Publications of the author

Total number of articles published by the author: 3
Cumulative impact factor: 3,268
Number of publications that form the basis of this thesis: 2 **(IF: 3,268)**

1. **A. Harmouche**, F. Kövér, S. Szukits, T. Dóczi, P. Bogner, A. Tóth, "XReport: An online structured reporting platform for radiologists," *SoftwareX*, vol. 17, p. 100993, Jan. 2022, doi: 10.1016/j.softx.2022.100993.
   Quartile: Q2, Impact factor: **2.868** (2023)

2. **A. Harmouche**, F. Kövér, S. Szukits, T. Dóczi, P. Bogner, A. Tóth, "WebMRI: Brain extraction and linear registration in the web browser," *Imaging*, vol. 15, no. 1, pp. 31–36, Jun. 2023, doi: 10.1556/1647.2023.00111.
   Quartile: Q4, Impact factor: **0.4** (2023)

Other publications

G. Jandó, E. Mikó-Baráth, A. Czigler, **A. Harmouche**, I. Szabó, L. Závor, and D. P. Piñero, "Amblyopia screening with the dynamic random dot stereotest," *Ophthalmology Times Europe*, vol. 16, no. 7, Sept. 2022

Acknowledgements

I would like to express my sincere gratitude and thanks to my program leader Prof. Dr. Péter Bogner for his support and guidance throughout the years leading up to this PhD thesis.

I am greatly thankful to my supervisor Dr. Arnold Tóth for his support and technical excellence. He helped conceptualize WebMRI, and thanks to his experience in using the FSL tools he gave me valuable feedback and directions during the development and testing of our software.

I am grateful to Dr. Ferenc Kövér for his involvement in the XReport development. He helped popularize XReport among his colleagues at the Diagnostic Center of Pécs, and also created multiple templates in the program. His feedback and guidance was really valuable to shape and fine-tune XReport.

I am thankful to Dr. Sándor Szukits for his involvement in testing the template creation and reporting workflow of XReport. His experience as a user with other structured reporting platforms was beneficial in building our software.

I am thankful to all the radiologists at the PTE ÁOK Department of Medical Imaging, and the Diagnostic Center of Pécs, who have been using XReport. Their feedback helps make XReport better and more reliable.

At last but not least I would like to thank my wife Margaréta and my sons Ádám, Sebestyén and Bendegúz, for their love and support throughout these challenging years. I am grateful to my whole family who encouraged me to pursuit this research, and helped me through the hard times.

Summary of the new scientific results

1, We developed WebMRI, a web-based, modular neuroimaging platform. We ported the FSL BET and FLIRT (brain extraction and linear registration) image processing algorithms to WebAssembly so that they can be run in a browser environment. To the best of our knowledge, no other web port of these FSL tools exist. We compared the runtimes of the native and ported FSL tools, and found that in everyday use, our versions are not significantly slower than the native programs. We added support for DICOM loading in WebMRI, thus eliminating the need for an external DICOM to NIfTI conversion step. We developed a plugin system, which allows other developers to create new algorithms, or port existing ones, and bring them into the WebMRI platform.

2, We developed XReport, a free and open-source, web-based structured reporting platform for radiologists, which supports both template creation and reporting in a user-friendly manner. We developed an LLM-based solution for automatic structured reporting template filling from free text report, using prompt-engineering techniques.

Our experiment with GPT4 demonstrates that the language processing capabilities of LLMs are highly advanced and could bridge the gap between free-text reporting and structured reporting.

Discussion

WebMRI

With WebMRI, we have created an extensible, platform-independent, open-source neurological image processing platform that supports brain extraction and linear registration, and can be run in any modern web browser without external plugins or a web server. For brain extraction, we used the BET, and for linear registration, we used the FLIRT FSL software package. With the help of Emscripten, we ported the native programs to WebAssembly. We developed a new plugin system in BrainBrowse that allows us to run our ported tools. We also created a demo application that, using WebMRI, allows for a complete neurological image processing workflow, from loading DICOM slices to linear registration.
With our software system, we demonstrated that not only neurological visualization but also image processing is possible in modern browsers. In contrast to Slicer 3D or other nonbrowser-based programs, our solution does not require installation.
One limitation of our software is the decreased performance in browsers due to their limited resources compared to native execution. In the future, we plan to port more neurological image processing algorithms and aim to simplify the integration of our system into clinical PACS environments.

