Web-based AI System for Medical Image Segmentation

Hao Chen[1*], Taowen Liu[1*], Songyun Hu[1], Leyang Yu[1], Yiqi Li[1], Sihan Tao[1], Jacqueline Lee[1], and Ahmed E. Fetit[1,2] (✉)

[1] Department of Computing, Imperial College London, UK
[2] UKRI CDT in Artificial Intelligence for Healthcare, Imperial College London, UK
a.fetit@imperial.ac.uk

Abstract. Image segmentation is a crucial step in the diagnosis of brain tumours, and machine learning has emerged as a promising tool for tumour characterisation from medical imaging data. Despite their enormous potential in automatic segmentation of brain tumours from complex MRI scans, the implementation and use of machine learning algorithms can often present practical challenges to medical imaging researchers. This paper introduces a web-based GUI application designed to integrate all the components needed in deep learning workflows, allowing medical imaging researchers to seamlessly train and infer on data stored on in-house servers or on local machines. Our platform simplifies the process of training and inferring on MRI data using state-of-the-art models, supports integration with XNAT servers, and incorporates powerful tools for visualizing inference results.

Keywords: Deep learning systems · Magnetic resonance imaging · Image segmentation · XNAT · Image informatics

1 Introduction

Magnetic Resonance Imaging (MRI) is a non-invasive technique that can be used to detect, characterise, and monitor various diseases and conditions including brain tumours. Machine learning (ML) techniques, including deep learning-based algorithms, have demonstrated enormous potential in automatic segmentation of brain tumours from complex MRI scans. However, the way modern ML workflows are implemented in research settings presents a number of barriers to medical imaging researchers who may be novice to the computing aspects of the work. First, GPU acceleration is normally needed to provide the necessary compute power for training computationally intensive models, making GPU cards an invaluable component in most ML workflows. Furthermore, implementing the source code of a network architecture, as well as configuring GPU cards on research labs and hospital hardware may impede the adoption of powerful ML innovations by non-computer scientists. Moreover, researchers interested in training and deploying ML models should ideally be able to seamlessly use in-house servers and compute resources for getting access to the training data as well as storing any processed data, introducing further challenges in terms of writing data to and from the available servers.

To address these challenges, in this paper we introduce a web-based GUI application to the medical imaging community, integrating all the components typically needed in deep learning workflows into a single system. Our proposed web-based system allows users to

[*] Equal contributions.

seamlessly train and infer on MRI data stored on in-house servers or local machines, without the need for a programming background, using a variety of advanced ML models whose architectures and training parameters can be easily configured in the GUI.

Our contributions are as follows: *1)* we develop a web-based platform that enables medical imaging researchers to configure, train, and evaluate segmentation models using MRI data stored locally, *2)* we incorporate ML segmentation models (e.g. HyperDenseNet3D [3], ResNet3D [13], Gibbs ResUnet [2]) while ensuring that the model architectures and training hyper-parameters remain configurable by the user, and *3)* we integrate the eXtensible Neuroimaging Archive Toolkit (XNAT) [9] into our system, a popular medical image data management system, to allow users to easily use data stored on XNAT servers and carry out model training and/or inference.

Our web-based artificial intelligence (AI) system for medical image segmentation enables the following capabilities: *i)* tracking training progress through TensorBoard[3], *ii)* saving trained models for later use or further fine-tuning, *iii)* visualising inferred results using embedded third-party Neuroimaging Informatics Technology Initiative (NIfTI) [10] viewers, *iv)* allowing users to also evaluate their trained models on a selected test dataset and view the results in the provided evaluation history table, *v)* allowing users to log in to existing XNAT accounts, download data and upload inferred results from/to XNAT servers, and *vi)* evaluating on images from XNAT or uploaded locally. Whilst the main use-case discussed throughout this paper is brain tumour segmentation from MRI scans, the system can be easily adapted to support a variety of datasets and other anatomies or imaging modalities.

