Asynchronous Federated Learning for Web-Based OCT Image Analysis

Hasan Md Tusfiqur Alam[a], **Tim Maurer**[b, c], **Abdulrahman Mohamed Selim**[a], **Matthias Eiletz**[c], **Michael Barz**[a,d], **Daniel Sonntag**[a,d]
[a]German Research Center for Artificial Intelligence (DFKI) Saarbrücken, Germany
[b]Saarland University, Saarbrücken, Germany
[c]ProOmea GmbH, Wangen, Germany
[d]Oldenburg University, Oldenburg, Germany

Author Accepted Manuscript (AAM).
This is the peer-reviewed accepted version of the following article:
Hasan Md Tusfiqur Alam, Tim Maurer, Abdulrahman Mohamed Selim, Matthias Eiletz, Michael Barz, Daniel Sonntag,
Asynchronous federated learning for web-based OCT image analysis, Journal of Medical Imaging, 13(1), 014501 (January 2026).
The final Version of Record is available at https://doi.org/10.1117/1.JMI.13.1.014501.
Code: https://github.com/tmaurer42/octdl-training.

Abstract.

Purpose: Centralized machine learning often struggles with limited data access and expert involvement. This study investigates decentralized approaches that preserve data privacy while enabling collaborative model training for medical imaging tasks.

Approach: We explore asynchronous Federated Learning (FL) using the FedBuff algorithm for classifying Optical Coherence Tomography (OCT) retina images. Unlike synchronous algorithms like FedAvg, which require all clients to participate simultaneously, FedBuff supports independent client updates. We compare its performance to both centralized models and FedAvg. Additionally, we develop a browser-based proof-of-concept system using modern web technologies to assess the feasibility and limitations of interactive, collaborative learning in real-world settings.

Results: FedBuff performs well in binary OCT classification tasks but shows reduced accuracy in more complex, multi-class scenarios. FedAvg achieves results comparable to centralized training, consistent with previous findings. While FedBuff underperforms compared to FedAvg and centralized models, it still delivers acceptable accuracy in less complex settings. The browser-based prototype demonstrates the potential for accessible, user-driven FL systems but also highlights technical limitations in current web standards, especially regarding local computation and communication efficiency.

Conclusions: Asynchronous FL via FedBuff offers a promising, privacy-preserving approach for medical image classification, particularly when synchronous participation is impractical. However, its scalability to complex classification tasks remains limited. Web-based implementations have the potential to broaden access to collaborative AI tools, but limitations of the current technologies need to be further investigated.

Keywords: Asynchronous Federated Learning, Medical Image Analysis, Decentralized Deep Learning, Optical Coherence Tomography (OCT), Browser-based Training, Interactive Machine Learning.

*Hasan Md Tusfiqur Alam, hasan.alam@dfki.de

1 Introduction

Traditional centralized machine learning (ML) approaches often rely on aggregating all training data in a single repository, which poses two critical issues in sensitive domains such as healthcare. First, domain experts, such as clinicians, radiologists, and other specialists, are rarely directly involved in model training, resulting in slow iteration and delayed integration of newly labeled data. Second, legal and ethical constraints often prohibit the sharing of patient data across institutional boundaries, thereby undermining both model performance and its broad applicability.

Interactive machine learning (IML), initially introduced by Fails and Olsen,[1] enables domain experts to iteratively refine model outputs through manual annotations and corrections by involving them in the training loop. For example, *Co-ML*[2] demonstrated how multiple users can collaboratively train image classifiers on tablet devices, primarily for educational dataset design. However, *Co-ML* does not address the privacy requirements inherent to clinical data.

Federated learning (FL) avoids data collection in a single repository by using decentralized model updates where each client trains locally and transmits only model parameters to a server for aggregation without sharing any of the training data.[3] This paradigm has shown strong promise in medical imaging applications, such as thyroid,[4] melanoma,[5] and brain tumor diagnosis.[6] Recent surveys have also shown its value for privacy-preserving collaborative learning.[7,8]

FL algorithms can be broadly categorized into synchronous and asynchronous aggregation strategies. In synchronous FL, all participating clients train simultaneously after receiving the latest global model, orchestrated by the central server.[3] The server then waits for all client updates before incorporating them into the global model, which can lead to delays caused by slow clients. In contrast, asynchronous FL allows clients to send updates independently and continu-

ously, improving efficiency in scenarios with heterogeneous availability.[9] Asynchronous FL can offer several benefits, including flexibility in participation for clients, as well as improved scalability in scenarios with a large number of collaborating devices. However, asynchronous training can also lead to the problem of stale updates, where outdated model states are incorporated into the global model.

Among asynchronous strategies, FL with Buffered Asynchronous Aggregation (FedBuff)[10] accumulates incoming updates in a buffer and applies them once a predefined threshold is reached. This buffering mechanism facilitates the integration of secure aggregation and differential privacy, making FedBuff particularly suitable for sensitive domains such as healthcare, while achieving comparable or superior performance to synchronous FL under both low and high privacy budgets.

In this work, we combine IML and asynchronous FL into a single framework for retinal disease diagnosis from optical coherence tomography (OCT) images. OCT is a widely used non-invasive imaging modality to capture high-resolution, detailed cross-sections of retinal tissue.[11] Prior studies have applied synchronous FL to OCT datasets using FedAvg with architectures such as VGG19 and ResNet18.[12–14] However, asynchronous FL has not yet been systematically evaluated on OCT data. We address this by applying FedBuff to the recently published OCTDL dataset,[15] which contains over 2,000 labeled images spanning seven retinal disease classes, including age-related macular degeneration (AMD), which is a particularly well-studied use case,[14] making it suitable for benchmarking asynchronous FL performance.

Despite growing adoption across diverse applications and significant algorithmic progress, FL still faces significant challenges in clinical environments due to compatibility with existing hospital systems, seamless integration, and end-user adoption.[8] To address these challenges, we additionally introduce a browser-based FL platform that requires no additional infrastructure and preserves

patient privacy by executing all training and inference on the client side. By using modern web technologies, such as WebAssembly,[16] WebGPU,[17] and ONNX Runtime, our system operates entirely within standard web browsers, simplifying deployment in existing hospital environments.

Prior web-based tools, e.g., Google's *Teachable Machine*,[18] demonstrate in-browser inference and labeling but lack federated protocols. Similarly, frameworks like *WebFed*[19] support only synchronous aggregation with minimal interactivity. In contrast, our open-source toolkit, InFL-UX,[20] integrates IML with asynchronous FL in a single front-end application, enabling domain experts to collaboratively label and train models without sharing sensitive data.

Overall, in this study, we focus on asynchronous model training applied to real-world medical data (i.e., OCT images) and evaluate diagnostic performance across multiple FL strategies. We highlight the challenges and limitations of previous works and present a proof-of-concept implementation based on modern web technologies. Our contributions are summarized as follows: (1) We evaluate the feasibility of asynchronous FL (FedBuff) for OCT image classification, representing the first application of FedBuff to this domain; (2) We compare the diagnostic accuracy of FedBuff against centralized training and synchronous FL (FedAvg) using state-of-the-art convolutional neural network (CNN) and transformer-based models; (3) We present a proof-of-concept web application for privacy-preserving and interactive FL, demonstrating the practical potential of browser-based model training in clinical workflows.

2 Related Works

In this section, we present the most relevant literature to our study. We start with publications on FL in medical imaging, then proceed with a few publications on browser-based deep learning and FL systems.

2.1 *Federated Learning in Medical Imaging*

The term *Federated Learning* was first introduced by McMahan et al.[3] It describes a decentralized learning paradigm that enables training a shared ML model with multiple participants (called *clients*), while the data remains on the participants' devices. In their work, the authors presented *FedAvg*, which serves as the foundational algorithm for FL. It combines local stochastic gradient descent on client devices with a server-side model averaging step, enabling efficient model training across decentralized and potentially non-independent and identically distributed (non-IID) data without sharing raw data.

