Research Article

Browser-based Data Annotation, Active Learning, and Real-Time Distribution of Artificial Intelligence Models: From Tumor Tissue Microarrays to COVID-19 Radiology

Praphulla M. S. Bhawsar[1], Mustapha Abubakar[1], Marjanka K. Schmidt[2], Nicola J. Camp[3], Melissa H. Cessna[4], Máire A. Duggan[5], Montserrat García-Closas[1], Jonas S. Almeida[1]

[1]Division of Cancer Epidemiology and Genetics, National Cancer Institute, National Institutes of Health, Maryland, USA, [2]Division of Molecular Pathology, Netherlands Cancer Institute, Antoni Van Leeuwenhoek Hospital, Amsterdam, The Netherlands, [3]Huntsman Cancer Institute, University of Utah, UT 84112, USA, [4]Department of Pathology, Intermountain Healthcare Biorepository, Intermountain Healthcare, UT 84107, USA, [5]Department of Pathology and Laboratory Medicine, Cumming School of Medicine, University of Calgary, Calgary, AB T2N 1N4, Canada

Submitted: 02-Nov-2020 Revised: 05-May-2021 Accepted: 18-Jun-2021 Published: 27-Sep-2021

Abstract

Background: Artificial intelligence (AI) is fast becoming the tool of choice for scalable and reliable analysis of medical images. However, constraints in sharing medical data outside the institutional or geographical space, as well as difficulties in getting AI models and modeling platforms to work across different environments, have led to a "reproducibility crisis" in digital medicine. **Methods:** This study details the implementation of a web platform that can be used to mitigate these challenges by orchestrating a digital pathology AI pipeline, from raw data to model inference, entirely on the local machine. We discuss how this federated platform provides governed access to data by consuming the Application Program Interfaces exposed by cloud storage services, allows the addition of user-defined annotations, facilitates active learning for training models iteratively, and provides model inference computed directly in the web browser at practically zero cost. The latter is of particular relevance to clinical workflows because the code, including the AI model, travels to the user's data, which stays private to the governance domain where it was acquired. **Results:** We demonstrate that the web browser can be a means of democratizing AI and advancing data socialization in medical imaging backed by consumer-facing cloud infrastructure such as Box.com. As a case study, we test the accompanying platform end-to-end on a large dataset of digital breast cancer tissue microarray core images. We also showcase how it can be applied in contexts separate from digital pathology by applying it to a radiology dataset containing COVID-19 computed tomography images. **Conclusions:** The platform described in this report resolves the challenges to the findable, accessible, interoperable, reusable stewardship of data and AI models by integrating with cloud storage to maintain user-centric governance over the data. It also enables distributed, federated computation for AI inference over those data and proves the viability of client-side AI in medical imaging.

Availability: The open-source application is publicly available at https://episphere.github.io/path, with a short video demonstration at https://youtu.be/z59jToy2TxE.

Keywords: Artificial intelligence, client-side artificial intelligence, consumer-facing governance, TensorFlowJS, web computing

Introduction

In the past few years, artificial intelligence (AI) techniques, particularly supervised deep learning, have gained wide adoption for medical image analysis. Applications of these methods have been shown to be effective for multiple tasks including image segmentation,[1] registration,[2] disease diagnosis,[3] and others, with state-of-the-art models demonstrating performance on par with or even better than human experts.[4] However, these results have also brought to the fore a concern about their reproducibility[5] – many

Access this article online

Quick Response Code:

Website: www.jpathinformatics.org

DOI: 10.4103/jpi.jpi_100_20

Address for correspondence: Praphulla M. S. Bhawsar, 1603 E Jefferson St., Rockville, MD 20852, USA. E-mail: bhawsarpm@nih.gov

How to cite this article: Bhawsar PM, Abubakar M, Schmidt MK, Camp NJ, Cessna MH, Duggan MA, *et al*. Browser-based data annotation, active learning, and real-time distribution of artificial intelligence models: From tumor tissue microarrays to COVID-19 radiology. J Pathol Inform 2021;12:38.

Available FREE in open access from: http://www.jpathinformatics.org/text.asp?2021/12/1/38/326825

of these models are developed in siloed environments using specific software libraries, and because reproducing the exact environment is not always convenient or possible, their reusability and performance are considerably compromised by their lack of portability across platforms. As a consequence, the full value of AI applications to Digital Medical Imaging represents a particular challenge to the findable, accessible, interoperable, reusable (FAIR) stewardship[6] of scientific data in conventional computational platforms, be it on-premises or in centralized cloud architecture.

Another major barrier to facilitating the FAIRness of data and models is the limited shareability of medical data outside the clinical, organizational, or geographical domain where it is generated.[7] Model training typically requires the dataset to be present in a location that is directly accessible to the machine on which the training is to be performed, most often as a locally downloaded copy. Sometimes the data are unlabeled, a scenario where the machine that provides the annotation service needs to be able to access the data as well. Moving files across computers to achieve these individual tasks is often unviable because of tight privacy and compliance requirements when dealing with medical data. Logistic barriers can also be a major obstacle, especially with large datasets common in Pathology and Genomics sequencing laboratories. Finally, in cases where the data are stored at a central on-premise location, setting up proper governance and access management is rarely straightforward. The cloud has emerged to resolve many of these challenges, but the software tools conventionally used by researchers and clinicians alike have not kept pace with these new computational architectures. Specifically, it remains difficult to integrate these tools with centralized cloud storage where (a) the governance of the data is inherently federated; and (b) the annotation process needed for AI training requires its distribution to domain experts.

