Cloud-based Privacy-preserving Medical Imaging System using Machine Learning Tools

João Alves*, Beatriz Soares*, Cláudia Brito, and António Sousa

INESC TEC & Universidade do Minho, Braga, Portugal

Abstract. Healthcare environments are generating a deluge of sensitive data. Nonetheless, dealing with large amounts of data is an expensive task, and current solutions resort to the cloud environment. Additionally, the intersection of the cloud environment and healthcare data opens new challenges regarding data privacy.
With this in mind, we propose MedCloudCare (MCC), a healthcare application offering medical image viewing and processing tools while integrating cloud computing and AI. Moreover, MCC provides security and privacy features, scalability and high availability. The system is intended for two user groups: health professionals and researchers. The former can remotely view, process and share medical imaging information in the DICOM format. Also, it can use pre-trained Machine Learning (ML) models to aid the analysis of medical images. The latter can remotely add, share, and deploy ML models to perform inference on DICOM images.
MCC incorporates a DICOM web viewer enabling users to view and process DICOM studies, which they can also upload and store. Regarding the security and privacy of the data, all sensitive information is encrypted at rest and in transit. Furthermore, MCC is intended for cloud environments. Thus, the system is deployed using Kubernetes, increasing the efficiency, availability and scalability of the ML inference process.

Keywords: Healthcare Application · DICOM Images · Cloud Computing · Machine Learning

1 Introduction

The vast amount of information created and ingested in clinical environments [17] makes the analysis and collection of data with labelled ground-truth a boundless challenge [26]. Thus, the development of technological tools that can assist medical professionals in accessing, processing, and interpreting that data in a timely and accurate manner is a significant concern.

Nowadays, medical images are extensively utilised to diagnose, plan, and guide the treatment and monitoring of disease progression [23]. In this context, systems that enable clinicians to remotely access and evaluate patients' medical imaging information have been increasingly sought-after. Recently, radiologists have collaborated with data scientists to develop web applications for radiological

* These authors contributed equally to this work

purposes [19,26]. Web applications allow radiologists to remotely view, share and interpret images within a browser and without additional software installed on their machines [18,19,26]. Moreover, cloud computing is becoming an attractive computing model for biomedical research. Hospitals and researchers are shifting to cloud environments to facilitate large-scale data analysis and remote sharing, and web applications are a practical way to interact with such environments [28]. For these reasons, web-based systems have been increasingly valuable for clinicians and researchers. Another fundamental element of the medical imaging field is the Digital Imaging and Communications in Medicine (DICOM) standard. It allows clinicians to view, store, and share medical images independently of their location or the devices they use and is considered the primary standard for image data management in healthcare [22].

There are still some technological challenges regarding medical image analysis to be addressed. Firstly, many users have sensitive clinical data that must be safely stored and retrieved. Therefore, data should be encrypted both at rest and in transit. Secondly, many of the methodological tasks (image registration, localisation, classification, detection, segmentation) involved in a medical image analysis process often encompass manual workflows that can be tedious, prone to observer variation and, most crucially, time-consuming. Artificial Intelligence (AI) is frequently required to enhance these tasks. Constant improvements in AI are helping to identify, classify, and quantify patterns in medical images. For these reasons, some web applications allow the deployment of Machine Learning (ML) or Deep Learning (DL) models to help diagnose clinically relevant results [25]. Current solutions cannot solve such challenges effectively, either by not offering privacy features or not being scalable and high available or not allowing easy integration of new AI algorithms.

To tackle this, we propose MedCloudCare, a web-based healthcare application. MCC integrates typical DICOM viewer features (*e.g.,* rotation, pan, annotation) with state-of-the-art pre-trained AI models. Due to the large volumes of data, MCC is intended to be deployed in cloud environments. Nonetheless, it also can be deployed locally. Additionally, the system is built for health professionals and researchers. First, it provides a visually appealing interface for health professionals to view and analyse patients' data. Secondly, researchers can add, store, deploy and test their pre-trained AI models on DICOM data.

The outline of the paper is as follows. Section 2 reviews some state-of-the-art medical imaging applications. Section 3 describes the designed system architecture, while Section 4 presents some obtained results. Finally, Section 5 outlines the main conclusions and the work to be done.