2 Related Work

Several frameworks have been developed that enable researchers to store, manage, and share medical imaging data, e.g. Dicoogle [14], an open source Picture Archiving and Communications System (PACS) archive, the Open Health Imaging Foundation (OHIF) Viewer [17], and the eXtensible Neuroimaging Archive Toolkit (XNAT) [9]. XNAT has been widely adopted as an infrastructure backbone for the organisation, management, and distribution of large imaging repositories [4]; examples include NeuroAI-HD [16], the Human Connectome Project (HCP) [15], the Developing Human Connectome Project (dHCP) [8], and the NITRC image repository [5]. Additionally, several projects have successfully interfaced XNAT with programming languages and interoperability standards, e.g. RXNAT [4], PyXNAT [12], and FHIR on XNAT [6]. Whilst researchers should be able to directly use the data held on available XNAT servers to carry out model training and/or inference, little work was reported in the literature on providing no-code interfaces for incorporating deep learning workflows with XNAT.

3 Designing a Web-based AI System for Medical Image Segmentation

In this section, we discuss the system architecture of our proposed web-based platform for medical image segmentation and the datasets used when deploying our system.

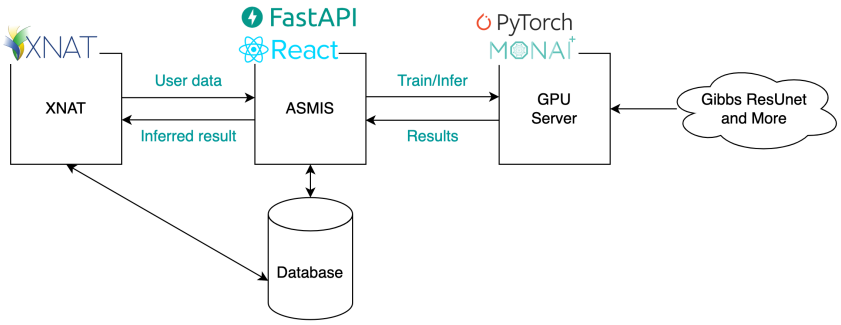


Fig. 1: System architecture of our web-based system for medical image segmentation.

3.1 System Architecture

Figure 1 gives an overview of the system's architecture. Our web-based platform can be used to either load MRI data stored locally or using XNAT [9]. XNAT provides a variety of tools for storing, organising, and exporting research imaging data and is widely used by medical imaging researchers worldwide across research labs, hospitals, and universities. We used FastAPI[4] and React[5] to implement the web interface that allows users to seamlessly communicate with XNAT and to upload data. Crucially, we used HTTPS in both the back-end and the front-end in order to ensure the safety of network traffic.

Regarding training and inference, we used PyTorch[6], a popular open-source machine learning library used for deep learning, for configuring, training deep learning models, and ultimately using the trained models to infer on MRI images. The web-based system assumes that a GPU-enabled server is available for the users to carry out image training and inference. However, if no GPU resources are available, our system automatically falls back to available CPUs instead. The web server communicates with the GPU server using a combination of FastAPI HTTPS calls and WebSocket connections; FastAPI HTTPS calls are used to transmit the training configuration, dataset, and model architecture, while WebSocket connections are used to provide real-time updates on the training process.

Figure 2 illustrates the workflow for training and inferring on medical images. The system uses data from XNAT or stored locally to create a dataset. With regards to data management in the context of MRI image segmentation, a 'dataset' is a collection of 3D scans and segmentation pairs used to train a model. To write a new dataset onto the system, the user can either upload local NIfTI images and segmentation labels, or retrieve and download an existing dataset from XNAT, which would require the XNAT Dataset Plugin. For XNAT access, the login page of our application enables users to log into their XNAT account using their access credentials, and identify the XNAT server they wish to use via a URL. This functionality enables users to easily switch between different XNAT servers, and ensures that they can only access data held on XNAT for which they have been granted permission. Crucially, XNAT offers fine-grained control of authorization which our system inherits so users could have great flexibility of authorization.

[3] https://www.tensorflow.org/tensorboard
[4] https://fastapi.tiangolo.com/
[5] https://react.dev/
[6] https://pytorch.org/

Once a dataset is written onto the system, it will be stored in the application's database in a ML-friendly storage format and could be readily used for model training and evaluation. The final step in the workflow is represented by the inference stage, where users can upload images from XNAT or from their local machines, and then view and save results.