In turn, this led to the adoption of FL across various medical use cases. These include classifying thyroid ultrasound[4] and melanoma images[5] for diagnosis, as well as identifying brain tumors in Magnetic Resonance Imaging (MRI) scans.[6] To gain an overview of this field of research, Guan et al.[8] provided a comprehensive survey on FL for medical images, suggesting that FL appears to be useful for a large variety of tasks. These include the classification and diagnosis of diseases from images obtained by different modalities, such as X-rays or MRI scans, as well as segmentation tasks for identifying the location of tumors. Similarly, Moshawrab et al.[7] conducted an extensive review on the usage of FL in disease prediction. Both studies also highlighted that FL in the medical domain faces several challenges in real-world applications, such as compatibility with existing hospital systems, integration challenges, and user adoption hurdles that need to be addressed.

Several works have also explored FL on OCT retina images for various use cases. Lo et al.[12] applied FL to referable diabetic retinopathy (RDR) classification. A larger-scale study was conducted by Ran et al.,[13] focusing on FL for glaucoma detection, which can lead to vision loss and blindness by damaging the optic nerve. Gholami et al.[14] tested FL on age-related macular degen-

eration (AMD), using three public OCT research datasets; AMD occurs when the macula, a part of the retina, is damaged, and causes vision loss, mostly among people aged 50 or older.

2.1.1 Asynchronous Federated Learning

Asynchronous FL, as the name suggests, does not require synchronous training among the clients. As the first algorithm of its kind, Xie et al.[9] proposed Asynchronous Federated Optimization (FedAsync). It functions in a fully asynchronous manner, meaning that each arriving model update from a client results in a new global model. That way, slow clients, referred to as *stragglers*, can no longer slow down the aggregation process, a common issue in synchronous FL. However, with this approach, client updates are potentially based on an older version of the global model, which is not a concern in the synchronous approach. Therefore, a *staleness function* needs to be used in order to assign a lower weight to *stale* updates when incorporating them into the global model. Then again, such fully asynchronous algorithms have issues with privacy, since they are not compatible with secure aggregation (SecAgg),[21] as individual updates are not hidden. Furthermore, there are increased communication costs due to the higher update frequency, as each update needs to be broadcast to all clients.

Nguyen et al.[10] introduced a semi-asynchronous algorithm called FedBuff, exploring the design space between synchronous and fully asynchronous FL. It aims to tackle the aforementioned challenges on privacy and scalability by not updating the global model with each incoming update. The server holds a buffer where client updates are stored. The global model is only updated once the buffer holds K client updates, where K is a tunable hyperparameter. This leads to less communication between the server and the clients, as the global model is updated less frequently in this approach. Privacy improvements are achieved in two ways: FedBuff can be extended with

differential privacy (DP), implemented on the server side, and it is compatible with SecAgg.[21] Numerous other works use FedBuff as the asynchronous FL for different use cases.[22–25]

Apart from the presented algorithms, there are other more recent approaches to asynchronous FL. For example, FedASMU[26] addresses common issues by implementing dynamic staleness-aware updates, where, in local training, clients periodically request the latest global model from the server to reduce staleness. However, the algorithm requires the server to trigger local training on the client devices. FedFix[27] is another example that uses a semi-asynchronous strategy similar to FedBuff.

The key factor in choosing FedBuff as the asynchronous FL for evaluation in this work is its ability to be extended with DP. In theory, local DP[28] can be applied to any FL algorithm, where clients apply the necessary operations to their update before sending it to the server. However, FedBuff applies DP on the server side, reducing the workload of clients. More importantly, the authors explicitly verify in their experiments that FedBuff performs well across different privacy budgets. For this reason, FedAsync and FedFix were not chosen, even though they satisfy the other requirements. FedASMU offers a new approach to address stale client updates, which is one of the biggest problems in asynchronous FL. Unfortunately, the training process in FedASMU is triggered by the central server, making it unusable for an interactive approach. Additionally, the implementation is more challenging due to the increased number of calculations and input parameters in the algorithm.

2.2 Browser-based Deep Learning & Federated Learning Systems

Deep learning in the web browser has already become feasible with the help of various JavaScript frameworks. Ma et al.[16] compared the performance of TensorFlow,[29] the state-of-the-art native ML

framework, against different JavaScript frameworks and found that for some tasks, the JavaScript frameworks can reach comparable performance.

Google proposed and implemented a no-code, browser-based ML application, called *Teachable Machine*,[18] which helps people with no technical ML knowledge train a model for their specific use case. The tool supports building image and sound classifiers using user-provided data that can be uploaded from the user's device through an intuitive interface. It utilizes TensorFlow's JavaScript implementation, i.e., TensorFlow.js,[30] for on-device inference and training in image classification using MobileNet.[31] However, Teachable Machine is dedicated to a single-user experience, without FL integration.

For browser-based synchronous FL, Lian et al.[19] implemented WebFed, a server-side synchronous FL framework that communicates with clients via WebSockets. The clients use local DP to add noise to their updates before sending them to the server. Experiments showed that their approach could reach an accuracy close to conventional FL, while choosing an appropriate privacy budget only has a subtle impact on the model's performance. However, due to the synchronous FL approach, a fixed number of clients need to participate in the training process without the possibility of joining or leaving the training at any time. Additionally, clients are required to remain online until all communication rounds are completed. The authors mentioned that a user interface is available for providing data, but no concrete details were provided on what such an interface could look like. Morell and Alba[32] presented an asynchronous algorithm for FL in this context and implemented it in their own platform. Their approach enables clients to join and leave the training process at any time, with experiments demonstrating that the platform can adapt effectively to changes in client availability. Their work, however, does not mention support for DP, and does not contain a user interface for providing the data.

3 Methodology

We evaluate models trained on centralized datasets against models obtained from FL simulations to compare their performance. As our study is model-agnostic and centers on the FL paradigm, this section describes the FL setups, while Section 4 reports the specific ML models and hyperparameter settings. The implementation details for the proof-of-concept application are separately described in Section 5.

3.1 Centralized Training Strategy

Centralized training serves as the baseline for comparison. Classification models are trained using labeled retinal OCT images on a unified dataset, simulating an ideal, fully aggregated data scenario. The objective is to identify optimal hyperparameters and baseline performance benchmarks before introducing FL constraints. A hyperparameter search is conducted using the Optuna framework[1], tuning batch size, learning rate, dropout rate, and augmentation settings. Each training session is executed with early stopping based on validation performance.

3.2 Federated Training Strategy

The FL setup simulates decentralized training across multiple clients with non-IID data partitions. In this work, we implement two widely recognized FL algorithms: FedAvg[3] and FedBuff[23] for OCT image classification. Both FedAvg and FedBuff algorithms are implemented to explore their performance under similar experimental constraints. Each client receives a local copy of the global model, trains it with local data for a defined number of epochs, and sends updates to the central server.

[1] https://optuna.org/ (Accessed October 15, 2025)

Algorithm 1 FedBuff Implementation

Input: server learning rate η_g, client learning rate η_l, client SGD steps Q, buffer size K, number of rounds R
Output: Trained model
$P \leftarrow \{\}$ ▷ map round number to parameters of that round
$CPV \leftarrow \{\ i = 0$ for each client $\}$ ▷ map client to its last used parameter version
$w_1 \leftarrow$ get weights from random client
for $r = 1 : R$ **do**
    $P[r] \leftarrow w_r$ ▷ save weights of current round
    $clients \leftarrow$ select K clients at random
    for each client c **do**
        $s_{\max} \leftarrow r - CPV[c] - 1$ ▷ calculate max staleness
        $s \leftarrow RandomStaleness(s_{\max})$ ▷ sample from half-normal distribution
        $paramVersion \leftarrow r - s$
        $CPV[paramVersion] \leftarrow r$ ▷ update last used parameter version
        $params \leftarrow P[paramVersion]$ ▷ get parameters based on staleness
        Run **FedBuff-client**$(params, \eta_l, Q)$ on c
    end for
    $U \leftarrow$ collect client updates
    $\overline{\Delta}^r \leftarrow \frac{\sum_{u \in U} \overline{\Delta}^u}{K}$
    $w_{r+1} \leftarrow w_r - \eta_g \overline{\Delta}^r$
end for

FedAvg operates under a synchronous protocol, where the server waits for all selected clients in each round to complete local updates before aggregating them. In contrast, FedBuff is employed as an asynchronous alternative; rather than waiting for a fixed group of clients, the server accumulates client updates into a buffer, and once the buffer reaches the predefined capacity, updates are averaged and applied to the global model. FedBuff introduces two key enhancements to support realistic simulation: learning rate normalization and staleness-aware update scaling. The former ensures training stability when batch sizes deviate from the global norm, and the latter reduces the impact of outdated updates from clients operating on stale model versions. Implementation details for FedBuff, with necessary adaptations for our use case, are detailed in Algorithm 1.