A number of efforts have been pursued to address these obstacles. For example, the DeepInfer framework by Mehrtash *et al*.[8] uses a Docker engine to standardize the evaluation environment and runs various deep learning models on the client machine. TOMAAT by Milletari *et al*.[9] facilitates deployment of trained models to the cloud, making them accessible as a remote service. These frameworks require a client like the 3D Slicer[10] visualization software to communicate with the model server and obtain inferences. More recently, Philbrick *et al*. developed RIL-Contour, a software application to accelerate image annotation for and with deep learning.[11] Although RIL-Contour provides a one-stop solution for annotation and modeling, it is a standalone application that requires setting up multiple external dependencies, something that clinically oriented users may find hard to do. Sedghi *et al*. took a browser-based approach for the user interface with Tesseract-MI.[12] However, it too uses a Docker engine that would need to be deployed – and maintained – for use across machines. These solutions approach the web browser as just a user interface rather than as a means for distributed web computing as proposed here. They also do not integrate with cloud storage or provide user-centric governance that largely drives cloud adoption in the first place. Instead, they assume the local availability of data as a system resource decoupled from user governance. The implications of centralized designs for applying AI models to sensitive data are just as problematic: the data have to be sent to the modeling engine for inference, leaving only two possibilities, one unscalable, the other insecure. Respectively, either (a) the engine is made available locally in a specialized computational environment or (b) sensitive data are entrusted to a central AI engine where inference is opaque to the users. In conclusion, there is an urgent need for platforms that both enable the migration of the AI inference code to where access to the sensitive data is available (e.g., a patient portal and/or a pathologist's workspace), and incorporate the type of consumer-facing cloud storage associated with the "Box model" as put forward by Bremer *et al*.,[13] where security and governance are delegated to and taken care of by the cloud storage service that hosts the data, and the client only need consume the Application Program Interfaces (APIs) exposed by the service to take advantage of this delegation.

In this study, we describe the implementation of a web platform that meets these challenges of distribution and FAIRness by providing tools for data annotation and model inference entirely on the browser at zero cost and with no installation (i.e., "zero-footprint"). Although local files can be used with the application for model training and inference, the study focuses primarily on how cloud storage services have been integrated into the platform, with Box.com as a specific example. Specifically, we aim to demonstrate the practical implementation of a ready-to-use hosted web application that has no server-side computing elements, requires no deployment, and can be accessed from any device with a web browser through a Uniform Resource Locator (URL). The accompanying application extends the "Box model" to provide researchers having no machine learning experience with the tools to train, use, and share AI models.

Methods

The web browser has evolved rapidly to keep pace with the development of the World Wide Web as a global computational space. Browser engines today support the full-stack of computational features, from persistent storage and background processes to in-browser web servers (service workers). Because of the ubiquitous nature of web computing, these are now optimized to the point that they are emerging as the preferred solution for distributed software ecosystems. The software entities that take full advantage of these ecosystems are called web applications, with the more device-agnostic ones often designated as Progressive Web Apps (PWAs). Like conventional websites, PWAs are accessible through a dereferenceable URL that one can visit on a web browser. Although web applications generally contain distinct client and server-side components, they do in fact allow for the full stack application logic to be executed entirely on the client as

well. The platform we describe here is an example of the latter, a completely client-side PWA. This approach guarantees that the data never leave the user's machine, that the code running on the browser is completely transparent, and that any network interaction can be readily examined through the developer toolbox packaged with all modern web browsers. The PWA leverages browser functionalities such as the HTML5 Canvas and Web Workers so as to provide an intuitive and smooth user experience while being capable of performing heavy computations in the background.

The practice of using cloud storage for data management has recently gained traction in digital pathology and biomedicine as a whole. Architecturally, these services act as cloud-based middleware providing a FAIR API ecosystem. In our case, the APIs securely exposed by Box enable our application to automatically make full use of their offerings (such as viewers for nonweb standard image formats like DICOM and TIFF) as they become available, while also allowing us to focus on developing features and letting Box ensure authorized access to the data. It cannot be overemphasized how much of the developer effort and compliance load is lifted from the development exercise by relying on consumer-facing/governed API ecosystems.

The data we work with in the described PWA are currently limited exclusively to medical images. Figure 1 shows the architecture of the platform. Below, we outline how it facilitates each aspect of the medical AI workflow. The application code is open-source and publicly available at: https://github.com/episphere/path.

Authentication and governance

Consumer-facing cloud storage services depend critically on the robustness and security of their authentication and authorization mechanisms. They also go to great lengths to integrate these mechanisms with institutional authentication systems, perform

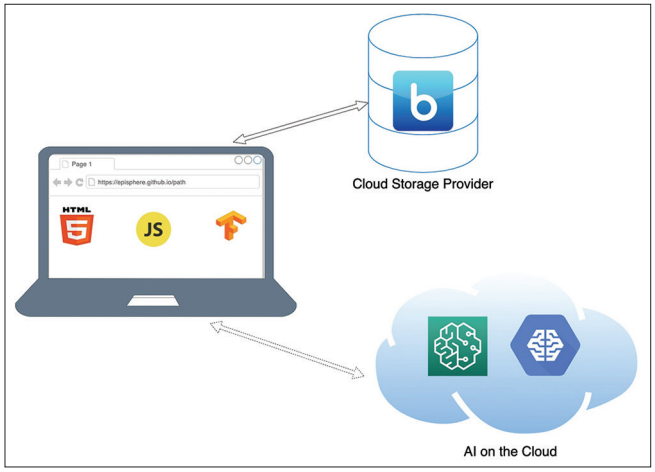

Figure 1: Platform architecture. The browser runs the web application which acts as the connector between the cloud storage provider and the AI service on the cloud. Communication happens with the cloud provider only if model training is to be performed on the cloud or if a deployed model needs to be used for running inferences on the data in the application

regulatory auditing, and provide security oversight. As a consequence, these mechanisms become available for use by applications registered with the cloud service, the so-called "Box Apps" in the case of Box. These applications can be thought of as client-side extensions to the cloud storage service, but without programmatic constraints to integrating additional services or components, such as AI-powered analytics. Our platform performs user authentication and authorization through such an application registered with Box. This enables researchers at other institutions to directly use the integration with their institutions' own Single Sign-On systems. To that end, we employ the OAuth2.0 protocol[14] where the user logs in to their Box account, and Box then provides our platform with a token that identifies the user and allows them to access their data in Box through the platform. OAuth2.0 ensures that authentication is handled completely by the service, not the platform, so there is no registration involved and the user's credentials are never exchanged or compromised. Essentially, the OAuth2.0 process generates a token that allows the application code to act on behalf of the user under scoped authorization where only content owned by or shared with that user is accessible to them.