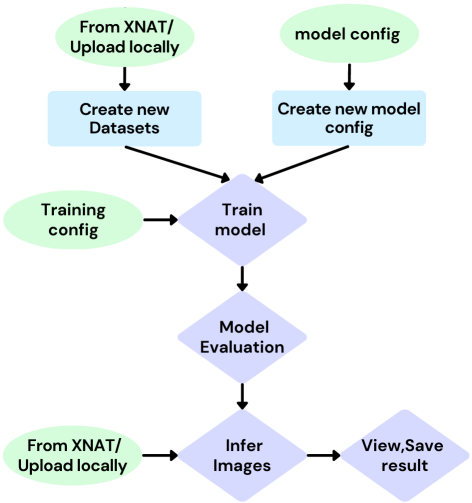


Fig. 2: Workflow for training and inferring on medical images.

To ensure that the application is robust and reliable, we employed static type checks and unit tests during our continuous iteration/continuous development pipeline. Specifically, we employed pytest for testing and mypy for performing static type checks; and we carried out a variety of validation checks for inputs to our application, e.g. training hyper-parameters.

3.2 Datasets

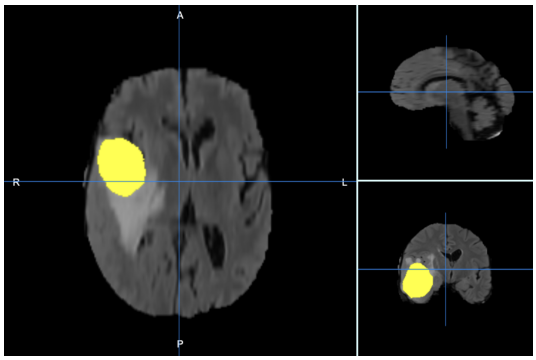


Fig. 3: Tumour label overlaid over a 3D MRI scan; scan obtained from the MSD dataset [1].

Our application provides users with the option to upload their own MRI data onto XNAT, which can help adapt any developed models to the statistical distributions of a specific hospital or research centre. In our system, we made use of the publicly available brain tumour segmentation task data obtained from the Medical Segmentation Decathlon (MSD) challenge [1]. The MSD is a benchmark dataset that was designed to evaluate the performance of state-of-the-art segmentation algorithms. It consists of 10 different medical imaging segmentation tasks, covering several anatomical structures, including the liver, pancreas, and brain tumours. Our system was deployed with the brain tumour segmentation task data, which comprises two 3D NIfTI files for each data point, one for the MRI scan and the other for the label. The NIfTI files in the dataset are 240x240x155 voxels, with a voxel size of 1 mm^3. The labeled data points include information about the location and size of the brain tumour, as well as the surrounding brain tissues.

4 Training and Inference

In this section, we describe the key components regarding training and evaluation in our system: model configuration, training configuration, and inference and evaluation, as well as evaluate the usability of our system. Figure 4 shows the user view of the web application for the training step.