XReport

With XReport, we have developed a free, platform-independent, open-source structured reporting software system for radiologists. Our software allows users to easily create new report templates and to generate reports. In the design of our program, we emphasized modularity, so we organized the core of the system into a separate library, thanks to which the reporting and template creation functions can easily be integrated into any other web application. In addition, we created a simple DSL called FormScript for the dynamic operation of templates. Our responsive templates can easily be integrated into any HIS system. Compared to the free service developed by RSNA, our solution is simpler and more user-friendly, mainly due to dynamic scripting capability and the responsive user interface. Our solution is open-source, so it can be easily developed further by anyone, unlike RadReport, which is proprietary. The number of templates available in XReport is negligible compared to the RadReport library, but this could change as our platform gains traction.

Code snippets associated with the templates in FormScript can be edited in a text box, which can be accessed by clicking on the gear icon in the upper right corner of the report.

LLM-based structured reporting

Based on the prompt presented in the methods section, GPT-4 generated the following structured report:


```
{
"morphology": ["polypoid", "ulcerating", "circumferential", "semicircumferential"],
"selectedMorphology": "polypoid",
"appearance": ["mucinous", "non-mucinous"],
"selectedAppearance": "mucinous",
"lengthOfTumor": { "value": 50, "unit": "mm" },
"sphincterInvolvement": ["none", "internal sphincter is involced", "intersphincteric space is involved",
"external sphincter is involved"],
"selectedSphincterInvolvement": "external sphincter is involved",
"distanceTumorMesorectalFascia": { "value": 3, "unit": "mm" },
"lymphNodeInvolvement": true,
"numEnlargedLymphNodes": 5,
"numLymphNodesLessThan5mm": 2,
"numLymphNodesGreaterThan9mm": 3,
"tumorDepositInsideMesorectalFascia": true,
"numTumorDeposites": 2
}
```


It can be observed that the information extraction occurred based on the JSON structure provided in the prompt. The correct values were selected from the appropriate lists into the "selectedMorphology" and "selectedSphincterInvolvement" fields. The extraction of the unit of measurement was also successful for the "distanceTumorMesorectalFascia" and the "lengthOfTumor" fields. The LLM correctly executed the extraction of data related to the number and size of lymph nodes and other lesions, which is clearly visible in the examples of "numEnlargedLymphNodes", "numLymphNodesLessThan5mm", "numLymphNodesGreaterThan9mm", and "numTumorDeposites".

XReport

We have developed the XReport application, the source code of which we have released on Github under the Massachusetts Institute of Technology (MIT) license. The template creator and reporting module of the program can be seen in Figure 5. At the top is the name of the template, and below that is the content of the template itself. To the right of the template, 4 buttons appear: the top button with a document icon brings up the view in which the software displays the generated report. The second icon from the top copies the content of the report to the clipboard, making it easy to paste it into the text boxes of other software. The white plus sign on a green background starts a new report, and by clicking on the bottom icon, the template can be shared (it is the link related to the template, not the generated report, that is copied to the clipboard).