Storage services give users the option to share files with others and define the permissions those users would have over the shared data. This user-defined governance is the basis for both the trustworthiness and distributed robustness of consumer-facing Box systems.[13] Access and permission management is taken care of by Box outside the scope of the application code; any update in the access arrangements is propagated everywhere instantly by the service. Consequently, because our application only acts as an intermediary between Box and the client, once a content sharing is enacted in the environment of the storage service, the shared files automatically become accessible to the user in our platform, in real time. In summary, decoupling governance by delegating it to the storage provider eliminates the need for the platform to maintain complex data management architectures on its own. It also facilitates collaboration, as researchers wishing to work on the same dataset need only set shared access to the relevant folders and files under data access policies they can establish and oversee on their own, rather than through the mediation of "trusted brokers."

Image viewing and operations

Once authentication is successful within the application, the user has access to all the files in their cloud storage account directly from the web application interface. They can navigate through the files and select images to view in the application. Supported image formats include JPEG, PNG, and TIFF. The images are viewed using an HTML5 Canvas; it should be noted here that advances in Web technologies, such as the native Canvas become available immediately to the PWA without it having to explicitly maintain dependencies with specialized external libraries, as is the case for more conventional systems in use for Digital Pathology and other digital imaging applications. As a consequence, advanced image processing operations, potentially involving the use of

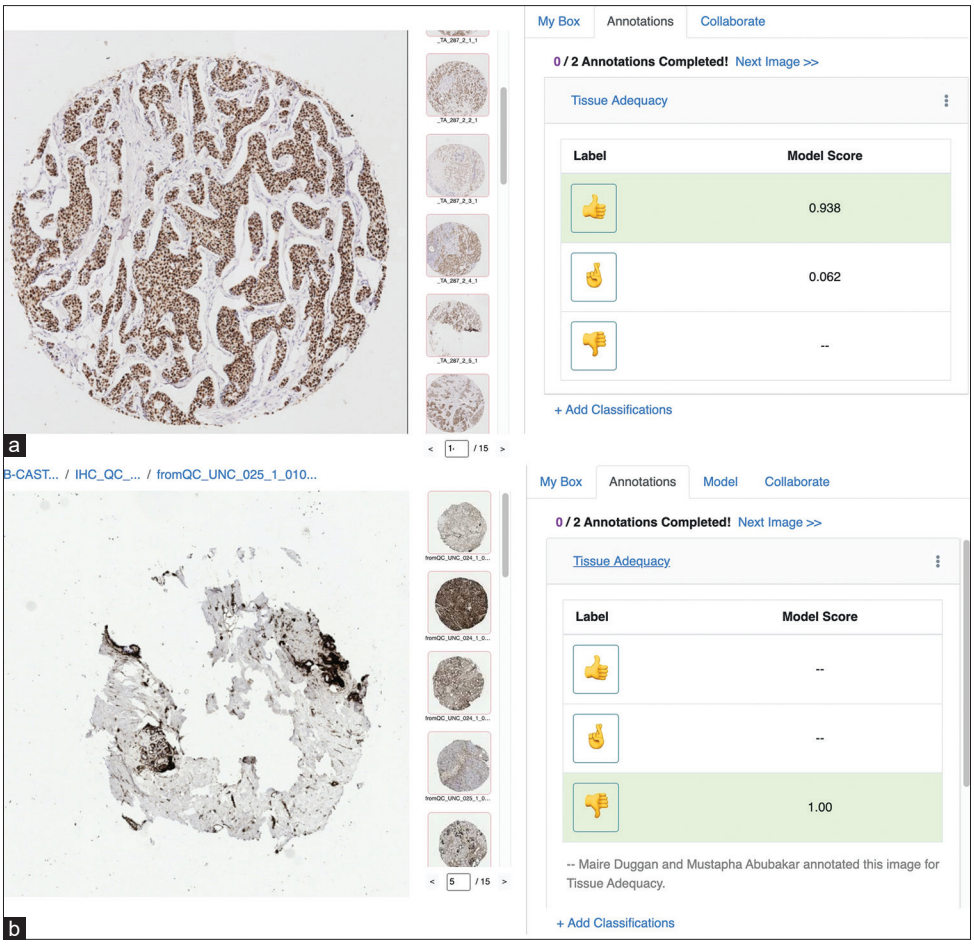

Figure 2: The web application with Tissue Adequacy model prediction scores for different Breast Cancer TMA core images. The "My Box" tab is the file browser that allows the user to navigate through their Box account within the application. The "Annotations" tab lets them define classifications and add labels or comments to an image; the model prediction score is shown immediately adjacent to where the user would assign the label. The thumbs-up, crossed fingers, and thumbs down buttons denote the Optimal, Suboptimal, and Unsatisfactory labels, respectively, and are what the pathologists used to classify the image in one of the three classes. The model score is the activation value of the final layer of the trained neural network and is a continuous value normalized between 0 and 1 – the higher the score, the greater the likelihood of the prediction being true. (a) An example of a core with optimal Tissue Adequacy, with a classification score of 0.938 to the Optimal class by the model. (b) A core showing inadequate tissue, and the model agrees with high "confidence" as can be seen from the classification score of 1 for the Unsatisfactory class

GPUs, such as zooming and foreground image segmentation, can be efficiently performed using the tools programmed into the application itself.