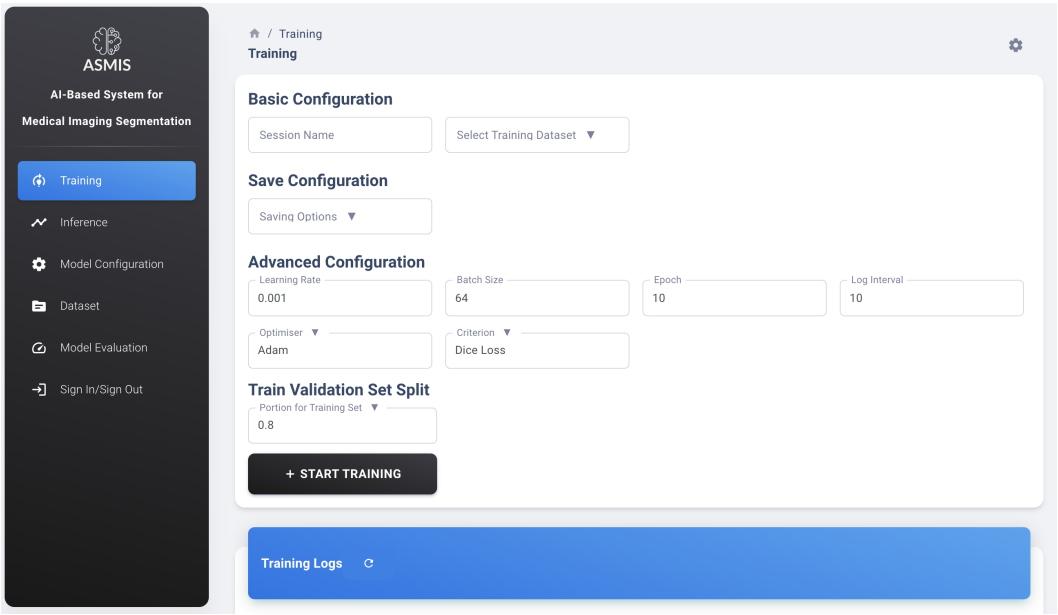


Fig. 4: User interface of the web application for the training step.

4.1 Model Configuration

To create a new segmentation model, users can use our system's model configuration page and choose their preferred network architecture. Several architectures are currently avail-

able, including a variety of models from the Medical Imaging Model Zoo [11] such as HyperDenseNet3D [3], HRNet3D [7], and ResNet3D [13]; as well as Gibbs ResUNet [2]. All models are pre-defined in PyTorch.

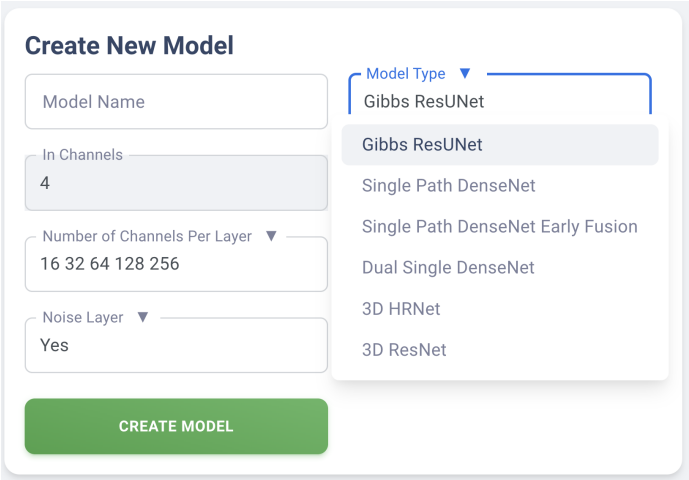
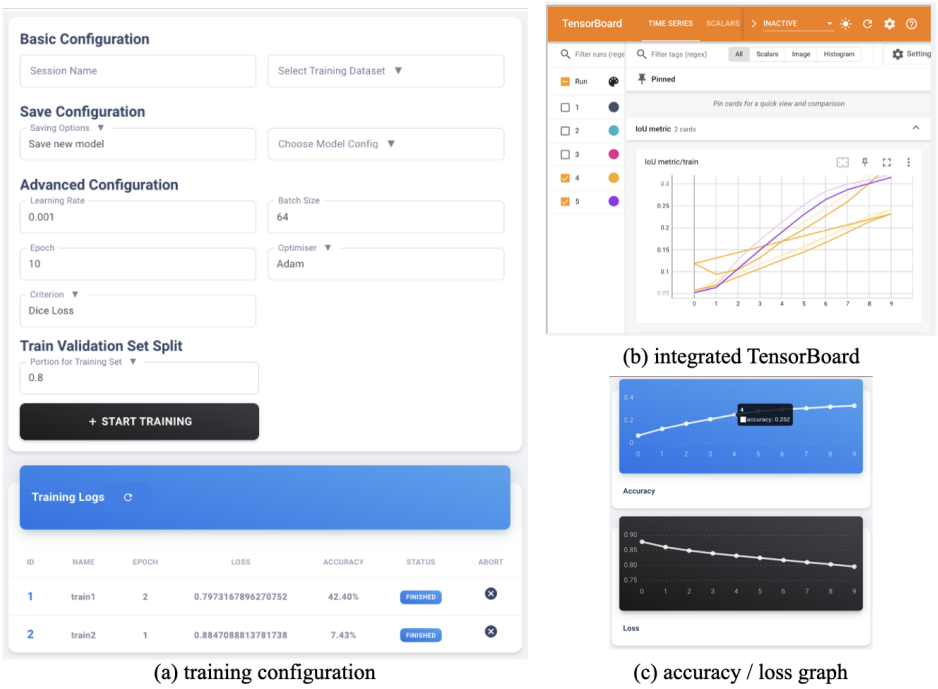