To ensure consistency with the centralized training setup, the same hyperparameters are similarly optimized. Additionally, a server-side learning rate is introduced in FedBuff. Evaluation of

the global model occurs after each aggregation, where the updated model is broadcast back to all clients for validation, and weighted averages of client-level metrics are computed. The same target metric used in centralized learning guides Optuna's pruning logic, allowing for the early rejection of underperforming configurations.

4 Experiments

This section describes the dataset and the experimental setup, including model architectures and hyperparameters, used to compare centralized training with federated learning simulations for OCT classification, as well as the results of these experiments.

4.1 Dataset

For the experiments conducted in this study, the *OCTDL*[15] (Optical Coherence Tomography Dataset for Image-Based Deep Learning Methods) is utilised. The dataset contains over 2000 OCT images, each labeled according to specific retinal diseases: Age-related Macular Degeneration (AMD), Diabetic Macular Edema (DME), Epiretinal Membrane (ERM), Retinal Artery Occlusion (RAO), Retinal Vein Occlusion (RVO), and Vitreomacular Interface Disease (VID). Scans from normal eyes (NO) that do not have one of the diseases are also included. Furthermore, each record contains a patient ID, so scans belonging to the same patient can be identified.

A summary of the dataset is provided in Table 1, which shows that the overall dataset is highly imbalanced. For our FL experiments, we split the dataset among different clients based on the patient ID, ensuring that all images from a single patient are assigned to only one client. This process realistically simulates a realistic scenario for a non-IID dataset, which is a common challenge in real-world medical applications. Figure 1 illustrates this partitioning for five clients; the left

Table 1 OCTDL dataset summary. The first two classes (AMD, NO) are used for the binary AMD diagnosis use case.

Disease	Label	Number of Scans	Number of Patients
Age-related Macular Degeneration	AMD	1231	421
Normal	NO	332	110
Diabetic Macular Edema	DME	147	107
Epiretinal Membrane	ERM	155	71
Retinal Artery Occlusion	RAO	22	11
Retinal Vein Occlusion	RVO	101	50
Vitreomacular Interface Disease	VID	76	51
Total		**2064**	**821**

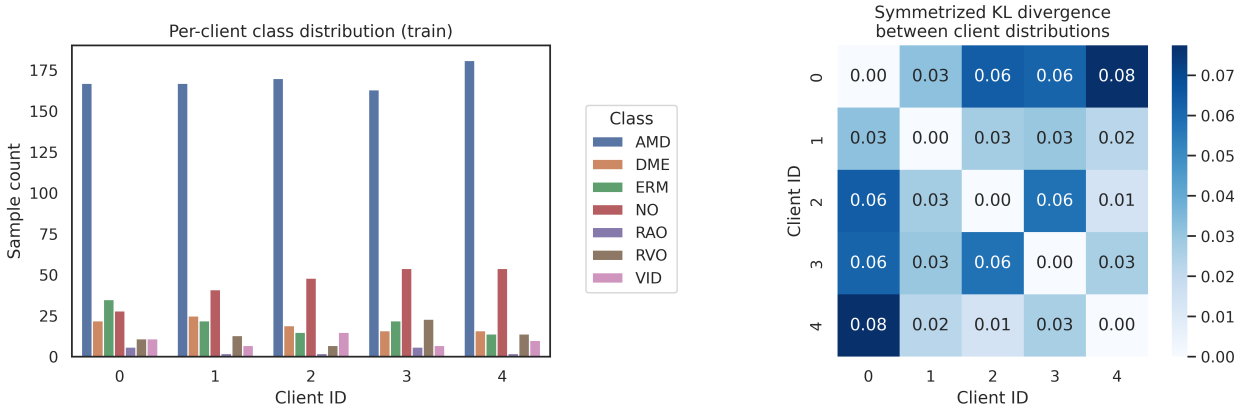

Fig 1 An illustration of the data distribution of the *OCTDL* dataset partitioned among five clients. The barplot of the left shows the per-client class distribution, highlighting the severe class imbalance within each client's local training data. In addition, the heatmap on the right shows the Kullback-Leibler (KL) divergence between the label distributions of every pair of clients.

plot shows a significant intra-client class imbalance, with the AMD class heavily outnumbering the other conditions in each local dataset; the right plot quantifies the inter-client heterogeneity by showing the Kullback-Leibler (KL) divergence between client label distributions. The non-zero divergence values confirm that each client holds a statistically distinct data distribution, validating the non-IID nature of our experimental setup.

4.2 Experimental Setup

We designed the experiments to evaluate two use cases: (1) AMD diagnosis, i.e., classifying an OCT scan as either AMD or NO, since Gholami et al.[14] also investigated this use case on previously published datasets; (2) A multi-class problem, including all classes of the dataset, following the

experiments conducted by Kulyabin et al.[15] on OCTDL.

In order to test the resulting models on unseen data and ensure a meaningful comparison, 15% of the dataset was set aside as test data for all experiments; the centralized experiments used an additional 15% for validation. Therefore, we ensured that each patient's data was exclusive to one of the subsets, in the same way the dataset's authors did. We also split the data at the patient level while distributing it among clients in the FL setting; this helped create realistic conditions for the experiments. Each client participating in the FL simulation reserved 20% of their data for validation. Due to the imbalanced nature of the dataset, a stratified split is employed here to ensure that each class is represented in both the training and validation sets for each client.

4.2.1 Machine Learning Models

We conducted the experiments on three CNN-based models, ResNet18, ResNet50,[33] and MobileNetV2[31] and on a Transformer based Vision model, ViT-B16.[34] We also reported the results for training the model from scratch without any weight initialisation and the transfer learning setting, where model weights were initialised using ImageNet[35] pretrained weights. We selected ResNet18 because it is a common baseline for OCT classification tasks, while ResNet50 is a deeper variant to explore the impact of model complexity; in addition, MobileNetV2 was chosen as a lightweight alternative to assess performance in resource-constrained environments, which is a key consideration for practical clinical deployment.

Additionally, we included a Vision Transformer (ViT) architecture in our comparison. Specifically, we used the *google/vit-base-patch16-224* model,[36,37] a state-of-the-art transformer available via the Hugging Face model repository[2]. This model was pre-trained on the large-scale ImageNet-

[2] https://huggingface.co/google/vit-base-patch16-224 (Accessed October 15, 2025)

21k dataset,[35,38] allowing us to evaluate the effectiveness of transfer learning from a transformer-based architecture for our classification tasks. The number of trainable parameters and approximate training time for these models are reported in Table 3

Table 2 Hyperparameters used in different learning strategies.

Hyperparameter	Used for	Values
Learning Rate	Centralized, FedAvg, FedBuff	[0.1, 0.0001]
Batch Size	Centralized, FedAvg, FedBuff	{8, 16, 32, 64, 128}
Apply Augmentation	Centralized, FedAvg, FedBuff	{True, False}
Dropout	Centralized, FedAvg, FedBuff	{0.0, 0.1, 0.2, 0.3, 0.4, 0.5}
Client Epochs	FedAvg, FedBuff	[1..10]
Server Learning Rate	FedBuff	[0.1, 0.0001]

4.2.2 Centralized Learning

In the centralized learning experiments, the models were trained for a maximum of 100 epochs. After each epoch, the model was evaluated on the validation set. Thereby, the metric by which we wanted to optimize the model was calculated based on the evaluation results. The implemented early stopping strategy started after epoch 20. Then, if the model's performance, i.e., the macro F1 score, did not improve for five epochs, the training stopped. During this process, the parameters of the best-performing model were continuously stored for testing purposes. The hyperparameter ranges used for optimisation are specified in Table 2.