2 Related Work

This section reviews some state-of-the-art medical imaging applications. ePAD [7] is a platform for visualisation, annotation, and quantitative analysis of medical images. Another extensible research platform is 3D Slicer [1], a desktop application requiring local installation, which is massively adopted for imaging

research. RayPlus [27] is a web application for medical image processing developed by Yuan et al. MedCloudCare distinguishes itself from these applications by offering a cloud-based alternative with security and privacy guarantees.

In the scope of web applications offering AI capabilities, Tesseract-MI supports the deployment of AI models while providing image viewing and reporting [24]. However, Tesseract-MI only supports the deployment of static and predefined AI models and requires a connection to a DICOM server to view images since it does not allow users to upload DICOM files in the application. Mehrtash et al. developed DeepInfer [3] as an extension of 3D Slicer. It uses Docker to enable users to run different DL models on their data on a local machine. Similarly, the TOMAAT framework allows users to serve their DL applications over the cloud [15]. Nevertheless, DeepInfer and TOMAAT applications require a client with a specific interface to connect with the server to deploy the DL models.

Distinctively, MCC allows users to upload DICOM files in the application and add and deploy their own AI models, which they can select and apply to those images. Furthermore, it has a visually appealing user interface (UI) while managing multi-users with distinct roles and permissions and having privacy features, scalability and high availability. Moreover, MCC intends to offer straightforward integration of AI models in the medical image analysis workflow, requiring minimal software installation to assure compatibility with clinical standards.

3 MedCloudCare

The proposed solution intends to provide four main features: user authentication; DICOM image viewing, processing and storage; sharing of imaging data and models between users; and addition and deployment of pre-trained models.

This section describes MedCloudCare's proof-of-concept (PoC) architecture (Figure 1). The latter follows the client-server model. The components executed on the server-side create the backend (Django API, PostgreSQL database, Orthanc server and Kubernetes cluster), and the elements executed on the client-side provide the UI, therefore, encompass the frontend (e.g., the OHIF viewer).

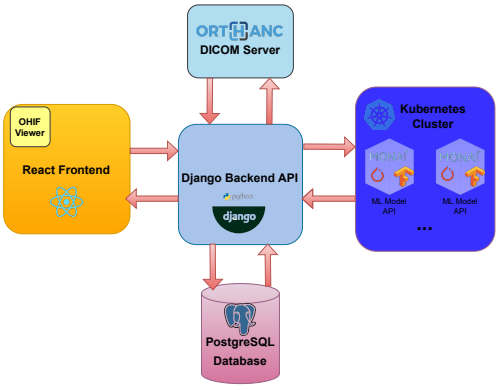

Fig. 1: MedCloudCare Architecture Components.

3.1 React Frontend

The client-side of MCC was developed using the JavaScript programming language, particularly with the React web framework [14]. It is executed in a browser, allowing users to access the backend services by sending API (Application Programming Interface) requests. Since the application is intended for medical and research use, the UI offers different functionalities according to the type of user. Nevertheless, when the application starts, the UI is common to all users, showing an authentication page where they can create an account and sign in. The authentication process can be performed using the users' email or social networks (namely, Google, Facebook, Linkedin, GitHub and Spotify). Once the users are successfully authenticated, they have access to the following features:

- *Image Storage with Orthanc*: the Orthanc DICOM server is currently considered the ubitiquous open-source solution for DICOM image data storage. One of its main strengths lies in its built-in REST API. Such an API gives full programmatic access to all core features of Orthanc, namely, the capability to upload, transfer and retrieve images [10];
- *Image Viewing and Processing with OHIF*: the Open Health Imaging Foundation (OHIF) viewer addresses the demand for open-source web imaging applications [28]. It is based on web technologies, including JavaScript, React [14], and the Cornerstone.js library [2], and can be used as a Single Page Application embedded into third-party applications. It is standards-conforming and relies on DICOMweb [4] for data exchange and connectivity to image archives, e.g., Orthanc. The UI components of the viewer are provided in an independent React library so that developers can customise the UI or use its components in their applications. In reality, the OHIF Viewer has been adopted in various clinical research platforms (e.g., Precision Imaging Metrics [13], XNAT [16]) and commercial applications (e.g., OsiriX [11]) [28]. With this in mind, MCC incorporated a customised version of OHIF to provide DICOM image viewing and processing capabilities;
- *Addition and Deployment of AI models*: MCC provides an interface component that enables the application to use pre-trained, out-of-the-box AI models. To that matter, users need to add models to the application. First, they must fill out a form with information about their AI model. Secondly, if the information is correct, users are redirected to a code editor where they can submit the algorithm's code and upload the corresponding pre-trained model files. Finally, all the algorithm files are zipped and sent to the backend. As mentioned, the application offers distinct functionalities according to the type of user: researcher or health professional. Both groups can view DICOM images using OHIF and upload studies to Orthanc. Nevertheless, from an AI perspective, the possibilities differ. Users with the researcher role may add AI models to the application to perform inference on the images they have uploaded. Deploying a model in order to perform inference means that users have trained a model, tested its performance, and decided to use it to make predictions on new data [21], in this case, DICOM images. This way, researchers test their models on new and undisclosed data. On the