Fig. 5: User interface showing how a model can be configured.

Figure 5 shows a selection of the model types that can be selected and configured. Users can also specify exact parameters before training, such as the number of channels, number of layers, and whether to include a Gibbs Noise Layer in the case of Gibbs ResUnet. This level of customization allows users to tailor the model's architecture to suit the specific needs of their dataset and experiment, potentially improving the overall performance of the segmentation task. Once the user has chosen their desired model configuration, these settings are stored in the system's database for easy retrieval and management. This feature enables users to revisit and modify their configurations, experiment with different settings, and compare the performance of various model architectures and training parameters. By providing a user-friendly interface for selecting and customizing pre-defined models in PyTorch, our system aims to lower the barriers for medical professionals to access and apply advanced ML techniques for brain tumour segmentation tasks.

4.2 Training Configuration

Model training can be initiated using the GUI via a training page (see Figure 6). Upon selecting the training set, the user can either train a new model or fine-tune an existing one. Hyperparameters can be specified in the advanced configuration section, including the learning rate, batch size, number of epochs, optimizer, as well as loss function. Users can then set the interval at which training logs are saved, and the split ratio determines the proportion of the training set and validation set. After the configuration step is complete, the training process can begin, and relevant metrics and the training status will be displayed in the 'training log' table. Moreover, our application has an integrated TensorBoard display which helps visualize training progress and results. A detailed training page will then be



(a) training configuration (b) integrated TensorBoard (c) accuracy / loss graph

Fig. 6: User interface of the training page.

available for each training session created, plotting the computed training accuracy and loss in real-time.

The dataset is stored in server in the supported NIfTI format. Upon training, the dataset is firstly preprocessed, which involves cropping, normalization, and other necessary adjustments to ensure consistency. The preprocessed data is then converted into PyTorch tensors, which are compatible with the machine learning library. The network architecture is prestored in the dataset and is automatically loaded during the training process.

Once the GPU server receives the necessary information, it begins the training process, periodically sending updates to the web server with the latest training metrics, such as loss and accuracy. The web server then updates the training log table and the integrated TensorBoard display in real-time, allowing users to monitor the progress of the ongoing training session. This implementation design ensures seamless communication between the user interface, web server, and GPU server, enabling users to focus on configuring and monitoring the training process without worrying about the underlying complexities.

4.3 Inference and Evaluation

Our application enables users to evaluate their trained models on a selected test set, providing valuable insights into the model's performance. The Intersection over Union (IoU) metric is displayed and automatically updated as the evaluation progresses, offering users a clear understanding of the model's segmentation accuracy. Once a user has finalized the training process, they can employ the trained model to infer the brain tumour segmentation