4.2.3 Federated Learning Simulations

All FL simulations were run using 20 clients who participated in the training process. Our goal was to select a number that is large enough to allow experimentation with different numbers of clients per aggregation round, yet small enough so that each client received a sufficiently large local dataset. Each FL algorithm was tested with 10, 5, and 3 client updates per global model update. The simulations ran until the global model was updated 260 times, giving the models enough time

Table 3 Model parameters and average training time per training round in OCTDL dataset using a single GPU RTX 3080

Model	Params (M)	Approx. train time / round (sec)
ResNet18	11.7	20-30
ResNet50	25.6	25-35
MobileNetV2	3.4	10-15
ViT-B/16	86.0	40-50

Algorithm 2 Federated Learning Experiments

Run for each model configuration for both algorithms: FedAvg, FedBuff

Input: Number of clients $N = 20$, Number of global updates $U = 260$, Model configuration M, Classes for use case CL, FL algorithm F
$D \leftarrow$ Load dataset with classes CL
$D_{\text{train}}, D_{\text{test}} \leftarrow$ Randomly split dataset D into training set and test set (15%) on a patient's level
for each client C_i, $i \in \{1, 2, \ldots, N\}$ **do**
    Assign a subset D_i of D_{train} to client C_i
    Reserve 20% of D_i as local validation set for client C_i using stratified split
end for
for each update frequency $f \in \{10, 5, 3\}$ **do**
    / FedAvg: f := clients per round, FedBuff: f := buffer size */*
    $r \leftarrow \lfloor U/f \rfloor$
    Set number of rounds in F to r
    Run hyperparameter optimisation using F with $C_1 \ldots C_N$ on model M
end for
Test the final model on D_{test}

to converge. For the FL optimisation, the number of local epochs for each client was defined as an additional hyperparameter with a range from 1 to 10, following the experiments of Gholami et al.[14] The server learning rate for FedBuff was set to the same range as the client learning rate. These hyperparameters are also included in Table 2. An overview of the FL experiments is shown in Algorithm 2.

For FedAvg, we used the implementation provided by *Flower*[3]. For FedBuff, on the other hand, we used a custom implementation within the framework. *Flower* does not provide support for asynchronous FL algorithms. However, with a few modifications to FedAvg, we were able to simulate FedBuff's aggregation process, as only the model's performance is tested, rather than

[3] https://flower.ai/ (Accessed October 15, 2025)

wall-clock time. First, the simulation selects clients equal to the buffer size. Then, staleness is simulated by giving each client a random staleness value. In their work, Nguyen et al.[10] discovered, during their experiments, that client delays and staleness in FedBuff follow a half-normal distribution. The maximum staleness of a client is $s_{max} = R - P_r - 1$, where R is the current round and P_r is the last used parameter version. Then, each client receives parameters of the model version according to their staleness. Finally, the aggregation logic on the server side and the calculation of the update on the client side are implemented. Algorithm 1 shows our FedBuff implementation.

4.3 Results

In the following section, the results of the previously described experiments are presented. All metrics collected during the testing of the final models are presented in tabular form, with the most important values and rows highlighted to emphasise the key findings. Furthermore, several charts are used to showcase the performance difference between the centralized and FL approaches, as well as different model configurations. Finally, for the overall best-performing models, we show the learning curves for FedAvg and FedBuff. All the code and the best-performing hyperparameters for each experiment can be found in the GitHub repository https://github.com/tmaurer42/octdl-training.

To evaluate the performance of the trained models, we used metrics that account for class imbalance, i.e., Balanced accuracy (**Bal Acc**), and Macro-averaged F1 score (**F1 Macro**), which calculates F1 score for each class individually and reports their unweighted mean as the final score. For the binary setting, we also report the **F1 score**. For the purposes of evaluating the differences in performance among approaches and models, Cohen's h is used to measure the effect size,[39] which

is calculated as follows, given two probabilities p_1 and p_2:

$$\text{effectSize}(p_1, p_2) = |2 * \arcsin(\sqrt{p1}) - 2 * arcsin(\sqrt{p2})|$$

Per Cohen's rule of thumb, we consider effect sizes around $h = 0.20$ as small, $h = 0.5$ as medium, and $h = 0.8$ as large differences.

Table 4 Centralized performance metrics for both tasks. Best F1 macro scores per task are in bold.

Task	Model	Transfer Learning	Bal Acc	F1 Macro	F1 Score
Binary	**ResNet18**	x	0.953	**0.942**	0.972
	ResNet18	✓	0.963	0.938	0.969
	ResNet50	x	0.928	0.939	0.973
	ResNet50	✓	0.935	0.936	0.971
	MobileNetV2	x	0.915	0.905	0.952
	MobileNetV2	✓	0.930	0.919	0.961
	ViT-B/16	x	0.921	0.915	0.960
	ViT-B/16	✓	0.936	0.938	0.971
Multi-class	ResNet18	x	0.640	0.480	–
	ResNet18	✓	0.655	0.545	–
	ResNet50	x	0.663	0.509	–
	ResNet50	✓	0.672	0.520	–
	MobileNetV2	x	0.662	0.565	–
	MobileNetV2	✓	0.669	**0.600**	–
	ViT-B/16	x	0.660	0.530	–
	ViT-B/16	✓	0.668	0.585	–

4.3.1 Centralized Training

The centralized test results, both binary and multi-class settings, are displayed in Table 4. For Binary AMD disease classification, the best-performing model was ResNet18 without transfer learning, reaching an F1 macro score of 0.942. The other tested models achieved a similar performance, with a small difference ($h = 0.092$) between the best and worst performing setup. In multi-class settings, the MobileNetV2 model with transfer learning outperformed the ResNet50 model, achieving an F1 macro score of 0.600. The difference in performance among models is also small here but more noticeable ($h = 0.182$).

4.3.2 *Federated Learning Simulations*

Test metrics for binary AMD diagnosis using FedAvg are displayed in Table 5. In all three client settings, the best-performing models achieved performance comparable to that of the centralized model. The best configuration with an F1 macro score of 0.958 was the ResNet18 model without transfer learning and three clients per round. The difference between the best and worst configurations is moderate to small ($h = 0.332$). FedAvg outperforms the centralized models in terms of F1 macro score by a small amount ($h = 0.074$).

Table 5 Performance metrics for AMD diagnosis with FedAvg. For each number of clients per round, the best F1 macro score is displayed in bold text. The overall best-performing model is also marked in bold.

Model	Transfer Learning	Bal Acc	F1 Macro	F1 Score	Clients per Round
ResNet18	x	0.923	0.928	0.967	10
ResNet18	✓	0.947	**0.941**	0.972	10
ResNet50	x	0.895	0.876	0.938	10
ResNet50	✓	0.932	0.934	0.970	10
MobileNetV2	x	0.907	0.893	0.951	10
MobileNetV2	✓	0.918	0.907	0.955	10
ViT-B/16	x	0.931	0.934	0.969	10
ViT-B/16	✓	0.942	0.940	0.971	10
ResNet18	x	0.937	**0.946**	0.975	5
ResNet18	✓	0.951	0.937	0.969	5
ResNet50	x	0.863	0.868	0.939	5
ResNet50	✓	0.908	0.905	0.966	5
MobileNetV2	x	0.920	0.912	0.959	5
MobileNetV2	✓	0.939	0.925	0.963	5
ViT-B/16	x	0.941	0.939	0.973	5
ViT-B/16	✓	0.946	0.943	0.974	5
ResNet18	x	0.955	**0.958**	0.981	3
ResNet18	✓	0.947	0.941	0.972	3
ResNet50	x	0.913	0.932	0.970	3
ResNet50	✓	0.935	0.938	0.974	3
MobileNetV2	x	0.922	0.913	0.958	3
MobileNetV2	✓	0.930	0.919	0.961	3
ViT-B/16	x	0.944	0.944	0.976	3
ViT-B/16	✓	0.949	0.952	0.978	3

For FedBuff, Table 6 shows the metrics for the test set. MobileNetV2 with transfer learning achieved the best performance with an F1 macro score of 0.792 and a buffer size of three. The difference in performance between the best and worst configurations is moderate to high ($h = 0.646$). FedBuff performed worse compared to the centralized training, showing a medium difference in

performance ($h = 0.456$).

A comparison between the best performing model architectures for the binary classification use case is presented in Figure 2 (a). For each model, the F1 macro score of the centralized learning, FedAvg, and FedBuff is displayed. ResNet18 and MobileNetV2 with transfer learning perform similarly well on all three approaches. The ResNet18 and ResNet50 models trained from scratch performed well in the centralized and FedAvg scenario, but worse using FedBuff.