other hand, instead of adding and deploying models, healthcare professionals are perhaps more interested in using available models to aid them in image analysis. With this in mind, MCC enables researchers to add and test their AI models on DICOM images while allowing health professionals to apply models made available in the application and see the outcome results;

– *Sharing of Imaging Data and AI Models*: from a healthcare point of view, there are potential benefits for patients' well-being when health professionals can share patients' medical imaging data. Such data not only includes the image itself. It also comprises measurements and annotations that the health professional user may perform in the patients' study. The ability to share such information with other health professionals that use the application may facilitate the analysis of that data, accelerate the diagnosis procedure and help achieve more accurate clinical results. From a biomedical research perspective, if investigators can get practical insights from their models and make them available to other users, the impact of the models is significantly enhanced. Also, if the model owner concedes access permission to other research users of the application, they can cooperate in editing, improving and testing such a model. With this in mind, MCC allows the creation of user groups, enabling the sharing of studies between health professional users and models between researchers, respectively.

3.2 Django Backend API

The server-side software architecture was implemented using Django, a Python web framework [5]. Since medical data is increasing exponentially, such architecture intends to be efficient, highly available and scalable. Besides, health professionals want to view, store, share and process images independently of their location or devices and with minimum downtime. Django comprises the main endpoint API whose methods allow tasks such as storing data in the database or processing the data model entities of Figure 2. Each method corresponds to a service, which can be called through Django's REST API.

The chosen database was PostgreSQL [12] since it has plenty of features to help developers build applications and administrators protect data integrity. The database comprises four entities: User, Study, Model and Patient. Each user (researcher or health professional) can upload several DICOM studies to the application. However, each study only belongs to one user (the one who uploaded it). Likewise, every researcher can upload various AI models, but each model only belongs to one user. The latter, however, does not imply that the owner of the study (or model) cannot share that entity with other users. The patient entity can also have several studies associated, but each study only belongs to one patient. Database attributes regarding users' and patients' sensitive clinical information were encrypted using 256-bit AES encryption. These include the user's password, medical certificate, code for two-factor authentication, patient's name and all keys and initialisation vectors for the AES cypher. AES is employed as the chief encryption primitive, which uses the permutation method in a specified number of rounds allowing better security, especially against brute force attacks.

Moreover, an instance of Orthanc was created to store the DICOM studies information in an encrypted state. The metadata related to the physician, hospital, and study modality were encrypted for enhanced privacy and security.

The application has two central users: the health professional and the researcher. Everyone whose intentions are only investigational-driven can be a researcher. In turn, for someone to authenticate as a health professional, must provide a medical certificate, whose validation is done manually by the application administrator. Through the application's authentication system REST API, a user can authenticate himself and ensure that no one can enter his account without the proper credentials. Such authentication system was implemented using Djoser, a library that provides a set of Django Rest Framework views to handle actions such as registration, login, logout, password reset and account activation. When a user signs in, he has an access token, which refreshes itself from time to time, allowing him to remain authenticated. Also, social networks authentication was incorporated with the help of Python Social Auth, using the OAuth2.0 protocol. All API methods are protected as, when a call occurs, an API key (access token) is required to get a valid response from that method. Furthermore, two-factor authentication was enabled to allow better security overall. Therefore, besides providing his credentials, the user needs to insert a secret key, which can be present in a third-party tool such as Google Authenticator.

