Caring is Sharing – Exploiting the Value in Data for Health and Innovation
M. Hägglund et al. (Eds.)

doi:10.3233/SHTI230325

Client-Side Application of Deep Learning Models Through Teleradiology

Sébastien JODOGNE[1]
ICTEAM, UCLouvain, Belgium
ORCiD ID: https://orcid.org/0000-0001-6685-7398

Abstract. Deep learning models for radiology are typically deployed either through cloud-based platforms, through on-premises infrastructures, or though heavyweight viewers. This tends to restrict the audience of deep learning models to radiologists working in state-of-the-art hospitals, which raises concerns about the democratization of deep learning for medical imaging, most notably in the context of research and education. We show that complex deep learning models can be applied directly inside Web browsers, without resorting to any external computation infrastructure, and we release our code as free and open-source software. This opens the path to the use of teleradiology solutions as an effective way to distribute, teach, and evaluate deep learning architectures.

Keywords. Medical imaging, teleradiology, deep learning, WebAssembly

1. Introduction

Deep learning applied to radiology has attracted a lot of interest in the recent years [1]. Convolutional neural networks (CNN) have proved to be an extremely promising building block for the tasks of classification, segmentation, localization, and detection applied to medical images [2]. In particular, the U-Net architecture is considered as a state-of-the art CNN for 2D biomedical image segmentation [3], with extensions to dense 3D segmentation [4]. However, the deployment of successful deep learning models in clinical setups involves multiple challenges that are notably related to the software distribution of the models, to the security of clinical data, and to the certification and reimbursement of associated clinical decision support systems.

Nowadays, the most encountered type of deployment for deep learning models is cloud-based. In this approach, hospitals internally setup a gateway that implements on-demand auto-routing of DICOM images from their PACS (Picture Archiving and Communication System) to a proprietary cloud platform managed by the vendor. After the analysis of the images, the cloud platform typically sends back a DICOM series that contains a PDF report, possibly associated with structured reports or annotated secondary capture images, which can be reintegrated into the PACS. The DICOM gateway is often implemented on the top of the DICOMweb protocol over HTTPS communications to protect the confidentiality of patient data. Despite its popularity, this approach can be

[1] Corresponding Author: Sébastien Jodogne Computer Science and Engineering Department (INGI), Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM), Université Catholique de Louvain, 1348 Louvain-la-Neuve, Belgium, E-mail: sebastien.jodogne@uclouvain.be.

slow because of the time that is necessary to transmit the hundreds of megabytes of a single DICOM study over Internet, and the delay for getting the results of the analysis can largely vary depending upon the workload of the cloud infrastructure, which might be problematic depending on the clinical application. Furthermore, using cloud platforms implies a centralization of imaging studies, which necessitates complex legal agreements with the hospital, and a careful de-identification to avoid leaks of personal data [5]. Finally, as the usage of deep learning increases, hospitals want to take advantage of multiple deep learning algorithms, which causes scalability issues that could be mitigated through the deployment of shared gateways communicating with multiple vendors.

An alternative to cloud-based platforms consists in deploying the deep learning models as central, on-premises software running directly inside the computing infrastructure of the hospital. This approach has the major advantage of restraining the network exchanges to the intranet of the hospital, which guarantees higher network bandwidth, reserved computational power, and security by design in the handling of patient data. On-premises deep learning algorithms can be distributed in two distinct ways: either directly integrated within the PACS, or maintained as a separate solution by a third-party vendor. The former distribution method ensures a single point of contact for the hospital, but is not proposed by all the PACS vendors, and provides a restricted portfolio of algorithms selected by the vendor. The latter method opens a larger choice of algorithms, but requires more complex and costly integrations, as each third-party vendor must install and maintain its own dedicated computing infrastructure inside the hospital.