For the multi-class use case, all results for FedAvg and FedBuff are presented in Table 7 and Table 8, respectively. Here, FedAvg performed better than the centralized model by a small amount ($h = 0.164$) with an F1 macro score of 0.678. The two used models differ in performance by a medium amount ($h = 0.517$), with the ResNet50 model yielding the highest score. For FedBuff, the FL approach yielded worse results, with the MobileNetV2 model achieving an F1 macro score of 0.295, representing a medium to large difference ($h = 0.623$) compared to the best centralized model. A comparison between the best performing model architectures is presented in Figure 2 (b).

Figure 3 compares the learning curves of ResNet18 with transfer learning using FedAvg and FedBuff for AMD diagnosis. In ResNet18 with FedAvg, the F1 score increases rapidly across all client counts, with more clients per global update resulting in fewer updates required to reach the maximum. ResNet18 with FedBuff shows slower, more gradual learning, with fewer clients per update (i.e., lower buffer size), and requires fewer global updates to reach the maximum. The training process appears to be more stable with a lower buffer size.

To better understand the classification performance, Figure 4 and Figure 5 present the normalized confusion matrices for the best-performing models from each training approach. In the binary task (i.e., Figure 4), the centralized and FedAvg methods achieved strong, comparable per-

Table 6 Performance metrics for AMD diagnosis with FedBuff. For each number of clients per round, the best F1 macro score is displayed in bold text. The overall best-performing model is also marked in bold.

Model	Transfer Learning	Bal Acc	F1 Macro	F1 Score	Clients per Round
ResNet18	x	0.755	0.651	0.737	10
ResNet18	✓	0.618	0.603	0.784	10
ResNet50	x	0.517	0.489	0.856	10
ResNet50	✓	0.642	0.621	0.828	10
MobileNetV2	x	0.768	0.725	0.846	10
MobileNetV2	✓	0.811	**0.773**	0.873	10
ViT-B/16	x	0.702	0.695	0.834	10
ViT-B/16	✓	0.784	0.762	0.862	10
ResNet18	x	0.697	0.662	0.801	5
ResNet18	✓	0.806	**0.787**	0.891	5
ResNet50	x	0.561	0.565	0.820	5
ResNet50	✓	0.612	0.601	0.826	5
MobileNetV2	x	0.648	0.629	0.812	5
MobileNetV2	✓	0.686	0.663	0.815	5
ViT-B/16	x	0.723	0.705	0.845	5
ViT-B/16	✓	0.767	0.758	0.868	5
ResNet18	x	0.563	0.567	0.838	3
ResNet18	✓	0.685	0.668	0.823	3
ResNet50	x	0.564	0.568	0.824	3
ResNet50	✓	0.622	0.611	0.835	3
MobileNetV2	x	0.774	0.759	0.864	3
MobileNetV2	✓	0.819	**0.792**	0.890	3
ViT-B/16	x	0.733	0.721	0.852	3
ViT-B/16	✓	0.801	0.784	0.872	3

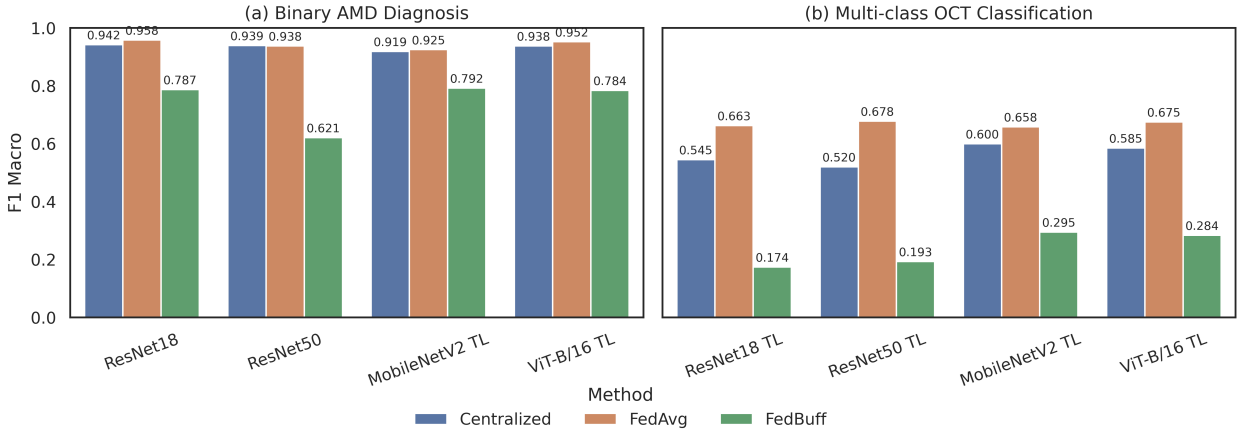

Fig 2 Comparison of macro-averaged F1 scores across Centralized, FedAvg, and FedBuff training methods. Figure a (left) shows performance on the binary AMD diagnosis task using four model variants. Figure b (right) shows results for the multi-class classification task. 'TL' in the suffix of the model name denotes transfer learning.

Table 7 Performance metrics for the multi-class use case with FedAvg. For each number of clients per round, the best F1 macro score is displayed in bold text. The overall best-performing model is also marked in bold.

Model	Transfer Learning	Bal Acc	F1 Macro	Clients per Round
ResNet18	x	0.402	0.418	10
ResNet18	✓	0.523	0.546	10
ResNet50	x	0.394	0.424	10
ResNet50	✓	0.576	0.559	10
MobileNetV2	x	0.588	0.573	10
MobileNetV2	✓	0.613	**0.601**	10
ViT-B/16	x	0.607	0.590	10
ViT-B/16	✓	0.628	0.596	10
ResNet18	x	0.512	0.535	5
ResNet18	✓	0.575	0.569	5
ResNet50	x	0.527	0.546	5
ResNet50	✓	0.590	0.584	5
MobileNetV2	x	0.602	0.597	5
MobileNetV2	✓	0.618	**0.613**	5
ViT-B/16	x	0.621	0.606	5
ViT-B/16	✓	0.633	0.611	5
ResNet18	x	0.603	0.649	3
ResNet18	✓	0.645	0.663	3
ResNet50	x	0.617	**0.678**	3
ResNet50	✓	0.655	0.671	3
MobileNetV2	x	0.661	0.657	3
MobileNetV2	✓	0.670	0.658	3
ViT-B/16	x	0.668	0.665	3
ViT-B/16	✓	0.682	0.675	3

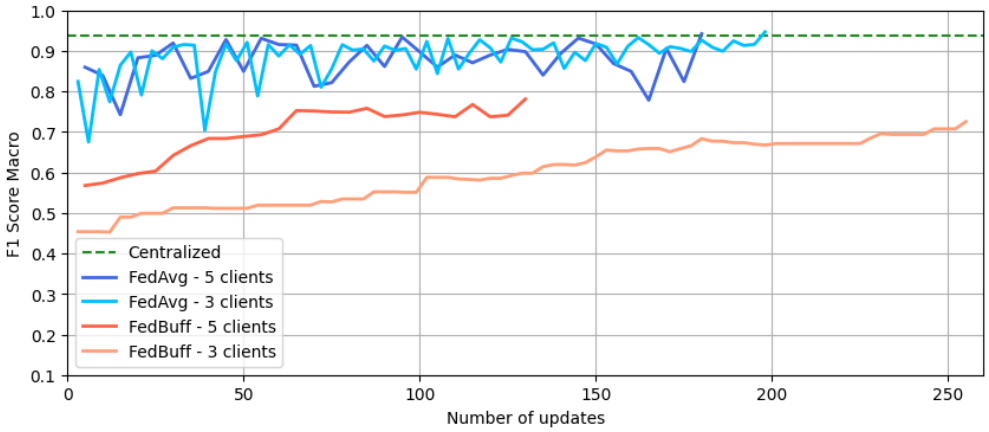

Fig 3 Validation learning curves of best models in AMD diagnosis, FedAvg / FedBuff with different no. of clients. The dotted horizontal line represents the centralized model's F1 macro score.

formance; their confusion matrices show high values on the main diagonal, indicating accurate predictions for both AMD and NO classes; although the FedBuff model exhibited a noticeable drop in performance, with a higher rate of misclassifying NO cases as AMD, its overall results remained competitive in this binary setting. However, this performance difference was amplified

Table 8 Performance metrics for the multi-class use case with FedBuff. For each number of clients per round, the best F1 macro score is displayed in bold text. The overall best-performing model is also marked in bold.