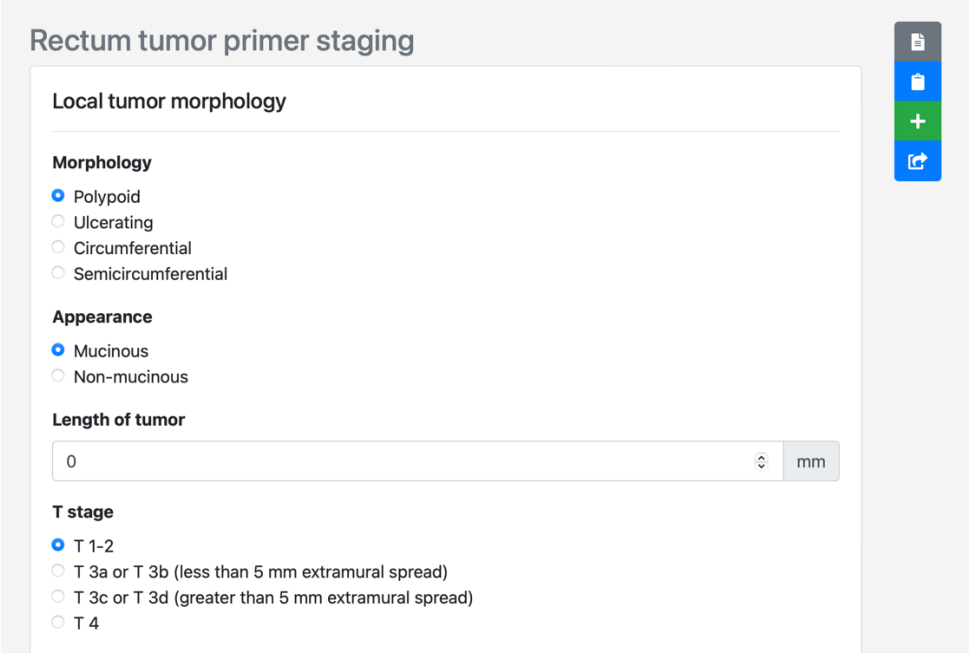

Figure 5. – A rectal tumor primer staging report template displayed on the XReport reporting interface.

In template creation mode, some elements of the reporting view change. Instead of the 4 buttons on the right-hand side, only 2 will be visible: save template and discard template. In addition, at the bottom center of the template, the program displays a plus button, by clicking on which another row can be added to the template. By hovering the cursor over the rows, there is an option to duplicate or delete the row, and by selecting the individual fields, the editing options for that item will appear.

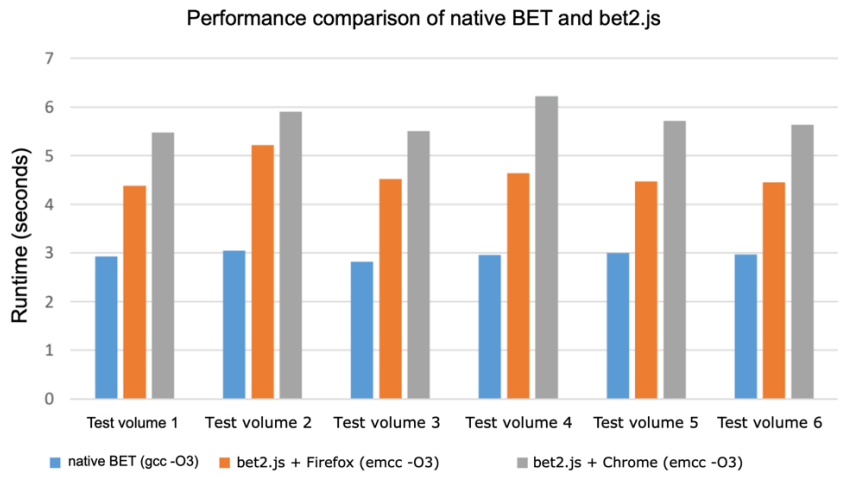

Figure 3. – Comparison of the runtime of the native BET and the bet2.js programs. The vertical axis shows the runtime expressed in seconds. The 6 column groups represent the processing times for the 6 test volumes: the blue column represents the native, the orange represents the time measured in Firefox, and the gray represents the runtime in Google Chrome.

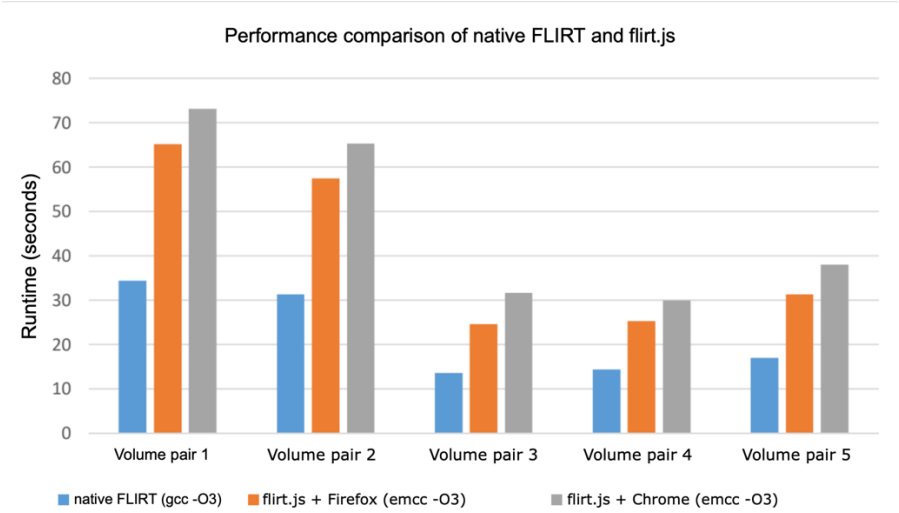

Figure 4. – Comparison of the runtime of the native FLIRT and the flirt.js programs. The vertical axis displays the runtime expressed in seconds. The 5 column groups represent the processing times for the 5 volume pairs: the blue column represents the native, the orange represents the time measured in Firefox, and the gray represents the runtime in Google Chrome.