A third possibility consists in shipping the deep learning models together with the heavyweight viewers installed on imaging workstations, making local computational resources profitable. This avoids the installation of the dedicated computer clusters, which contrasts with on-premises infrastructures, while keeping the DICOM instances inside the intranet. This solution, however, vastly complicates the deployment of the algorithms, as it requires the local team of system administrators to keep the software and the models continuously up-to-date on the target computers. Furthermore, for security reasons, it should stay limited to fixed workstations that are directly connected to the PACS through DICOM query/retrieve, which is typically not the case with physicians' laptops. This type of deployment is also still dependent on the portfolio of deep learning models that are provided by the vendors of heavyweight viewers.

Besides their complexity and cost, the three types of deployments described above are inherently aimed at radiologists working inside state-of-the-art hospitals. This raises concerns about the need of democratizing deep learning models by sharing the technical knowledge behind such algorithms to a more general audience. Indeed, despite a general call to use deep learning, few skilled workers have the opportunity to experiment with machine learning algorithms during their training or as a part of their continuing education. Such technologies are also hardly affordable for emerging economies. Furthermore, even if there exists free and open-source software for medical imaging with advanced deep learning features [6], physicians, or more generally healthcare professionals, expect simple, user-friendly interfaces that are entirely focused on the application of models. It also remains challenging to evaluate the relevance of deep learning models for healthcare, because of the gap between research software and clinical environments. According to this discussion, this paper introduces a free and open-source teleradiology solution that can be used to distribute, then apply deep learning models directly inside Web browsers, without the need of deploying a dedicated computation infrastructure.

2. Methods

Deep learning is inherently asymmetrical: The training of models requires much data and computational power, notably provided by graphics processing units (GPU), whereas their application is within the reach of any modern computer. This allows to envision the application of deep learning architectures by Web browsers, if care is taken to optimize the operations on tensors wrt. a straight JavaScript implementation.

To this end, we propose to rely on WebAssembly, which is an emerging Web technology that has been designed for situations in which complex computations must be carried on by a Web browser. WebAssembly is an open standard specified by W3C that defines a portable binary code (bytecode) to create high-performance Web applications. Thanks to the Emscripten compiler, C++ source code can be converted to WebAssembly bytecode, which is then executed at almost native speed, i.e., faster than JavaScript, by a stack virtual machine running inside the Web browser. WebAssembly is actively backed by the industry and is nowadays supported by all the major Web browsers. It has recently been demonstrated that WebAssembly can be used to develop a cross-platform C++ library for the rendering of DICOM images that is compatible with desktop applications, mobile applications, as well as Web applications [7]. This novel library has been used to design the so-called "Stone Web viewer" as the first free and open-source implementation of a teleradiology solution leveraging WebAssembly.

The technical contribution of this paper consists in demonstrating that 2D U-Net models can be entirely applied within a teleradiology environment, in this case the Stone Web viewer. For this proof-of-concept, we have trained a sample U-Net model to segment the lungs on 2D images of a CT-scanner, using the images of the "*Lung CT Segmentation Challenge 2017*" (LCTSC) dataset [8]. This dataset contains DICOM CT-scan images together with the delineation of both lungs encoded as DICOM RT-STRUCT instances. The resulting U-Net model contains 7,759,521 floating-point parameters and was trained using the TensorFlow library [9] to minimize a smoothed version of the Dice loss function. The training process took 7 hours on a standard desktop computer (Intel Core i7-9700 CPU equipped with a NVIDIA GeForce GTX 1650 GPU). The resulting Dice score was 95%, which indicates average performance.

In order to run the segmentation process inside the Web browser, we have implemented a highly portable library written in plain C++ to apply CNN models. This library is fully compatible with WebAssembly (it is consequently single-threaded) and provides a reference implementation for all the distinct types of layers that are encountered in the U-Net architecture. Figure 1 explains the pipeline by which the models learned by TensorFlow are brought to the Stone Web viewer.