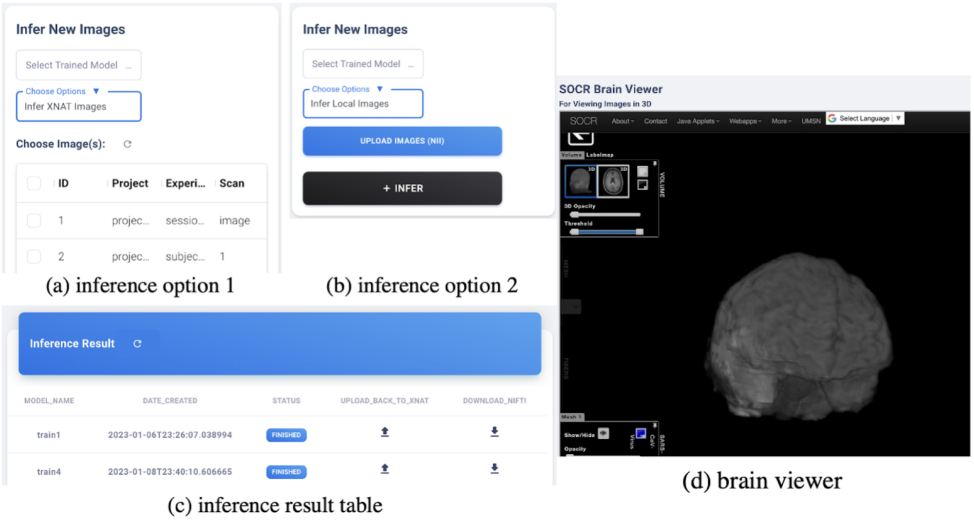


Fig. 7: User interface of the inference page, including the third-party brain viewer.

of an MRI image. The user has the option to choose between using images from XNAT or uploading local images for inference, and all the data can either be downloaded to the local machine or uploaded to the XNAT server as needed, providing ample flexibility in data management.

Furthermore, our application incorporates two third-party viewers, SOCR Brain Viewer[7] and Papaya[8], that allow users to rapidly inspect and visualize the segmentation output. These image viewers provide various visualization options, allowing users to assess the quality of the segmentation and identify potential areas for improvement. Figure 7 shows the user view of the inference page, including the brain viewer.

4.4 Evaluating Usability

To evaluate the usability of our system, we asked five users with extensive ML background to use our application, and assigned them tasks focused on model training, evaluation and inference. With an average of 2.2 questions asked per user, each task took an average of 341 seconds to complete. We also asked the users to estimate the amount of time they would take to perform the same process with Python code, based on their prior experience. This amounted to 35 minutes on average, suggesting that our application can substantially improve the efficiency of modern ML workflows (see Figure 8).

5 Conclusion and Future Work

In this paper, we presented an efficient web-based platform that integrates ML models with medical imaging tools through a user-friendly and intuitive interface which does not require

[7] https://socr.umich.edu/HTML5/BrainViewer/
[8] https://www.fmrib.ox.ac.uk/ukbiobank/group_means/index.html

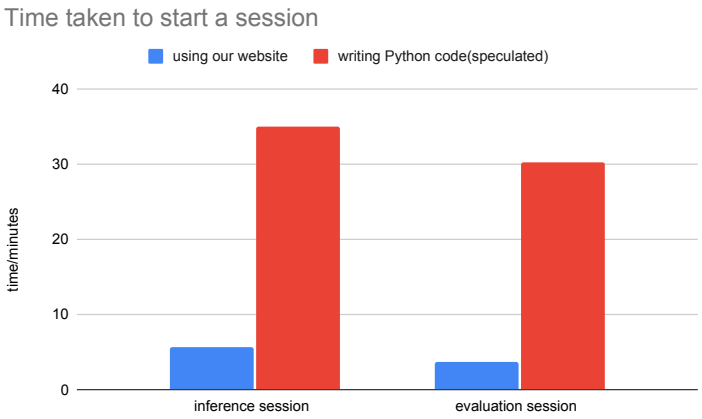


Fig. 8: Time taken to start an inference/evaluation session

any programming experience. The application greatly simplifies the process of training and inferring MRI data for brain tumour segmentation research, empowering users to interact with the system by integrating all the components typically needed in deep learning workflows. The platform also supports uploading and downloading data to/from to XNAT servers, tracking training progress through TensorBoard, and viewing inferred results using embedded third-party NIfTI viewers.

Our web-based AI system surpasses the individual components it integrates by carefully handling edge cases and validating all inputs before training, evaluation, and inference sessions, thus simplifying the user process for debugging ML algorithms. There are several avenues for future work; first, we aim to provide flexibility for users to be able to define, implement, and upload their own PyTorch model. Moreover, we plan to extend our system to incorporate other datasets and other anatomies or imaging modalities. Finally, for a more secure approach to handling sensitive medical data, implementing federated learning with XNAT servers represents a promising direction of future work.