Data annotation

Adding annotations to raw imaging data is the first step toward the supervised training of a model. These annotations, however, need to be managed in such a way that the association between them and the data is preserved within the governance scope of each user. Keeping these annotations in external files or databases require a dedicated entity to manage those associations. To solve this problem, the DICOM standard prescribes embedding annotations as metadata within the individual files themselves. While this does make the association implicit, it is not available to native image formats. Adding to the challenge, developing a "shadow" metadata system would be a major engineering effort, comparable to what went into developing the DICOM format.

Unsurprisingly, the metadata management functionality is a key feature of cloud-based storage providers like Box, where it is included as a service with its own dedicated APIs alongside version control, backup, and other core features. Again, the application described here benefits from these advanced metadata management APIs as a direct result of integrating cloud storage. The way Box enables the addition of metadata on a file is not by embedding it inside the file itself but by storing it in a separate managed database along with the association, in essence creating an appearance of it being actually present on the file. Because storage and association are taken care of by the cloud service, we can adhere to the DICOM standard in spirit by saving annotations to this "pseudo-metadata." One could argue that this solution goes beyond the DICOMs metadata embedding, since the original files stay the same, and it is up to the user to decide whether the annotations travel with the file if it needs to be moved elsewhere.

The annotation solution described above has the added benefit of the metadata not having to conform to file format restrictions – in fact, cloud services allow any textual string to be applied as metadata. Our application uses this to allow the user to define their own annotations and/or labels for the entire dataset. These definitions are stored in a configuration file on the cloud, and once created, they are accessible to anyone with permission to access the dataset files. Any annotations or comments made on an image are then saved as this "pseudo-metadata" on the cloud in the JavaScript Object Notation (JSON) format. Ultimately, these annotations are the mechanism by which the targets and inputs for AI training are configured. Having the annotation visually close to the file also makes it much easier to create the training dataset: no querying needs to happen to fetch specific annotations from an external location (a process that used to be described as "database enrichment"), since the Box platform maintains that association index by default. The annotation manifest serves as a reference to the current state of the dataset as well as for AI model training on the cloud.

Model training – from server-side AutoML to client-side TensorFlow.js

Most commonly used machine learning libraries allow for cloud-based deployment of trained models, exposing them through APIs over HTTP. More recently, frameworks such as ConvNetJS,[15] TensorFlow.js,[16] and ONNX.js[17] have proven the viability of using full-fledged neural networks directly from the web browser instead. Our application uses TensorFlow.js as the framework of choice, primarily because it supports WebGL for running complex matrix computations on the machine's GPU (if available). It is still true, however, that training models with millions of parameters from scratch might not be feasible on the local machine; it is also nontrivial to design the best convolutional neural network (CNN) topology for a given problem. To address this, our platform can be used with both in-browser and cloud-based model training approaches, with the option to use Google's Cloud AutoML service.[18] The latter is a managed neural network modeling service that runs various automated algorithms to find model parameters best suited for a given dataset, trains a CNN on the provided imaging data, and outputs a performant model all on its own, effectively removing any barrier for researchers with limited machine learning experience from getting to an acceptable model. This is yet another reason why the platform uses TensorFlow.js, given its out-of-the-box support for models trained through Cloud AutoML. The aforementioned manifest created from the annotated files is structured in such a way as to be usable with Cloud AutoML straightaway.

Model inference

Once training is complete, the model can either be downloaded or deployed on the cloud. While Cloud AutoML provides an option for cloud deployment immediately after training, this approach is fairly expensive. This is one of the main reasons why we opted to load the model locally through our platform instead, given that it can be used in-browser at zero cost and with an improvement in performance (no remote communication needed, see 3.1 in Results). Furthermore, the trained model can be stored in the same location in Box as the dataset, effectively resulting in the access permissions of the dataset propagating automatically to the model and sharing it instantly with all users having access to the initial folder without breach of governance. This was not anticipated by the original design but ended up being a major feature of the solution proposed: because multiple users have access to the same trained models, and these can be used to classify images stored under their individual governance, this amounts to federated AI classification that scales with the number of users.

The files that represent a TensorFlow.js model include a JSON file that contains the model topology, along with one or more binary files containing the weights at each node. These files can be used with the TensorFlow.js library to load the model in any environment that can run JavaScript, such as the browser. Our platform loads the model inside a Web Worker and uses it to obtain inferences in the background. Intriguingly, from a provenance perspective, these inferences behave as annotations in and of themselves and are therefore saved back to Box as additional "pseudo-metadata" to be used for future analyses. To help incorporate model predictions into the user's analysis, we display the predictions in the same place where the user would add their annotations.

Real-time collaboration and extension by code injection

The web application accompanying this report allows users to collaborate with each other on the platform in real time, using the TogetherJS library over peer-to-peer WebRTC to facilitate screen-sharing and communication over audio or chat. Users can also choose to inject their own code within the application context simply by using the URL hash parameter "extModules" and specifying the URL to the script file containing the code they wish to run. Because there are no server-side components, there is no risk of the injected code making any unauthorized changes.