Comparison of the runtimes of the ported plugins and the native versions

We compared the performance of the WebAssembly-ported bet2.js and flirt.js to the native FSL algorithms in terms of runtime. Both programs were compiled at the highest -O3 optimization level. The native programs were run on the Windows 10 Linux subsystem, while the ported versions were executed in Google Chrome and Mozilla Firefox browsers. The specifications of the computer used for testing were:

- CPU: Intel® Core™ i5-3230 @ 2.60Ghz
- RAM: 6.00 GB
- System Type: 64-bit operating system, x64-based processor

We ran bet2.js and BET on 6 MR volumes in such a way that each was processed 5 times by both the native and ported software, thus eliminating the variance due to a cold start. The average runtime for the native BET was 2.96 seconds. The bet2.js took an average of 5.75 seconds in Google Chrome and 4.62 seconds in Mozilla Firefox to complete the processing (Figure 3).

For the comparison between flirt.js and the native FLIRT, we used 10 MR volumes, which formed 5 pairs. For each volume pair, one element was a T1-weighted reference, and the other was an SWI volume, which the algorithm registered to the reference. The T1-SWI pairs always came from the same subject. The processed files were in NIfTI format, and as a preprocessing step, we applied brain extraction to them. To eliminate the cold start variance, we proceeded in the manner described earlier.
The native FLIRT took an average of 22.12 seconds to complete the registration. The WebAssembly version, when run in Firefox, took an average of 40.79 seconds, while in Chrome, it took 47.64 seconds (Figure 4).

Results

WebMRI

The start screen of our WebMRI demo application (Figure 2) displays the loaded volume on the right in sagittal, coronal, and axial sections, while on the left the list view of the loaded volumes and those generated by the plugins can be seen. Below the list view, GUI elements appear with sliders and inputs that allow for windowing of the image and other modifications. In the menu bar above, three menu items are visible: "File", "Tools", and "About". Through the "File" menu, users can browse and upload the files they want to process. In the "Tools" menu, plugins loaded into the program can be accessed. By clicking on the "About" menu item, users can visit a webpage providing information about the usage of the software and potential further development opportunities.

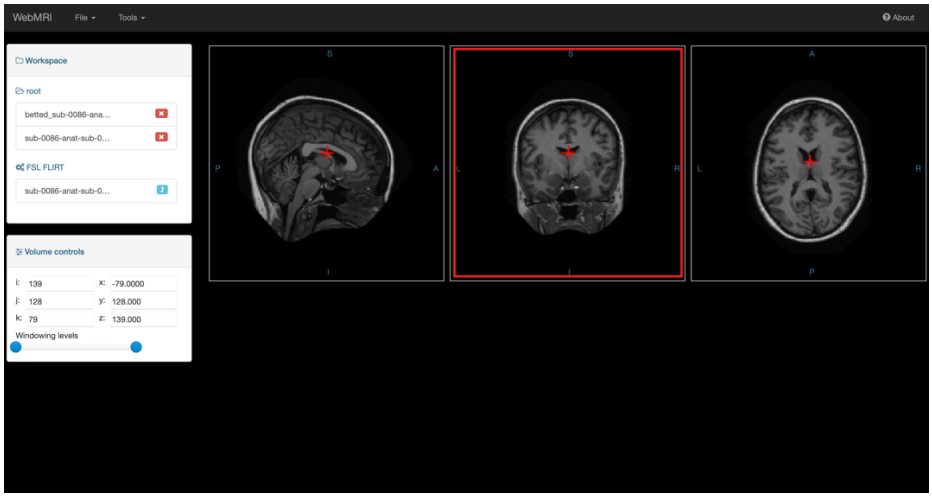

Figure 2. – The image displays the user interface of WebMRI. At the top, the menu bar is shown. Below the menu bar on the left is a window labeled "Workspace", which displays the files loaded and those generated by the plugins. Below that, under "Volume controls", elements that allow for manipulation and navigation of the loaded image are visible. On the right, the loaded volume is shown in sagittal, coronal, and axial sections.