Model	Transfer Learning	Bal Acc	F1 Macro	Buffer Size
ResNet18	x	0.115	0.102	10
ResNet18	✓	0.158	0.137	10
ResNet50	x	0.126	0.107	10
ResNet50	✓	0.171	0.155	10
MobileNetV2	x	0.236	0.224	10
MobileNetV2	✓	0.277	**0.259**	10
ViT-B/16	x	0.252	0.238	10
ViT-B/16	✓	0.268	0.246	10
ResNet18	x	0.162	0.145	5
ResNet18	✓	0.188	0.174	5
ResNet50	x	0.185	0.158	5
ResNet50	✓	0.212	0.193	5
MobileNetV2	x	0.243	0.262	5
MobileNetV2	✓	0.267	**0.295**	5
ViT-B/16	x	0.251	0.273	5
ViT-B/16	✓	0.262	0.284	5
ResNet18	x	0.128	0.113	3
ResNet18	✓	0.154	0.139	3
ResNet50	x	0.142	0.114	3
ResNet50	✓	0.177	0.152	3
MobileNetV2	x	0.193	0.181	3
MobileNetV2	✓	0.204	0.194	3
ViT-B/16	x	0.211	0.187	3
ViT-B/16	✓	0.226	**0.198**	3

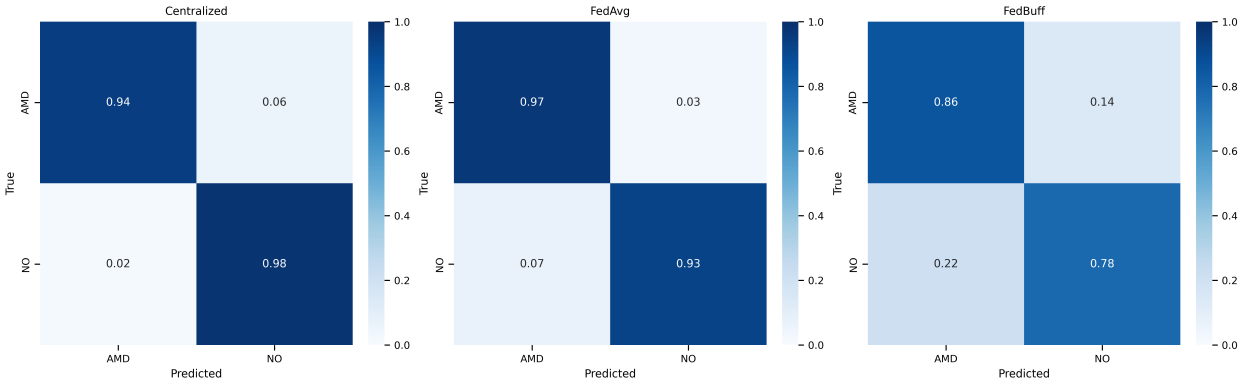

Fig 4 Normalized confusion matrices for the binary AMD classification task. The matrices compare the performance of the best-performing model configurations for the Centralized, FedAvg, and FedBuff training approaches.

in the multi-class setting (i.e., Figure 5); the centralized and FedAvg models achieved reasonable separation between classes; while the FedBuff model struggled significantly, its confusion matrix reveals widespread misclassifications and a strong tendency to incorrectly predict the majority class

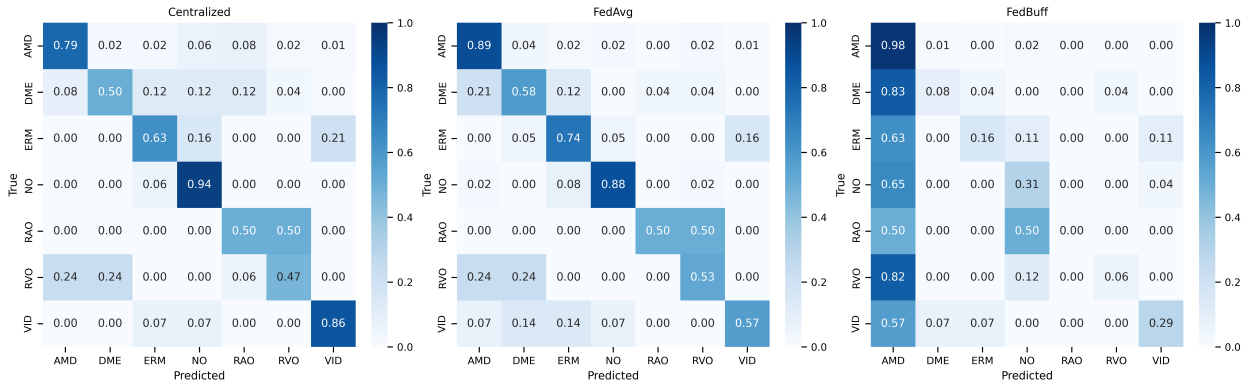

Fig 5 Normalized confusion matrices for the multi-class classification task. The matrices compare the performance of the best-performing model configurations for the Centralized, FedAvg, and FedBuff training approaches across all seven classes.

(AMD) for several other conditions, highlighting its difficulty handling the severe class imbalance within the FL environment.

5 Proof-of-Concept Application

The proof-of-concept application demonstrates how modern web technologies can be combined to enable collaborative IML. This section introduces these technologies and describes the application design in terms of architecture, user interface (UI), and workflow. Additionally, we present relevant implementation details.

5.1 Application Design

The application architecture in Figure 6 consists of two main services: (1) the **FL Server**, which handles update aggregation and stores the global model; (2) the **FL Web Server**, which serves the web-based user interface and necessary files to the client's edge device. A workflow of the web application from the client side is shown in Figure 7. Once users upload images to the client interface, the application performs label inference using the current global model. Users can accept, modify, or reject these suggestions before proceeding. After validation, users initiate local training on their

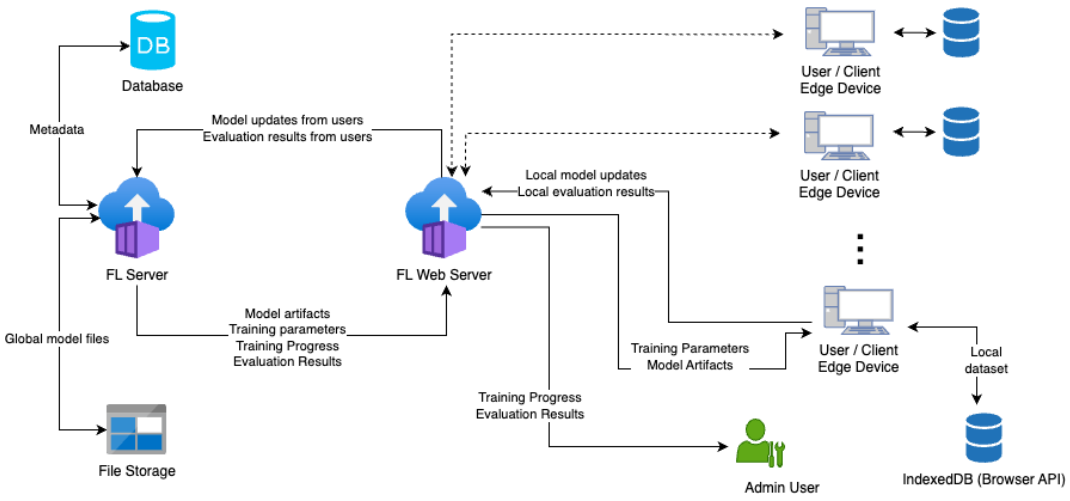

Fig 6 Proof-of-Concept Application: Architecture Overview

labeled images. Upon completion, the model update (gradient or weight delta) is automatically sent back to the FL server. This update is then considered for aggregation depending on the FL strategy being employed. While the proof-of-concept application focuses on image classification, other computer vision tasks, such as object detection or image segmentation, can be implemented through the abstract classes provided by the implementation.