Results

Digital pathology case study: Analyzing breast cancer tissue microarray images

Tissue microarrays (TMAs) have gained rapid popularity over the last decade for histological analysis in digital pathology. The technique has been especially impactful in the field of oncology, such as for biomarker discovery in clinical studies and large-scale epidemiological studies investigating etiological heterogeneity by cancer subtypes.[19] An important challenge for large-scale studies involving unsupervised image analysis of individual tissue core images is the institution of rapid preanalytical quality control (QC) checks, such as for adequacy of the tissue core (e.g., adequate amount of tumor) and biomarker staining for scoring (e.g., percent cells stained or intensity of biomarker staining).[20,21] Although both classes of parameters can be assessed visually by a pathologist, this approach is not readily scalable to thousands of cores from

TMAs, hence the need for automation. To test the utility of our platform on this problem, we employed a dataset of breast TMA cores from the B-CAST project[22] stored in our institutional Box instance. These TMAs were collated from several studies and have been stained for a variety of immunohistochemical biomarkers, making this dataset diverse enough to assess the usability of the application in a generic digital pathology context. The TMA core images themselves range from sizes of 40KB to 35MB and are in the JPEG, PNG, and TIFF file formats.

The objective of this experiment was to see if the application could be used to first have pathologists add labels for Tissue Adequacy – denoting amount of visible tissue present in the image – to a smaller dataset of examples, then train a deep CNN on these images and subsequently obtain in-browser inferences from it on a larger dataset. To start off, we selected 110 images from the Amsterdam Breast Cancer Study (ABCS) and the Utah Breast Cancer Study as our training dataset, with varying degrees of tissue adequacy across the chosen images. We then created and defined categorical labels for the Tissue Adequacy class, namely Optimal, Suboptimal, and Unsatisfactory, so as to make it easier and faster to add labels to the images. We then used the interface to subjectively assign one of the three labels to each image. It should be noted that the data stayed under controlled access in a shared folder in our Box accounts this entire time, with all annotations being stored in the form of "pseudo-metadata" as described before. We used this labeled dataset to train a CNN using the Cloud AutoML service. Although model training could just as well have been completed on the local machine, delegating it to a managed AI modeling service essentially allowed us to take full advantage of the advanced hyperparameter tuning functionality provided by Google as a managed service. To test the trained model in different environments, it was deployed on the cloud as well as downloaded to the same folder in Box that housed the training dataset images, thus giving anyone authorized to view those images access to the model as well. The downloaded model was then consumed by the application and ran in the same browser context [see Figure. 2].

To evaluate performance, both model deployments were used to obtain inferences for over 5000 images from the ABCS study, none of which were present in the training dataset. This took the model deployed on the cloud about 5 h with parallelized asynchronous requests; comparatively, it took the client-side model <2 h on a commodity computer without a discrete GPU and without any parallelization whatsoever. Despite the considerably small training dataset size, the model was surprisingly accurate in classifying TMA images across formats, sizes, and stains for tissue adequacy. The receiver operating characteristic (ROC) curves in Figure 3 show the excellent performance of the model as a binary classifier of tissue adequacy as satisfactory and unsatisfactory based on two scenarios, one where satisfactory includes the suboptimal class with the optimal and another where suboptimal is excluded [see legend of Figure 3].

COVID-19 case study: Identifying COVID-19 positive cases from chest X-ray images

The need for an urgent response to the COVID-19 pandemic has led to several collaborative research efforts around the world with unprecedented sharing of data in the public domain. The availability of FAIR data in this regard has significantly motivated the use of AI and deep learning to complement these initiatives. As a case study to examine the utility of the platform in a nonpathological context, we applied it on a public dataset compiled by Subramanian *et al*.[23] containing labeled chest computed tomography images of unaffected people and patients of COVID-19 and viral pneumonia. The dataset comprises 296 JPEG images in total, with 92 observations of COVID-19 cases and 102 each of unaffected people and pneumonia patients. We imported these images into our institutional Box instance and created a class definition through the platform to classify each image based on the dataset labels. Once classification was complete, we generated the manifest for model training and used Cloud AutoML over the entire dataset. Training took a little over an hour and the resulting model showed a validation AUC of about 0.835. This model was then used in-browser to obtain inferences on a different labeled dataset[24] containing 388 images stratified similarly as the original data [see Figure. 4]. Using the predictions made on the second dataset, we identified images that the model could not classify with a high enough score or those that were classified incorrectly. These images were then added to the training data, and a new model was trained on a randomly shuffled dataset with the same number of images as before. This second version generalized markedly better with a validation AUC of 0.88, providing an illustration for how our application can be used for active learning, in which predictions by the AI model on new data are subsequently checked and re-classified by experts, with the extended annotated results provided back for model retraining. Figure 5 shows the ROC curves for the model before and after active learning.

Discussion

Although it is becoming increasingly popular to deploy models on the cloud, this approach can be costly. In our case studies, we tried using the cloud deployment option offered by Cloud AutoML, and we were billed about USD 30 per day to deploy and use the TMA model on the Google Cloud Platform. This is especially noticeable given that the PWA was able to load the same model in the browser to make predictions at zero cost (apart from the electricity usage of the local machine). In addition, the remotely deployed model was also constrained by the fact that each file needed to be sent to it through HTTP for analysis, which is undesirable for medical images both for privacy reasons and because sending sizeable files over the network is quite slow. With the two factors put together, we found that prediction time for the in-browser model was around 150 ms on average per image, while the remote model took over 5 s per image mainly due to network overhead. We attempted to reduce costs on the cloud by deploying the models through a serverless function instead of a separate virtual instance. While