LLM-based structured reporting

During our experiment with GPT-4 LLM, we performed the automatic structuring of a free-text report segment by providing the following prompt:

We will provide a free text radiological report, and a structured radiological report template. Convert the free text format to the template format.

Free text:
A polypoid, mucinous mass is visible 3mm from the mesorectal fascia. The mass infiltrates the external sphincter. The length of the mass is 50mm. There are 5 enlarged lymph nodes, 2 of them less then 5 mm in size, and 3 of them larger than 9mm. There are 2 tumor deposits inside the mesorectal fascia.

Template:
{
„morphology": [„polypoid", „ulcerating", „circumferential", „semicircumferential"],
„selectedMorphology": „",
„appearance": [„mucinous", „non-mucinous"], „selectedAppearance": „",
„lengthOfTumor": { „value": „", „unit": „mm" },
„sphincterInvolvement": [„none", „internal sphincter is involced", „intersphincteric space is involved", „external sphincter is involved"], „selectedSphincterInvolvement": „",
„distanceTumorMesorectalFascia": { „value": „", „unit": „mm" }
„lymphNodeInvolvemnet": False,
„numEnlargedLymphNodes": 0,
„numLymphNodesLessThan5mm": 0,
„numLymphNodesGreaterThan9mm": 0,
„tumorDepositInsideMesorectalFascia": False,
„numTumorDeposites: 0
}

XReport

During the development of the XReport web application, we used two programming languages: JavaScript and TypeScript. The project can be divided into two parts: the library and the application. The library is a standalone, reusable JavaScript software package (which can be integrated into other programs) that encompasses the main functionalities required for template creation and reporting. The application is a Single Page Application (SPA) that, utilizing the library, displays the templates, the template builder, and the reporting interfaces.

The library

The entry point of the XReport library is the "makeWidget" function call, which, without specifying a Uniform Resource Locator (URL), allows for the creation of a new template, and with a URL, allows for the loading of an existing template. Specific Document Object Model (DOM) elements or compositions of DOM elements can be used in the library to create the template.

We created a domain-specific language called FormScript to support the dynamic behavior of our templates in such a way that they are protected from Cross Site Scripting (XSS) attacks. If we allowed users to arbitrarily insert JavaScript code into the templates, it would pose a security risk. FormScript ensures that only the implemented operations can be executed in the templates, and nothing else. Based on the commands and conditions described in the FormScript code assigned to the template, the disappearance, appearance, filling, and other properties of the fields can be automated.

The application

We developed the XReport application using the Angular SPA and the Bootstrap Cascading Style Sheets (CSS) frameworks, and it is written in a mix of JavaScript and TypeScript languages. For hosting the site, we used the free web hosting service of Google Firebase.

Porting FSL BET and FLIRT to WebAssembly

FSL is a modular software package in which BET and FLIRT are defined as separate command-line programs. FSL is mainly written in C and C++ to maximize performance. To port these components, we had to modify parts of the code as well as the scripts (Makefile) that compile the programs. We used a compiler called Emscripten for automating the porting, which is based on the Low Level Virtual Machine (LLVM) infrastructure and is capable of translating LLVM bitcode to WebAssembly. Emscripten greatly simplified the porting process since only minimal modifications were required to the original C/C++ codebase. To ensure the ported programs did not block user interface responsiveness, they were run on background threads using Web Workers.

DICOM support

While BrainBrowser allows for the loading of various neurological file formats (NIfTI, MINC, MGH), it does not support DICOM. To load DICOM slices, they must first be converted to a format supported by BrainBrowser using an external program. BrainBrowser was extended with a DICOM volume loader, which operates with the help of a C++ based software named dcm2niix (https://github.com/rordenlab/dcm2niix) that was also ported to WebAssembly using Emscripten.

Plugin system

We extended BrainBrowser with a plugin system, making it easy to integrate our ported software (BET, FLIRT, DICOM volume loader). Each plugin is described by a JavaScript Object Notation (JSON) file. This file defines the parameters supported by the plugin, their type, and name. From this, the WebMRI GUI generating system renders the user interface of the plugin and runs the program registered to the plugin.

Materials and methods

WebMRI

The development of WebMRI can be divided into three main parts:

- Porting FSL BET and FLIRT to WebAssembly.
- Modifying BrainBrowser and extending it with a plugin system to run the ported tools within BrainBrowser (Figure 1).
- Developing a demo web application that utilizes WebMRI, allowing for brain extraction and linear registration to be executed within the browser via a simple user interface.