5.2 Technologies

The application is designed to be lightweight, browser-compatible, and modular enough to support future extension to other imaging tasks such as detection or segmentation. We used Open Neural Network Exchange *(ONNX) Runtime*[4], which is a cross-platform machine learning library that enables fast on-device inference and training; most importantly, the library is available for web browsers. It utilizes two modern browser APIs to ensure an efficient machine learning workflow: *WebAssembly*[40] and *WebGPU*.[17] WebAssembly is a low-level, assembly-like language for

[4] https://onnxruntime.ai/ (Accessed October 15, 2025)

the browser, enabling web applications to execute code with near-native performance. It can serve as a compilation target for many popular programming languages. With WebGPU, browsers can use the system's GPU for high-performance computations. The implementation as a web application makes it easy to integrate with existing systems, since a web browser is present on nearly all computer systems. One big advantage of ONNX Runtime over other browser-based ML libraries, such as TensorFlow.js,[30] is the compatibility with the ONNX format. Since models from the most popular ML frameworks can be exported to ONNX, the application can easily work with existing models. We use *IndexedDB*,[41] which allows the browser to retain user data in cache memory, such as images, labels, and training history across sessions, ensuring persistence even if the browser is closed.

The backend of the system is implemented using *Flask*,[42] a Python-based web framework that serves the application and mediates communication between the web interface and the FL server. The frontend is built with modular *Web Components*, facilitating reuse and consistent rendering across different browsers. The system supports model updates in ONNX format, allowing flexibility in swapping in different architectures exported from PyTorch or TensorFlow.

The workflow is designed to be intuitive and minimally technical, allowing clinicians or field workers to participate in training without requiring deep ML expertise. This design makes the system suitable for deployment in low-resource or privacy-sensitive environments, such as small clinics or cross-institutional research collaborations.

6 Discussion

This section discusses and interprets the experimental results presented in Section 4.3 and evaluates the challenges and limitations encountered during the development of the proof-of-concept

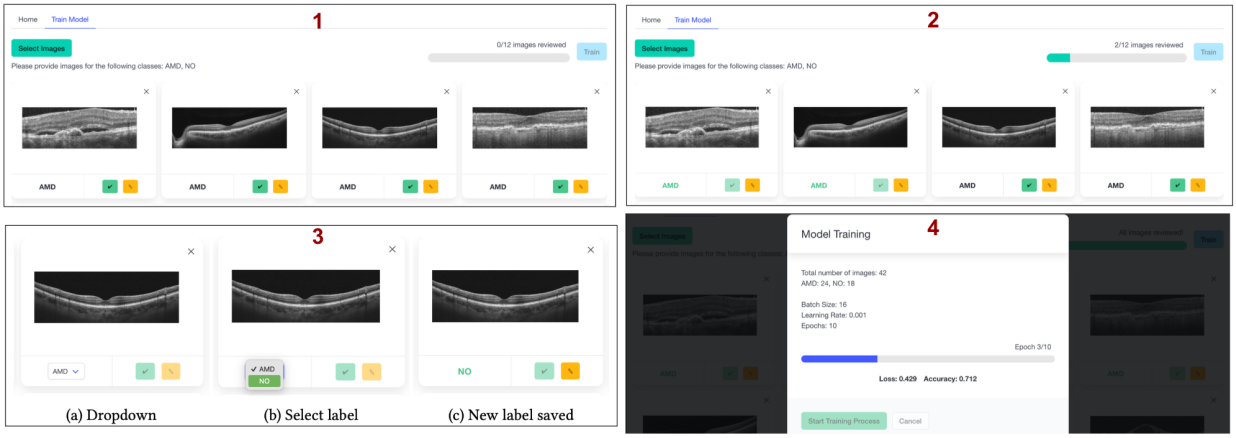

Fig 7 Proof-of-Concept Application: User Interface for OCT Image Review and Model Training. (1–2) Users upload and review OCT images for AMD diagnosis. (3) Labels can be changed using a dropdown menu. (4) Once all images are reviewed, model training begins with progress shown in real time.

application. The aim is to provide thorough answers to the research questions defined in Section 1.

RQ1: How viable is an asynchronous FL approach to optimize ML-based medical diagnosis using OCT retina images?

Our experimental results for asynchronous FL using the FedBuff algorithm are shown in Table 6 for the binary AMD classification task. In this setting, FedBuff achieved the highest F1 macro score of 0.792, using MobileNetV2 with transfer learning and a buffer size (i.e., clients per round) of three. A close second was the pretrained ResNet18 model with five clients per round.

For the multiclass classification task, as shown in Table 8, FedBuff's performance dropped significantly, reaching a maximum F1 macro score of only 0.295. This was again achieved with MobileNetV2 and transfer learning, using five clients per round. A key reason for this drop is the class imbalance in the dataset. As Table 1 illustrates, the class distribution ranges from 1,231 samples for AMD to only 22 samples for RAO. Partitioning the dataset at the patient level among 20 clients further exacerbates this imbalance, with some clients receiving no samples of underrepresented classes, such as RAO.

Table 9 Qualitative comparison of centralized, synchronous, and asynchronous FL strategies, highlighting trade-offs between coordination, latency tolerance, diagnostic accuracy, and communication overhead on OCT data.

Method	Model Coordination	Latency Tolerance	Accuracy	Comm. Overhead
Centralized	High (Single Server, full data)	Low	**Upper bound**	Low
FedAvg	Moderate (Synchronous rounds)	Moderate	**Near-centralized**	Moderate
FedBuff	**Low (Asynchronous buffer updates)**	**High**	Moderate drop	**Low (per client)**

Based on these findings, asynchronous FL with FedBuff appears to be a viable and promising approach for binary OCT diagnosis tasks. However, its effectiveness diminishes for imbalanced multiclass problems. A comparative summary of the three learning strategies is provided in Table 9. Centralized training achieves the highest accuracy as it has access to the full dataset, but requires all data to be collected centrally, which limits scalability and privacy. FedAvg offers a strong trade-off between accuracy and privacy but requires synchronous coordination across clients. In contrast, FedBuff trades a moderate drop in accuracy due to staleness and delayed updates for greater latency tolerance and lower per-client communication overhead, which is particularly important in web-based and heterogeneous clinical deployments. Additional techniques—such as advanced sampling, class rebalancing, or synthetic augmentation—may be necessary to improve performance in such cases. Overall, asynchronous FL is a valid strategy for privacy-preserving medical imaging, warranting further investigation and refinement.

RQ2: How does asynchronous FL perform in terms of diagnostic accuracy for OCT retina images compared to established centralized and synchronous FL methods?

For binary AMD classification, centralized training methods yielded consistently strong results across all architectures. As shown in Figure 2(a), the centralized model reached an F1 macro score of 0.942, and almost all models achieved similar performance with the FedAvg method, while FedBuff scored lower across configurations, with the best score of 0.792 for MobileNetV2 with transfer learning. This moderate drop may be attributed to staleness in asynchronous updates,

Table 10 Effect sizes between centralized and FL model performance for AMD diagnosis. TL indicates that the model uses transfer learning.

Model	FedAvg	FedBuff
ResNet18	0.074 (small)	0.756 (large)
ResNet18 TL	0.014 (small)	0.456 (medium)
ResNet50	0.030 (small)	0.936 (large)
MobileNet TL	0.023 (small)	0.461 (medium)

increased complexity in hyperparameter tuning due to the server learning rate, and less optimized transfer learning configurations.

For the multiclass task, centralized models performed poorly in our setup, with the best reaching an F1 macro score of 0.600, as shown in Table 4. This underperformance is likely linked to the patient-level data split, which may have inadvertently led to unbalanced or unrepresentative training and test sets. FedAvg slightly outperformed centralized models in this case, with a score of 0.678, while FedBuff again underperformed with a score of 0.295, as shown in Figure 2(b).

To verify our assumptions, we trained a ResNet50 model under ideal conditions, ensuring equal class distribution across all data splits. This resulted in a balanced accuracy of 0.820 and an F1 macro score of 0.810, which is comparable to those reported by the dataset authors, validating our hypothesis that data partitioning is a key performance bottleneck. Across both tasks, ResNet18 and MobileNetV2 emerged as the best-performing architectures. While models trained from scratch sometimes performed better in centralized settings, they underperformed in FedBuff, underscoring the utility of transfer learning in resource-constrained or asynchronous environments.