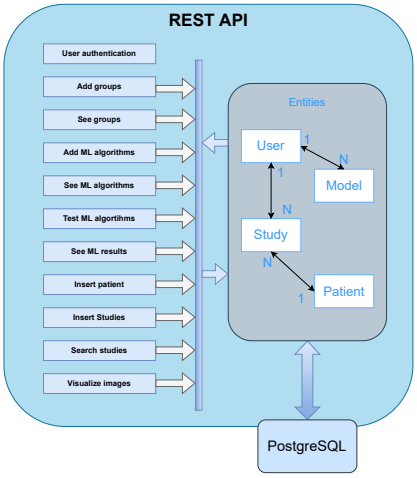

Fig. 2: Backend API.

3.3 Machine Learning Modelling

Machine Learning model deployment is the method by which a model is integrated into an existing production environment to make decisions based on data. One of the typical ways to deploy an ML model is to create a web service for prediction. Usually, the first step is to create an ML model, train it and validate its performance. Second, the model needs to be persisted. Persistence can be

achieved by storing the trained model in a file. Finally, the pre-trained model can be served using a web framework [20]. MCC focus on this last stage of the model's life cycle – using the pre-trained model to make predictions on new data, in this case, medical images.

For that matter, AI models need to be added to the application. In this case, the application accepts algorithms written in Python. To add a model, researchers first need to fill out a form with information about the algorithm, e.g., name, description, model architecture, and task (image segmentation, object localisation, image processing, lesion detection or classification). In the second step, they are redirected to a code editor where they can submit the algorithm's code and upload the corresponding pre-trained model files (Figure 3a).

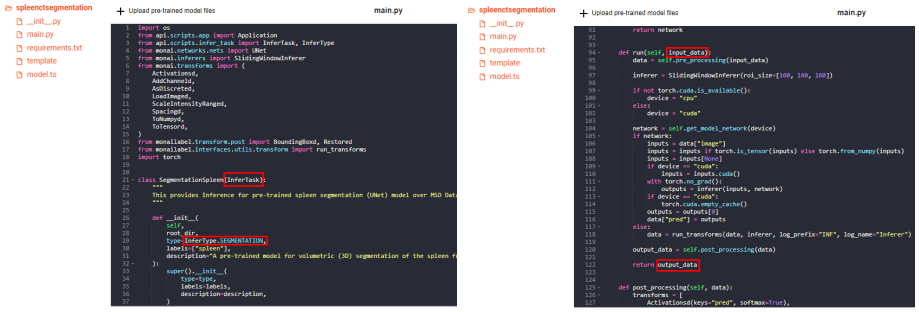

Fig. 3: UI of a researcher. Code editor.

The editor also includes a template that users should follow to submit their code. Such a template uses three key Python modules: "InferTask", "InferType", and "Application". Each algorithm is seen as an inference task, so it must correspond to a class that inherits the "InferTask" module. Each task has a "type". For instance, if the model performs image segmentation, its type is "SEGMENTATION". Each "type" is defined as shown in Figure 3a. Finally, the "main.py" file needs another class that must inherit the "Application" module. The latter is what turns the algorithm's code into code that can run on MCC.

The "run" method (Figure 3b) is mandatory, and it should contain all the code that the user wants to run, i.e., the workflow of the algorithm. All the other methods or functions possibly added to the algorithm's class should be called here. To apply the algorithm, the backend will specifically look for the "run" method and execute its code. Finally, the output will be written in the appropriate format to be displayed in the viewer.

When a "Publish" button is clicked, all the algorithm's files are zipped and sent to the backend, which, first, validates the files, that is, checks if there are no missing files, if the code template was respected and, then, stores the algorithm in the database. In parallel, a Docker [6] image based on the model API is

created for that algorithm. It will be used for the creation of the correspondent deployment with Kubernetes [8].

Once the model is submitted, it is available for the user to perform inference on DICOM images. Since AI is incorporated as a second reader functionality, it is required that AI predictions are available in the same image viewing environment as the images and accessible through a simple click. The user has a choice of multiple models to select from, and the series of images that the user is viewing at that moment is used as input to the chosen model. Once a model is selected, the unique identifier of that series is sent to the backend, which requests those images from Orthanc and caches them. Moreover, the model zip file is retrieved from the database, unzipped, and an instance of the algorithm's class is created. Ultimately, the "run" method of that instance is executed, and once the output is returned, it is sent to the viewer, which displays it.

It is worth mentioning that the AI workflow of the application is based on the open-source MONAI Label project, which provides a framework for developing and deploying AI applications [9].