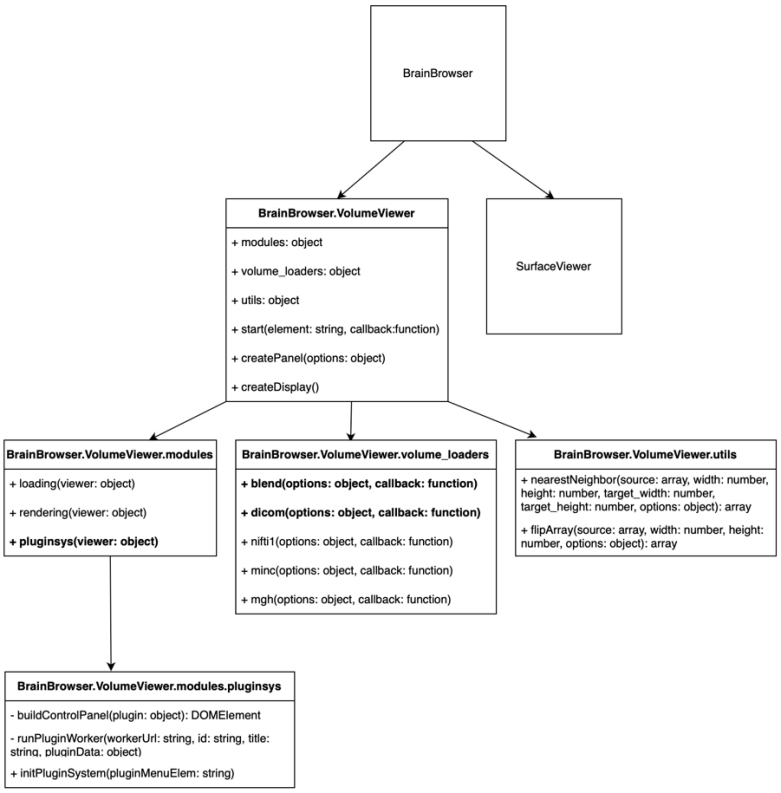

Figure 1. – The figure illustrates the software architecture of BrainBrowser, with components that were introduced during the development of WebMRI highlighted in bold ("blend": allows for the blending of two volumes, "dicom": assists in loading DICOM volumes, "pluginsys": entry point for the plugin system).

Aims

The primary aim of this paper is to examine the applicability of web-based medical software systems in the domains of medical image processing and radiological reporting. Our goal is to create a fully browser-based neurological image analysis software, WebMRI, built upon the open-source BrainBrowser. By combining the volumetric visualization capabilities of BrainBrowser with the image processing algorithms of FSL, we aim to establish a platform capable of performing brain extraction and linear registration without the need for a dedicated web server or plugins. This is made possible by porting the FSL BET and FLIRT tools to WebAssembly. The ported software versions are then compared to the native FSL programs.

We also develop another web-based medical software, XReport, which is an open-source, freely usable structured reporting platform. We compare our software to another freely available reporting platform, the RSNA RadReport.

With our web-based medical software solutions, we aim to promote browser-based neuroimage processing and structured reporting, as well as advocate for open-source medical software. The source code for both of our programs will be made available on Github.

The characteristics of large language models, as described above, could also be utilized in radiological reporting.

available for free. Even trying them out requires submitting a request through a form, and only after the request is approved can the applications be tested in a trial mode.

Artificial neural networks in radiology

As mentioned in the first section of the introduction, in addition to the web, another major technological breakthrough in radiology has come from artificial intelligence, specifically artificial neural networks. The rapid hardware and software advancements in GPUs and Tensor Processing Units (TPU) now allow for the training and running of neural networks with tens of billions of parameters. This means that current AI models can learn incredibly complex relationships, be it from two-dimensional, three-dimensional datasets, or even free text.

- Convolutional Neural Networks: Convolutional Neural Networks (CNN) are models whose primary operation is convolution. Convolution can be interpreted as applying various filters to the input data. By using these filters, input data can be simplified: the model can identify edges and corners and, once this filtered image is passed to later layers of the model, these layers can recognize shapes and objects. CNNs are often used for various segmentation tasks (e.g., fully outlining organs in a CT scan) and object recognition tasks (e.g., identifying pneumonia in an X-ray) in medical imaging and image processing.