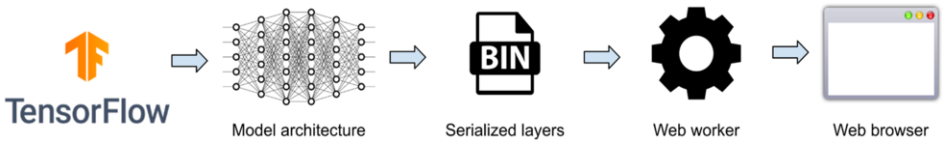

Figure 1. The pipeline used in this work. The U-Net model is first learned in Python using TensorFlow. A conversion script written in Python uses the TensorFlow API to explore the parameters and weights of the various layers in the trained model, then to serialize this information as a binary file using the Google's Protocol Buffers (Protobuf) cross-platform library. This preprocessing is done once and offline. Whenever the Stone Web viewer (that is executed by the Web browser) must apply the model to some image, it asks a HTML5 Web worker to load the model from the Protobuf file, to carry on the computations, then to generate an overlay from the result of the segmentation. This Web worker can be thought of as a "thread" that is executed in the background by the Web browser alongside the main user interface of the Stone Web viewer.

3. Results

The results of our work are depicted in Figure 2. As can be seen, we are able to segment CT-scan slices using our U-Net model, entirely inside the Web browser. The computation time is 11.3 seconds on a standard laptop computer equipped with an Intel i7-1165G7 CPU running Mozilla Firefox 109.0. Importantly, this computation only uses the CPU, so it is not required for the client computer to have dedicated local GPU hardware. Our C++ library for CNN uses the well-known "img2col" trick that rewrites convolutions as matrix products, hereby improving data locality [10]. If the direct "schoolbook" implementation of convolutional layers is used instead, the execution time slows down at 13.6 seconds. Our library also features a custom implementation of matrix product that leverages the optimized WebAssembly SIMD ("single instruction, multiple data"), since this set of instructions is currently not supported by standard C/C++ libraries for linear algebra. By comparison, executing the same code natively on the same laptop takes 5.4 seconds: This halving of the execution time is due to the fact that WebAssembly's SIMD instructions work on a width of 128 bits, whereas native AVX2 SIMD instructions work on a width of 256 bits.

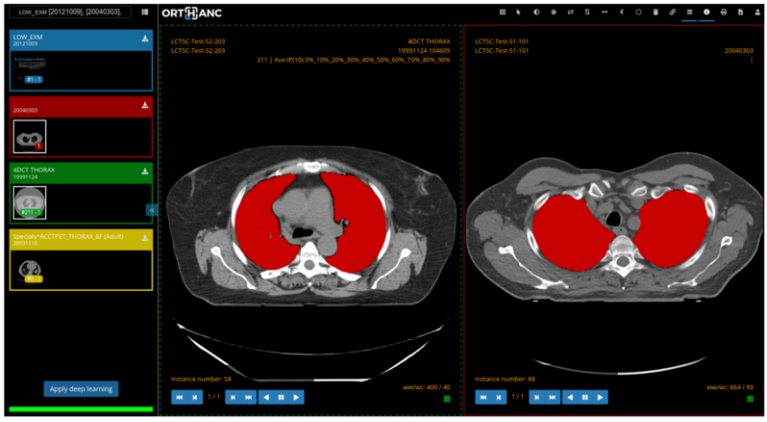

Figure 2. Segmentation using 2D U-Net models in the Stone Web viewer, applied to two test slices. The user launches the segmentation by clicking on the "*Apply deep learning*" button at the bottom of the Web interface. Then, a green progress bar below the button provides feedback about the computation by the Web worker. The segmentation mask is finally rendered as a red, semi-transparent overlay on the top of the slice.