3.4 Docker and Kubernetes for ML inference

Machine learning models can take quite a long time to predict a result, and, during that time, the user may want to navigate through the application while waiting for the response. Thus, deployment is perhaps one of the most overlooked topics in the Machine Learning world. Accordingly, for the application to have high availability, scalability and efficiency, technologies such as Docker and Kubernetes are extremely valuable. Docker takes away repetitive, mundane configuration tasks and is used throughout the development lifecycle for fast, easy and portable application development. Kubernetes, in turn, is utilised for automating deployment, scaling, and management of containerised applications using Docker runtime. Such implementation allows the user to update or roll back the version of his models. If he wants to change the model itself, a new image will be created for that model, and the deployment will be updated so that pods can run a container with that updated image. A pod is the smallest and most basic deployable object in Kubernetes. It represents a single instance of a running process (docker container) in the cluster.

As described earlier, each model has its API and runs in a container, where all its dependencies are stored. A Kubernetes cluster with one node was created to orchestrate numerous containers. In such node, each container is allocated to a unique pod and three pod replicas, called a deployment, are constantly running each model API and return the response to the frontend. Each deployment has its service, and each service is exposed to the outside via an ingress. Ingress is a controller that redirects to a specific service depending on the path provided in the URL. Thus, depending on the model that the user selects, the ingress will redirect to the appropriate service, which, in turn, redirects to one of the three pods of the deployment that runs the model API (Figure 4).

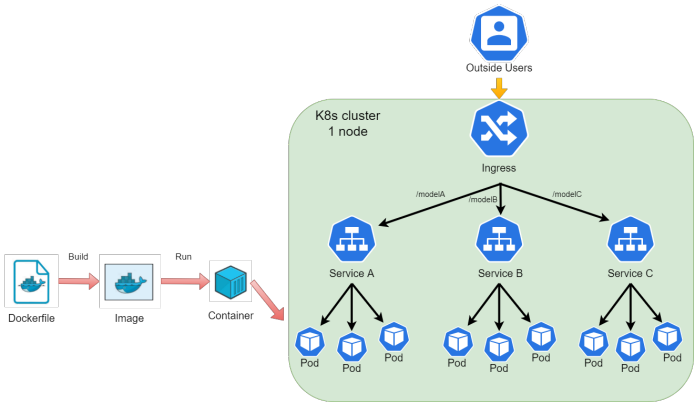

Fig. 4: Kubernetes Cluster Architecture and Pipeline.

4 Results

The proposed web application is in the proof-of-concept stage. Nevertheless, it has been validated from a technical perspective through a set of experiments, such as testing the performance of the backend in terms of scalability, availability, latency, privacy and security; and the efficiency of PostgreSQL database and Orthanc server querying.

The first step that any user must go through to use the application is to authenticate himself with the correct credentials (email and password) as shown in Figure 5 or pass the two-factor authentication phase.

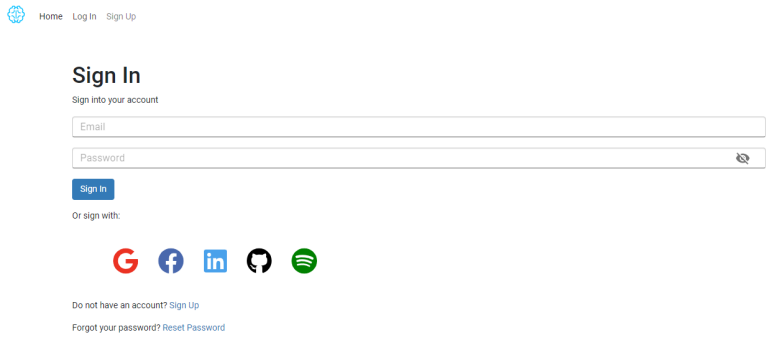

Fig. 5: Sign in page.

Furthermore, both groups of users, researchers and health professionals, can access a list (Figure 6) where all the studies they have uploaded (or to which they have been granted access) are displayed. Those studies can be filtered by the information of some DICOM tags, namely, patient name or ID, study modality and date. On this page, users can also upload more studies using the "+" button on the right side of the screen.