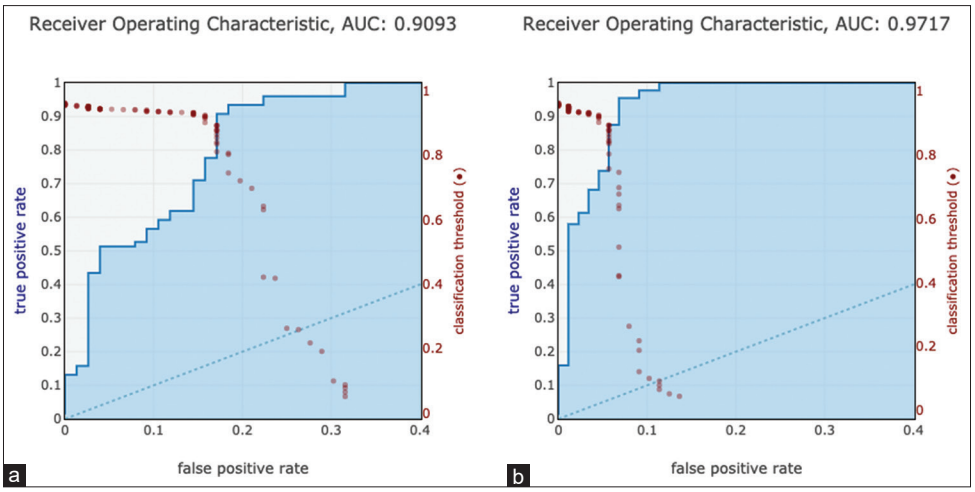

Figure 3: The receiver operating characteristic (ROC) curves for the Tissue Adequacy mode considering (a) only the optimal classification as the true positive, and (b) both the optimal and suboptimal classifications combined as the true positive. The curves suggest that the model can be reliably used for QC on the basis of tissue presence. The two y-axes provided for each receiver operating characteristic plot denote the conventional true positive rate on the left (in blue) and the corresponding threshold values on the right (in red). The latter is provided to allow the graphic determination of what AI model scores [from Figure 2] will correspond to different proportions of true and false positives

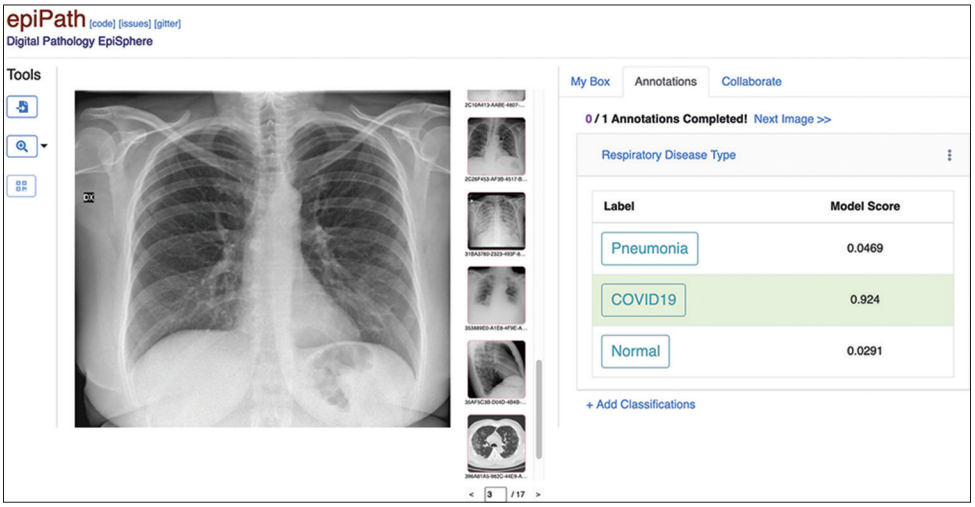

Figure 4: The web application showing the model prediction for a chest X-ray image of a COVID-19 patient. Note that the annotation classes from the breast cancer tissue microarray dataset were not carried over – the annotation classes are defined by the user and are specific to the dataset

this option is less expensive, file transfer is still necessary, and there is a significant increase in average prediction time because of the extra "warm-up" period of an inactive serverless container. As a result, any difference in computational performance between the in-browser and cloud-based approaches effectively favors the former. The in-browser AI deployment model is ranked higher in terms of availability and explainability as well. While this was predictable in principle, the 30-fold decrease in latency, zero footprint, no data transfer, and virtually no costs of deployment puts to rest any doubts about the practicality of the proposed in-browser AI solution.

Training a model on a third-party cloud provider is possible only if the storage service that houses the data can be connected to and used by the modeling service as a data source. In the above case studies, this was a definite limitation for our platform. Since Box does not tie into Google Cloud Storage, which is where Cloud AutoML requires training data to be placed, data transfer is unavoidable in this training system. This challenge to integrating advanced AI hyperparameterization engines that do not require localized data is not particular to our platform, however. A number of initiatives both in academia and industry have recognized this limitation and are advancing federated learning[25] as an alternative. In this model, training takes place in a distributed manner by taking advantage of idle compute resources at the locations ("at the edge") where the benefits of model predictions are ultimately realized but without requiring shared access to the training data. This is a recurring theme in this report: once federated learning is part of the cloud ecosystem, i.e., once its APIs allow for the distribution of computation to client applications, the

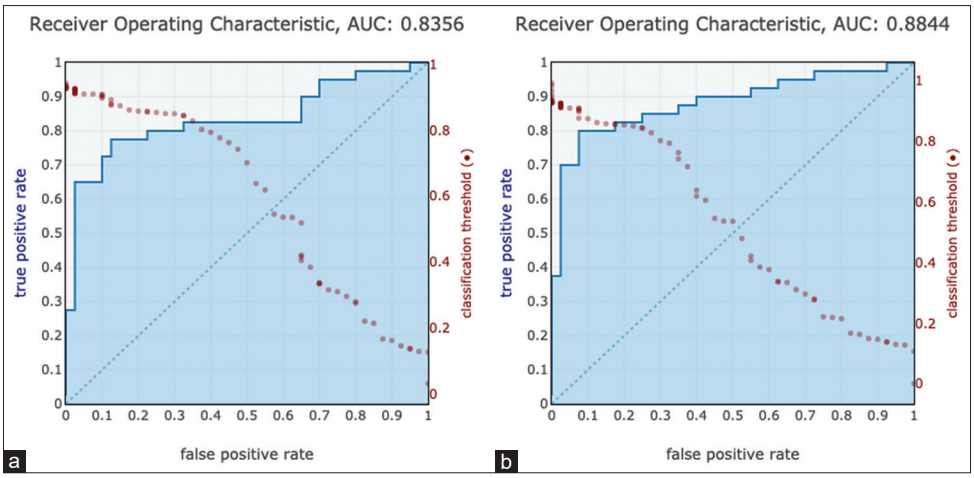

Figure 5: The receiver operating characteristic curves for the COVID-19 chest computed tomography model considering the COVID-19 classification as the true positive (a) trained on just the initial dataset and (b) after retraining the model through active learning. The two Y-axes are the same as in Figure 3, with the left axis showing the true positive rate and the right axis denoting the threshold value

approach reported here will be particularly well positioned to make full use of that distribution instead of having to develop it independently.