- Large Language Models: Large Language Models (LLM) are neural networks with a large number of parameters (sometimes in the hundreds of billions) that can be effectively trained for linguistic tasks such as translation, text generation, conversation, information extraction, and summarizing longer texts. A characteristic feature is their "few-shot learning" capability, meaning they only need to see a few examples of a task type to generalize and learn that specific task. Due to their vast number of parameters, training LLMs is highly energy-intensive and expensive. The highly popular GPT4 LLM, developed by OpenAI, reportedly cost close to 100 million dollars to train. Due to these high costs, LLMs are typically trained once, and users can fine-tune them for various linguistic tasks using "prompts." Open-source LLMs are also available, such as LLaMA developed by Meta, which has 65 billion parameters.

Radiological reporting

Radiological reporting is essential in the medical imaging process. Through the radiological report, the radiologist responds to the questions of the referring clinician, thus playing a vital intermediary role in communication between the clinician and the radiologist. The information conveyed in the report should be accurate, concise, and clear. Another important aspect is the amount of time a radiologist takes to prepare the report, i.e., how quickly the clinician gets an answer to their question, and how rapidly the patient receives the appropriate diagnosis and, ultimately, the treatment.
There are two main approaches to radiological reporting:

- **Dictation-based reporting:** With this method, the radiologist describes the abnormalities seen on the image using a microphone. Analyzing software converts the sound waves into free text in real-time and inputs it into a text box, which the doctor can then edit and format. While this method is efficient and flexible, it may result in significant variability in report content and format due to different wordings, phrasings, and reporting styles.

- **Template-based structured reporting:** With this approach, radiologists have to fill out a preselected template, similar to filling out a Google Forms questionnaire. Structured reports enhance consistency at the expense of flexibility.

Web-based software for structured reporting

There are several software solutions available for structured reporting, of which perhaps the most well-known is the RadReport web platform developed by the Radiological Society of North America (RSNA). Essentially, it is a vast template database where users can submit structured reporting templates, which can be created on the site using the T-Rex Template web application. Templates can be filtered based on modality, date, and various other parameters. Although the program (both the template browser and the template builder) can be used for free, third-party modifications are not possible since the software is not open source.
Several companies have developed structured reporting solutions (Smart Reporting: https://www.smart-reporting.com, RadioReport: https://radioreport.com), but these are not

programming knowledge, so its clinical use is not typical, but it is extensively used in research.

Another noteworthy open-source neuroimage processing application is Slicer3D. While FSL is primarily command-line based, Slicer3D focuses on a graphical user interface (GUI) based useage, making it easy to use even without programming experience or deep expertise in informatics. Slicer3D also supports the previously mentioned linear registration and brain extraction, although not natively, but through third-party extensions.

Web software in neuroimage processing

As mentioned earlier, thanks to technologies like HTML5, WebGL, and WebAssembly, it is possible to run computationally intensive algorithms in browsers.

HTML5 is the latest version of the HyperText Markup Language (HTML), introducing innovative technologies detailed below into the world of browsers.

WebGL is a graphics library that allows browsers to utilize the hardware acceleration of the Graphics Processing Unit (GPU) for displaying two or three-dimensional images. This significantly reduces the rendering time of an image compared to the software based drawing time of the Central Processing Unit (CPU). This efficiency arises because, while the CPU can draw only one pixel at a time, the GPU can perform this in parallel, allowing for the processing of potentially millions of pixels simultaneously.

WebAssembly is a binary format containing machine instructions that a virtual machine running in a browser can execute. Unlike JavaScript, the "native" programming language of the browser, which is dynamically typed, WebAssembly is statically typed. This means the type correctness of the program is checked during compilation, paving the way for various optimization opportunities that result in faster execution times.

Neurological data sizes often span gigabytes, and their processing has high computational requirements.

BrainBrowser is a browser-based software that employs the technologies mentioned above for the two-dimensional slice-by-slice and three-dimensional surface visualization of volumetric neurological data. The software is open-source, and both its modular architecture and licensing (GNU Affero General Public License v3.0) allow for modifications and extensions.

Foundations of medical image processing

Medical image processing encompasses all techniques aimed at improving, analyzing, and processing images. The communication between medical imaging devices and the RIS is done according to the Digital Imaging and Communications In Medicine (DICOM) standard. DICOM defines both a communication protocol and a file format. The images generated by different modalities are usually saved in DICOM format (typically with a ".dcm" extension) in the database of the device, and they are also forwarded in this format to the next DICOM node. Modalities that produce spatially coherent images (like CT, MRI) generate, save, and forward images in so-called slices. The advantage of this is that the evaluation of the image can begin without loading the entire volume, as the slices are individually addressable and downloadable. However, the downside is that analyzing and processing these images becomes more complicated. From the perspective of processing volumetric images, a file format that describes the entire volume in a single file and treats the image as a three-dimensional data set is more advantageous.