4. Discussion

These results are promising and call for further work. For instance, this paper only considers the task of 2D segmentation using the U-Net architecture, but our library for deep learning in WebAssembly can be generalized to other tasks, such as classification, detection, or 3D segmentation. Further optimization of the CNN library is possible, by considering the specificity of the memory management of WebAssembly, by implementing more advanced techniques to compute convolutions [11], and by using WebGL to delegate parts of the computations to the GPU. Also, our CNN library could perfectly be embedded inside heavyweight viewers since it is written in plain C++.

Importantly, all the computations are done within the Web browser, so the DICOM instances never leave the PACS/viewer environment, making this deep learning solution

secure-by-design. This is notably a key advantage in the context of GDPR compliance that is implied by the MDR regulation. Furthermore, as the Stone Web viewer downloads its images using the DICOMweb standard, it is compatible with any modern PACS. Our work is released as free and open-source software in branch 3.0 of the Stone Web viewer. Interesting further research includes publishing a library of open-access models that could be used by our system, applying the system to selected clinical tasks, and evaluating the system for the training of skilled workers.

5. Conclusions

This paper demonstrates the feasibility of efficiently applying deep learning models client-side, directly by Web browsers, through a user-friendly teleradiology interface. This contribution allows to envision a scalable deployment of deep learning models, in the sense that these models could be immediately made available to many standard computers without installing any dedicated computation infrastructure or heavyweight software. It has also been shown that thanks to the WebAssembly SIMD, complex deep learning architectures such as U-Net with millions of parameters can be processed by a standard client computer in about ten seconds. This work is released as free and open-source software, in the hope that it will foster the evaluation artificial intelligence algorithms in clinical setups, and that it will promote the education of physicians, nurses, yet general audience, to artificial intelligence applied to medical imaging.

References

[1] Cheng PM, Montagnon E, Yamashita R, Pan I, Cadrin-Chênevert A, Perdigón Romero F, et al. Deep Learning: An Update for Radiologists. RadioGraphics. 2021;41(5):1427-45.
[2] Sarvamangala DR, Kulkarni RV. Convolutional Neural Networks in Medical Image Understanding: A Survey. Evolutionary Intelligence. 2022 jan;15(1):1-22.
[3] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N, Hornegger J, III WMW, Frangi AF, editors. Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015) 18th International Conf. Munich, Germany, October 5 – 9, 2015, Proceedings, Part III. vol. 9351 of LNCS. Springer; 2015. p. 234-41.
[4] Milletari F, Navab N, Ahmadi SA. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In: 4th International Conf. on 3D Vision (3DV). IEEE; 2016. p. 565-71.
[5] Paul M, Collberg C, Bambauer D. A Possible Solution for Privacy Preserving Cloud Data Storage. In: 2015 IEEE International Conference on Cloud Engineering. IEEE; 2015. p. 397-403.
[6] Kikinis R, Pieper SD, Vosburgh KG. 3D Slicer: A Platform for Subject-Specific Image Analysis, Visualization, and Clinical Support. In: Intraoperative Imaging and Image-Guided Therapy. Springer New York; 2013. p. 277-89.
[7] Jodogne S. Rendering Medical Images using WebAssembly. In: Proc. of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies (Volume 2); 2022. p. 43-51.
[8] Yang J, Veeraraghavan H, Armato SG, Farahani K, Kirby JS, Kalpathy-Kramer J, et al. Autosegmentation for thoracic radiation treatment planning: A grand challenge at AAPM 2017. Medical Physics. 2018 Sep;45(10):4568-81.
[9] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al.. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems; 2015.
[10] Chellapilla K, Puri S, Simard P. High Performance Convolutional Neural Networks for Document Processing. In: Lorette G, editor. Tenth International Workshop on Frontiers in Handwriting Recognition. La Baule (France): Université de Rennes 1; 2006. p. 1-6.
[11] Georganas E, Avancha S, Banerjee K, Kalamkar D, Henry G, Pabst H, et al. Anatomy of High-Performance Deep Learning Convolutions on SIMD Architectures. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE; 2018. p. 830-41.