The model classifications and the corresponding scores, as mentioned previously, are stored with the file in Box as "pseudo-metadata." This gives users a significant computational bonus because once one person has used a model to classify an image, others automatically get that classification for free by simply fetching this stored metadata from Box, instead of having to run the image through the model themselves. Consequently, despite the fact that the models run locally, the application enables all users to benefit from the combined computational resources of everyone using the models on the shared dataset, essentially making this a distributed, federated computing platform. Moreover, storing the predictions as annotations that can be accessed on the platform also facilitates incrementally selective analyses. For instance, researchers can use the model predictions for QC to guide decisions on data inclusion or exclusion based on prespecified quality metrics. In the TMA case study, we could choose to only keep images that the Tissue Adequacy model classifies to be optimal, avoiding wasted analytical effort – and added analytical error – with inadequately visible tissue. This would significantly reduce the number of images that would otherwise have to be seen and manually discarded by the pathologist. Moreover, having the models in a local environment allows for rapid iterative model development and active learning, as evidenced by the COVID-19 case study.

An added bonus of developing this as a completely client-side web application is that no setup, downloads, or installation whatsoever are required. All that is needed is for the data to be put into a cloud storage system supporting "Box-like" APIs with consumer-facing governance. This allows everyone with or without computing expertise to use it collaboratively and thus accelerates the process of creating standardized datasets and subsequent AI modeling. This combination of easy access and faster analysis was observed to be especially helpful when trying to analyze a rapidly evolving problem such as the ongoing COVID-19 outbreak. While this case study was only conducted as a demonstration of the flexibility of a system originally conceived for digital pathology, it shows that the proposed solution can be effectively redirected to other image analysis scenarios in real time. When rendered as a PWA, the platform described here is accessible on just about any device with a web browser, so medical staff on the field could make use of models trained by experts elsewhere, potentially without requiring network access when analyzing local data. Moreover, as another direct result of relying on the APIs supporting consumer-facing governance, this platform gives the researcher the ability to fine tune their models through active learning as and when more data becomes available, evolving the classifier toward increasingly effective and reliable analyses.

Although this work focused primarily on classification, our approach can also be applied to other image analyses involving supervised learning, such as segmentation, registration, and object detection. Furthermore, a recent effort by Bremer *et al*.[26] toward web-based interoperation with whole slide imaging data makes it possible to extend our tool to that domain. This is directly enabled by the authorized API-based approach described in this report in that it can engage whole slides images stored on the cloud one tile at a time, in a manner consistent with the security and resource constraints of the web browser. The resulting effect of retrieving individual tiles with HTTP range requests can be assessed, for example, at https://episphere.github.io/svs

Conclusions

Web applications are rapidly becoming the preferred way of delivering software solutions for a variety of applications. This reflects their ubiquity and inherent extensibility by code injection onto a safe sandbox environment in the browser. These features are in drastic contrast with both conventional desktop and HPC software. The accompanying open-source platform showcases how digital pathology, radiology, and medical imaging in general

can take advantage of these trends by providing researchers with the ability to perform the entire image analysis workflow without leaving the browser. The application of the platform to the breast cancer TMA case study demonstrates its viability and utility for research. The COVID-19 case study illustrates how it can also be used for rapid dataset formulation, AI modeling, and active learning, potentially in real-time. Together, these case studies show how the problems of data governance, security, and annotation management are best solved jointly by leveraging the extensive capabilities of consumer-facing cloud storage services like Box. The outlined approach creates zero-footprint, zero-cost model predictions without requiring transfer of possibly sensitive data. These features have the potential to deflect regulatory deterrents to AI deployment by advancing a model in which the code moves to where the data are governed, rather than the latter being transferred to third-party execution environments. This is just as important for regulatory models that can only be addressed with user-centric governance as it is for data socialization schemes advanced by community-driven open source projects. Moreover, because the model can be tested in an environment as ubiquitous as the web browser merely by sharing it through cloud storage, reproducibility becomes much more straightforward for those generating the data. Finally, the browser as a full-stack computational platform is now adding new native APIs enabling advanced features previously available only in specialized environments. In the context of applying federated AI to Digital Pathology, the adoption of the WebGPU standard[27] as well as of frameworks for distributed edge computing such as those approached by QMachine[28] suggest that Web Computing backed by consumer-facing Box-like cloud middle-layers may be ideally suited for deep learning applications to computational pathology.

Financial support and sponsorship

Nil.

Conflicts of interest

There are no conflicts of interest.