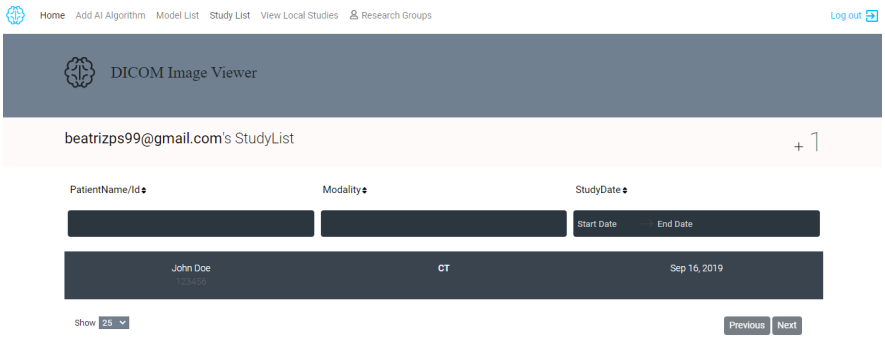

Fig. 6: UI. Example of a researcher screen displaying a study list.

Besides a study list, researchers also have access to a model list, where all the ML models they have access to are displayed. It also allows them to pick a model and be redirected to a page where they can edit its information and code.

When users click on one of the studies from Figure 6, they are redirected to the OHIF viewer, where they can view the corresponding DICOM images. As mentioned, since AI is incorporated as a second reader functionality, it is required that AI predictions are available in the same image viewing environment as the DICOM images. Therefore, users have access to a panel where they can select and run AI models. Figure 7 shows an example of a user screen where the user chose a model that performed the automatic segmentation of the spleen in a CT series. As noted, sensitive patient information in the DICOM studies must be private and secure. For that matter, data is encrypted at rest in the Orthanc server and at transit over HTTPS. Figure 7 presents an example of a DICOM study in which the metadata tags were encrypted with 256-bit AES encryption.

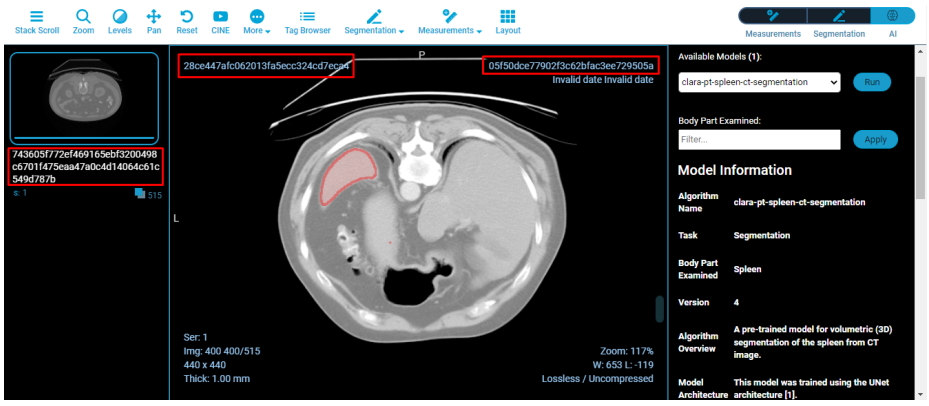

Fig. 7: UI. Example of a user screen displaying images and applying AI to them.

5 Conclusions and Future Work

MedCloudCare enables healthcare professionals to remotely access and evaluate patients' medical imaging information, which is encrypted to assure security and privacy regarding sensitive information. This platform provides a method for enhancing treatment adherence as it allows health professionals to share medical images and quickly and correctly analyse them. It is also possible to measure biomedical parameters and identify, classify, and quantify patterns in those images with the help of AI.

From a biomedical research perspective, MCC provides research users with a way to add and use pre-trained ML or DL models. The fact that they can test their AI models on medical images and see the result can help them understand if those models have the desired quality.

Additionally, MCC was tested using trials that allow the validation from a technical point of view, assuring that there are no errors. However, besides all the core features and functionalities, there is still work to be done. A core function is the need to encrypt the pixel data of the DICOM files and not only their metadata. Finally, MCC requires validation in third-party cloud infrastructures, namely Google Cloud Platform (GCP). This step will be focused on using the Google Kubernetes Engine (GKE) and will be automated using Terraform.

Acknowledgements This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within the project LA/P/0063/2020, and through a PhD Fellowship (SFRH/BD/146528/2019 - Cláudia Brito).

