging in the browser," in *ACM SIGGRAPH Computer Animation Festival*. ACM, 2013.

[14] Microsoft, "Microsoft/onnxjs: Onnx.js: Run onnx models using javascript," `https://github.com/microsoft/onnxjs`, 2018.

[15] Shruti Varade, "Shrutivarade/boxcraft: Roi selection widget," `https://github.com/shrutivarade/BoxCraft`, 2023.

[16] Parcel-Bundler, "Parcel-bundler/parcel: The zero configuration build tool for the web.," `https://github.com/parcel-bundler/parcel`, 2018.

[17] "OWASP Secure Headers Project — OWASP Foundation — owasp.org," `https://owasp.org/www-project-secure-headers/`, [Accessed 31-10-2024].

[18] Tim Meyer, Gabi Dreo Rodosek, and Daniel Haehn, "Webgl-based image processing through javascript injection," in *ACM Conference on 3D Web Technology*, 2024, pp. 1–5.