Neuroimage processing

Neuroimage processing is a subset of medical image processing that focuses on visualizing and analyzing the structures and functions of the brain and the central nervous system. MRI is especially important in this field as it can produce high-resolution images of the brain without exposing patients to ionizing radiation. While DICOM is considered a gold standard file format and specification across almost all areas of medical imaging, the Neuroimaging Informatics Technology Initiative (NIfTI) format, as a successor to Analyze, was specifically developed for neuroscientific use. The goal was to create a format that facilitates the analysis and processing of neurological volumetric data, bridging the limitations of DICOM from an image analysis perspective.

Two crucial algorithms in neuroimage processing are brain extraction and linear registration, most often performed using programs from the FMRIB Software Library (FSL). FSL is an open-source software package, with its tools covering the entire spectrum of neurological image analysis. The FMRIB's Linear Image Registration Tool (FLIRT) is used for linear registration, and the Brain Extraction Tool (BET) is used for brain extraction. Although the software package is available for Windows, Linux, and MacOS platforms, it is predominantly used in a Linux environment. Its installation and use are not trivial without at least basic

Introduction

The digital advancements of the 21st century have modernized medicine and enabled significant technological progress. The two perhaps most significant achievements of the century are the web and artificial intelligence (AI), both of which quickly revolutionized medicine. Radiology stands out, in particular, as it is a field where any technology that aids in the interpretation of medical images (X-ray, CT, MRI, etc.) can quickly become beneficial. Hospital Information Systems (HIS) and Radiology Information Systems (RIS) are mostly web-based, even if not all of their components are. Initially, browser-based solutions appeared in those parts of radiological workflows that did not require high computational capacity (mainly administrative tasks). However, with the evolution of the web, especially with the emergence of the HyperText Markup Language 5 (HTML5), the Web Graphics Library (WebGL), and WebAssembly, running computationally intensive image processing and image analysis algorithms in browsers became possible. Since web browsers are available on almost every digital platform (desktops, smartphones, tablets), web programs are platform-independent. Besides being platform-independent, browser applications do not require installation or maintenance, as every dependency and prerequisite for running the program is downloaded upon visiting the website. Manual updates for the applications are also unnecessary in this manner, as the new version of the program becomes available immediately upon opening the page. These characteristics of web applications have greatly contributed to their conquest in medical informatics. In addition to the previously mentioned radiological use cases, web-based solutions have also been developed for reporting, specifically for structured, template-based reporting.

In the following, we would like to introduce the informatics aspects, specifically the web-related aspects, of two essential radiological workflows: image processing and reporting.

List of abbreviations

AI – Artificial Intelligence

BET – Brain Extraction Tool

CSS – Cascading Style Sheets

CT – Computerized Tomography

DICOM – Digital Imaging and Communications in Medicine

DOM – Document Object Model

FLIRT – FMRIB's Linear Image Registration Tool

FSL – FMRIB Software Library

GUI – Graphical User Interface

HIS – Hospital Information System

HTML – HyperText Markup Language

HTML5 – HyperText Markup Language 5

LLM – Large Language Model

LLVM – Low Level Virtual Machine

MGH – Massachusetts General Hospital

MINC – Medical Imaging NetCDF

MIT – Massachusetts Institute of Technology

MRI – Magnetic Resonance Imaging

NIfTI – Neuroimaging Informatics Technology Initiative

RIS – Radiology Information System

SPA – Single Page Application

TPU – Tensor Processing Unit

URL – Uniform Resource Locator

WebGL – Web Graphics Library

XSS – Cross Site Scripting

Developing Browser-Based Software Systems for Medical Image Processing and Radiological Reporting

PhD thesis

Dr. Ahmed Mouhamad Harmouche

Clinical Neurosciences Doctoral School (D221)

Imaging in neuroscience (B-3/2014)

Leader of the Doctoral School: Prof. Dr. Sámuel Komoly

Program leader: Prof. Dr. Péter Bogner

Supervisor: Dr. Arnold Tóth

University of Pécs Medical School

Department of Medical Imaging

Pécs, 2023