References

1. 3D Slicer, https://www.slicer.org/, Last accessed in April, 2022.
2. Cornerstone.js, https://www.cornerstonejs.org/, Last accessed in December, 2022.
3. DeepInfer, http://www.deepinfer.org/, Last accessed in April, 2022.
4. DICOMweb™, https://www.dicomstandard.org/using/dicomweb, Last accessed in December, 2022.
5. Django, https://www.djangoproject.com/, Last accessed in April, 2022.
6. Docker, https://www.docker.com/, Last accessed in April, 2022.
7. ePAD - Web-based platform for quantitative imaging in the clinical workflow, https://epad.stanford.edu/, Last accessed in April, 2022.
8. Kubernetes, https://kubernetes.io/, Last accessed in April, 2022.
9. MONAI Label, https://github.com/Project-MONAI/MONAILabel, Last accessed in January, 2022.
10. Orthanc - DICOM Server, https://www.orthanc-server.com/, Last accessed in December, 2022.
11. OsiriX DICOM Viewer, https://www.osirix-viewer.com/, Last accessed in April, 2022.
12. PostgreSQL, https://www.postgresql.org/, Last accessed in April, 2022.
13. Precision Imaging Metrics, https://www.precisionmetrics.org/, Last accessed in April, 2022.

14. React – A JavaScript library for building user interfaces, https://reactjs.org/, Last accessed in December, 2022.
15. TOMAAT, https://tomaat.readthedocs.io/en/latest/, Last accessed in April, 2022.
16. XNAT, https://www.xnat.org/, Last accessed in April, 2022.
17. Brito, C.: Cloud-based Analytics for Monitoring and Classification of Arrhythmias (2018)
18. Min, Q., Wang, X., Huang, B., Xu, L.: Web-Based Technology for Remote Viewing of Radiological Images: App Validation. Journal of Medical Internet Research **22**(9) (September 2020). https://doi.org/10.2196/16224
19. Min, Q., Wang, Z., Liu, N.: An Evaluation of HTML5 and WebGL for Medical Imaging Applications. Journal of Healthcare Engineering **2018** (2018). https://doi.org/10.1155/2018/1592821
20. Murallie, T.: 3 Ways to Deploy Machine Learning Models in Production, https://towardsdatascience.com/3-ways-to-deploy-machine-learning-models-in-production-cdba15b00e, Last accessed in April, 2022.
21. Pinhasi, A.: Deploying Machine Learning models to production - Inference service architecture patterns, https://medium.com/data-for-ai/deploying-machine-learning-models-to-production-inference-service-architecture-patterns-bc8051f70080, Last accessed in April, 2022.
22. PostDICOM: Top 25 Free Dicom Viewers for Doctors, Medical Students, and Health Professionals, https://www.postdicom.com/en/blog/top-25-free-dicom-viewers, Last accessed in December, 2022.
23. Ramos, A.: Deep Learning Applied to Medical Imaging (2019)
24. Sedghi, A., Hamidi, S., Mehrtash, A., Ziegler, E., Tempany, C., Pieper, S., Kapur, T., Mousavi, P.: Tesseract-medical imaging: open-source browser-based platform for artificial intelligence deployment in medical imaging. Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling **10951**, 446–451 (April 2019). https://doi.org/10.1117/12.2513004
25. Suganyadevi, S., Seethalakshmi, V., Balasamy, K.: A review on deep learning in medical image analysis. International Journal of Multimedia Information Retrieval 2021 pp. 1–20 (September 2021). https://doi.org/10.1007/S13735-021-00218-1
26. Toshpulatov, Z., Marti, R., Diaz, O.: DeepDraw! Developing a web application for medical image annotation and computer aided analysis (2019), http://eia.udg.edu/~aoliver/maiaDocs/bookMaia2nd_small.pdf
27. Yuan, R., Luo, M., Sun, Z., Shi, S., Xiao, P., Xie, Q.: RayPlus: a Web-Based Platform for Medical Image Processing. Journal of Digital Imaging **30**(2), 197 (April 2017). https://doi.org/10.1007/S10278-016-9920-Y
28. Ziegler, E., Urban, T., Brown, D., Petts, J., Pieper, S., Lewis, R., Hafey, C., Harris, G.: Open Health Imaging Foundation Viewer: An Extensible Open-Source Framework for Building Web-Based Imaging Applications to Support Cancer Research. JCO Clinical Cancer Informatics (4), 336–345 (September 2020). https://doi.org/10.1200/cci.19.00131