References

1. Hesamian MH, Jia W, He X, Kennedy P. Deep learning techniques for medical image segmentation: Achievements and challenges. J Digit Imaging 2019;32:582-96.
2. Lewis K, Rost NS, Guttag J, Dalca AV. Fast learning-based registration of sparse 3D clinical images. In: Proceedings of the ACM Conference on Health, Inference, and Learning (CHIL '20). New York, NY, USA: Association for Computing Machinery; 2020. p. 90-8.
3. Cheng JZ, Ni D, Chou YH, Qin J, Tiu CM, Chang YC, *et al*. Computer-aided diagnosis with deep learning architecture: Applications to breast lesions in US images and pulmonary nodules in CT scans. Sci Rep 2016;6:24454.
4. Litjens G, Kooi T, Bejnordi BE, Setio AA, Ciompi F, Ghafoorian M, *et al*. A survey on deep learning in medical image analysis. In: Medical Image Analysis. Elsevier; 2017. Available from: https://doi.org/10.1016/j.media.2017.07.005. [Accessed on 28 Aug 2021]
5. Carter RE, Attia ZI, Lopez-Jimenez F, Friedman PA. Pragmatic considerations for fostering reproducible research in artificial intelligence. NPJ Digit Med 2019;2:42.
6. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, *et al.* The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 2016;3:160018.
7. Kalkman S, Mostert M, Gerlinger C, van Delden JJM, van Thiel GJ. Responsible data sharing in international health research: A systematic review of principles and norms. BMC Med Ethics 2019;20:21.
8. Mehrtash A, Pesteie M, Hetherington J, Behringer PA, Kapur T, Wells WM, *et al*. DeepInfer: Open-source deep learning deployment toolkit for image-guided therapy. In: Medical Imaging 2017: Image-Guided Procedures, Robotic Interventions, and Modeling. SPIE; 2017. Available from: https://doi.org/10.1117/12.2256011. [Accessed on 28 Aug 2021]
9. Milletari F, Frei J, Ahmadi S. TOMAAT: Volumetric Medical Image Analysis as a Cloud Service; March, 2018. Available from: https://arxiv.org/abs/1803.06784v2. [Accessed on 28 Aug 2021]
10. Pieper S, Halle M, Kikinis R. 3D Slicer. In: 2004 2nd IEEE International Symposium on Biomedical Imaging: Macro to Nano. IEEE; 2004. Available from: https://doi.org/10.1109/isbi.2004.1398617. [Accessed on 28 Aug 2021]
11. Philbrick KA, Weston AD, Akkus Z, Kline TL, Korfiatis P, Sakinis T, *et al*. RIL-contour: A medical imaging dataset annotation tool for and with deep learning. J Digit Imaging 2019;32:571-81.
12. Sedghi A, Hamidi S, Mehrtash A, Ziegler E, Tempany C, Pieper S, *et al*. Tesseract-Medical Imaging: Open-Source Browser-Based Platform for Artificial Intelligence Deployment in Medical Imaging. SPIE; 2019. Available from: https://doi.org/10.1117/12.2513004. [Accessed on 28 Aug 2021]
13. Bremer E, Kurc T, Gao Y, Saltz J, Almeida JS. Safe "cloudification" of large images through picker APIs. In: AMIA. Annual Symposium Proceedings. AMIA Symposium; 2016.
14. The OAuth2.0 Authorization Framework. Available from: https://tools.ietf.org/html/rfc6749. [Accessed on 28 Aug 2021]
15. Andrej Karpathy. ConvNetJS: Deep Learning in your Browser. Available from: https://cs.stanford.edu/people/karpathy/convnetjs. [Accessed on 28 Aug 2021]
16. Smilkov D, Thorat N, Assogba Y, Yuan A, Kreeger N, Yu P. *et al*. TensorFlow.Js: Machine Learning for the Web and Beyond; 2019. Google Research. Available from: https://arxiv.org/abs/1901.05350. [Accessed on 28 Aug 2021]
17. ONNX.js: Run ONNX Models Using JavaScript. Available from: https://github.com/Microsoft/onnxjs. [Accessed on 28 Aug 2021]
18. Google Cloud AutoML: Train High-Quality Custom Machine Learning Models with Minimal Effort and Machine Learning Expertise. Available from: https://cloud.google.com/automl.
19. Broeks A, Schmidt MK, Sherman ME, Couch FJ, Hopper JL, Dite GS, *et al*. Low penetrance breast cancer susceptibility loci are associated with specific breast tumor subtypes: Findings from the Breast Cancer Association Consortium. Hum Mol Genet 2011;20:3289-303.
20. Howat WJ, Blows FM, Provenzano E, Brook MN, Morris L, Gazinska P, *et al*. Performance of automated scoring of ER, PR, HER2, CK5/6 and EGFR in breast cancer tissue microarrays in the Breast Cancer Association Consortium. J Pathol Clin Res 2015;1:18-32.
21. Sherman ME, Howatt W, Blows FM, Pharoah P, Hewitt SM, Garcia-Closas M. Molecular pathology in epidemiologic studies: A primer on key considerations. Cancer Epidemiol Biomarkers Prev 2010;19:966-72.
22. The Breast CAncer STratification (B-CAST) Project. Available from: http://www.b-cast.eu/. [Accessed on 28 Aug 2021]
23. Subramanian T, *et al*. Sansten AI Labs COVID-19 X-Ray Image Dataset. Available from: https://lnkd.in/geyyd3V. [Accessed on 16 Jun 2020]
24. COVID19 High Quality Images: 120 High Quality Images for COVID19, Viral Pneumonia and Normal. Available from: https://www.kaggle.com/theroyakash/covid19/data. [Accessed on 28 Aug 2021]
25. Brendan McMahan and Daniel Ramage. Federated Learning: Collaborative Machine Learning without Centralized Training Data. Available from: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. [Accessed on 28 Aug 2021]
26. Bremer E, Saltz J, Almeida JS. ImageBox 2 – Efficient and rapid access of image tiles from whole-slide images using serverless HTTP range requests. J Pathol Inform 2020;11:29.
27. The WebGPU Specification, GPU for the Web Community Group. Available from: https://gpuweb.github.io/gpuweb. [Accessed on 28 Aug 2021]
28. Wilkinson SR, Almeida JS. QMachine: Commodity supercomputing in web browsers. BMC Bioinformatics 2014;15:176.