6 Acknowledgments

The research of Dr Ahmed E. Fetit was supported by the UKRI CDT in Artificial Intelligence for Healthcare in his role as Senior Teaching Fellow (grant number EP/S023283/1). For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

References

1. Antonelli, M., Reinke, A., Bakas, S., Farahani, K., Kopp-Schneider, A., Landman, B.A., Litjens, G., Menze, B., Ronneberger, O., Summers, R.M., et al.: The medical segmentation decathlon. Nature Communications **13**(1), 4128 (2022)
2. Cabrera, Y., Fetit, A.E.: Reducing CNN textural bias with k-space artifacts improves robustness. IEEE Access **10** (2022)

3. Dolz, J., Gopinath, K., Yuan, J., Lombaert, H., Desrosiers, C., Ben Ayed, I.: Hyperdense-net: a hyper-densely connected CNN for multi-modal image segmentation. IEEE Transactions on Medical Imaging **38**(5) (May 2019)
4. Gherman, A., Muschelli, J., Caffo, B., Crainiceanu, C.: Rxnat: An open-source R package for XNAT-based repositories. Frontiers in Neuroinformatics **14**, 572068 (2020)
5. Kennedy, D.N., Haselgrove, C., Riehl, J., Preuss, N., Buccigrossi, R.: The NITRC image repository. NeuroImage **124**, 1069–1073 (2016)
6. Khvastova, M., Witt, M., Essenwanger, A., Sass, J., Thun, S., Krefting, D.: Towards interoperability in clinical research-enabling fhir on the open-source research platform XNAT. Journal of Medical Systems **44**, 1–5 (2020)
7. Li, S., Ke, L., Pratama, K., Tai, Y.W., Tang, C.K., Cheng, K.T.: Cascaded deep monocular 3D human pose estimation with evolutionary training data. In: 2020 IEEE/CVF CVPR (Jun 2020)
8. Makropoulos, A., Robinson, E.C., Schuh, A., Wright, R., Fitzgibbon, S., Bozek, J., Counsell, S.J., Steinweg, J., Vecchiato, K., Passerat-Palmbach, J., et al.: The developing human connectome project: A minimal processing pipeline for neonatal cortical surface reconstruction. Neuroimage **173**, 88–112 (2018)
9. Marcus, D.S., Olsen, T.R., Ramaratnam, M., Buckner, R.L.: The extensible neuroimaging archive toolkit: An informatics platform for managing, exploring, and sharing neuroimaging data. Neuroinformatics **5**(1) (Mar 2007)
10. Moore, C.M.: Nifti (File format) | radiology reference article | radiopaedia.org
11. Nikolaos, A.M.: Deep learning in medical image analysis : a comparative analysis of multi-modal brain-MRI segmentation with 3D deep neural networks (Jul 2019)
12. Schwartz, Y., Barbot, A., Thyreau, B., Frouin, V., Varoquaux, G., Siram, A., Marcus, D.S., Poline, J.B.: PyXNAT: XNAT in python. Frontiers in Neuroinformatics **6**, 12 (2012)
13. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition (2017)
14. Valente, F., Silva, L.A.B., Godinho, T.M., Costa, C.: Anatomy of an extensible open source pacs. Journal of Digital Imaging **29**, 284–296 (2016)
15. Van Essen, D.C., Smith, S.M., Barch, D.M., Behrens, T.E., Yacoub, E., Ugurbil, K., Consortium, W.M.H., et al.: The WU-Minn human connectome project: an overview. Neuroimage **80**, 62–79 (2013)
16. Vollmuth, P., Foltyn, M., Huang, R.Y., Galldiks, N., Petersen, J., Isensee, F., van den Bent, M.J., Barkhof, F., Park, J.E., Park, Y.W., et al.: Artificial intelligence (AI)-based decision support improves reproducibility of tumor response assessment in neuro-oncology: An international multi-reader study. Neuro-Oncology **25**(3), 533–543 (2023)
17. Ziegler, E., Urban, T., Brown, D., Petts, J., Pieper, S.D., Lewis, R., Hafey, C., Harris, G.J.: 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 (2020)