In terms of reproducibility, our results align with those of Gholami et al.,[14] which found that ResNet18 models using FedAvg performed comparably to centralized ones across multiple datasets. Table 10 quantifies this alignment using Cohen's h, confirming method reproducibility. However, we did not fully reproduce the results of Kulyabin et al.,[15] who reported an F1 score of 0.866 for ResNet50 trained on the full dataset. Our experiments under the same model architecture

yielded an F1 macro score of 0.600, which is likely due to the different data splits and class distributions. However, on the binary subset, we achieved a comparable F1 score of 0.939, confirming partial reproducibility.

Validation learning curves (Figure 3) further show that ResNet18 offers more stable training, while MobileNetV2 achieves faster convergence. Both models are lightweight, with sizes of 13 MB for MobileNetV2 and 45 MB for ResNet18, and have relatively few trainable parameters when using transfer learning. This makes them well-suited for browser-based FL, where computational resources are limited.

Transfer learning appears more robust in the asynchronous FedBuff setting because pretrained feature extractors provide a stable and semantically meaningful initialization, reducing sensitivity to stale or inconsistent client updates. Prior work has shown that heterogeneous client data induces client drift and unstable optimization dynamics in federated settings.[43–45] Asynchronous aggregation further amplifies these effects.[46] Pretrained backbones require smaller and more coherent gradient updates, which helps mitigate drift and stabilizes convergence under asynchronous conditions. This behavior is consistent with findings from FedBN and related studies addressing feature-level priors in non-IID federated learning.[47]

In summary, FedBuff underperforms relative to FedAvg and centralized learning but shows potential, especially for binary classification tasks and future large-scale deployments. The trade-offs between performance and flexibility justify further exploration of asynchronous FL in this domain.

RQ3: What are the challenges and limitations of implementing FL in a browser-based environment, and how can they be addressed?

Implementing FL in a browser-based environment presents several technical and usability challenges. First and foremost is the choice of algorithm. To accommodate asynchronous client participation, the FL algorithm must support independent client–server communication. FedBuff satisfies this requirement while offering acceptable performance in binary OCT classification, making it a suitable candidate for proof-of-concept deployment.

Model size and training efficiency on low-resource clients are also critical considerations. Our selected models, i.e., ResNet18 and MobileNetV2, performed well in FL experiments and are computationally lightweight, which makes them practical for browser execution using transfer learning. Integration with hospital systems and user adoption, as highlighted by Guan et al.,[8] are common barriers in real-world deployments. A browser-based solution helps address this by eliminating the need for software installation and ensuring cross-platform compatibility. To support adoption, the proof-of-concept application provides an intuitive, guided user interface for data upload, training, and monitoring.

On the technical side, training must be implemented efficiently using modern browser-compatible ML frameworks. Our application utilizes ONNX Runtime, a framework-agnostic ML framework that supports model execution in the browser. Unlike TensorFlow.js,[30] ONNX Runtime supports the WebGPU API, offering potential acceleration through the GPU. However, ONNX Runtime Web is still in active development and has several limitations. Currently, only a limited set of loss functions is implemented, and the optimizer's learning rate cannot be dynamically adjusted. Additionally, weighted loss functions, used to address class imbalance in our experiments, are not yet

supported in browser-based training. Furthermore, although ONNX Runtime supports inference on WebGPU, training is still limited to the WebAssembly backend. Another constraint is browser storage. Most modern browsers cap local storage (e.g., IndexedDB) based on available disk space, which can limit the size of locally stored datasets. Finally, WebGPU itself is not yet universally supported and may be restricted in high-security environments, as it is currently only available in developer versions of some browsers.

In conclusion, while the proof-of-concept demonstrates the technical feasibility of browser-based asynchronous FL, current limitations in tooling, storage, and hardware access must be addressed. As browser technologies continue to evolve, particularly with full WebGPU support and enhanced ONNX capabilities, such applications will become more viable for deployment in medical environments.

7 Conclusion and Future Direction

In this study, we explored asynchronous FL in comparison to synchronous FL and centralized training, demonstrating its feasibility through a browser-based proof-of-concept application built with ONNX Runtime and modern web technologies. Applied to the OCTDL dataset, our experiments showed that while FedAvg performed on par with centralized learning, the asynchronous FedBuff algorithm lagged behind, especially in multi-class classification. Lightweight models, such as MobileNetV2 and pre-trained ResNet18, proved more effective under asynchronous conditions.

Despite technical constraints in current web environments, for example, limited storage, fixed learning rates, and restricted WebGPU availability, our proof-of-concept application showed that client-side training and FL aggregation are already achievable in the browser. Looking ahead, several promising directions remain:

- **Optimize FedBuff performance** through systematic tuning of server learning rates, client counts, and transfer learning layer configurations.
- **Apply FedBuff to broader tasks** such as object detection or larger OCT datasets, and evaluate its robustness across different class splits.
- **Explore alternative asynchronous FL algorithms** and compare their effectiveness in medical imaging scenarios.
- **Integrate active learning** techniques[48,49] to prioritise informative samples, reducing labelling effort and improving training efficiency.
- **Enhance the application** with support for segmentation tasks, explainability tools, such as Grad-CAM,[50] and interpretable training pipelines.[51,52]
- **Enable multi-task training** with support for multiple datasets per client to reflect realistic deployment settings.

These directions offer pathways toward scalable, privacy-preserving, and user-centric FL systems deployable directly in web environments.

Acknowledgments

This work was funded, in part, by the German Federal Ministry of Education and Research (BMBF) under grant number 01IW23002 (No-IDLE) and grant number 01IW24006 (NoIDLEChatGPT), by the Lower Saxony Ministry of Science and Culture (MWK) in the zukunft.niedersachsen program, and by the Endowed Chair of Applied AI at the University of Oldenburg. OpenAI ChatGPT was used for grammatical correction and language cleanup in the manuscript writing process.

Reproducibility and Conflict of Interest

The authors declare no conflicts of interest. All code and experimental results supporting this work are publicly available at: https://github.com/tmaurer42/octdl-training

Author Biographies

Hasan Md Tusfiqur Alam completed his M.Sc. in Computer Science at Saarland University in 2023. He is currently pursuing a Ph.D. and working as a researcher in the Interactive Machine Learning Department at the German Research Center for Artificial Intelligence (DFKI). His research focuses on explainable AI, multimodal learning, and federated learning, with applications in medical imaging and intelligent user interfaces.

Tim Maurer received the B.Sc. degree in Computer Science from Saarland University, Germany, in 2024. His undergraduate research centered on interactive and privacy-preserving machine learning for medical imaging, with a focus on optical coherence tomography. He is currently a Software Engineer at ProOmea GmbH, Germany.

Abdulrahman M. Selim received the B.Sc. degree in Systems and Biomedical Engineering from Cairo University, Egypt, in 2019, and the M.Sc. degree in Computer Science from Saarland University, Germany, in 2022. He is currently a researcher in the Interactive Machine Learning Department at the German Research Center for Artificial Intelligence (DFKI). His research lies at the intersection of applied machine learning and human–computer interaction, with a focus on inferring human behavior from multimodal interactions.

Dr. Michael Barz received the M.Sc. degree in Computer Science from Saarland University, Germany, in 2015, and the Ph.D. degree from the University of Oldenburg, Germany, in 2025. He is a senior researcher and department manager in the Interactive Machine Learning Department

at the German Research Center for Artificial Intelligence (DFKI). His research interests include intelligent user interfaces, multimodal interaction—with an emphasis on eye tracking—and human factors in AI-based systems.

Matthias Eiletz holds a Master's degree in Computer Science. He previously conducted research on distributed systems at the August-Wilhelm-Scheer Institute in Saarbrücken, focusing on scalable and resilient software architectures. He is currently Managing Director of ProOmea GmbH, where he applies technology-driven approaches to digital solutions in the real estate sector.

Prof. Dr. Daniel Sonntag leads the Interactive Machine Learning Department at the German Research Centre for Artificial Intelligence (DFKI) and holds the Professorship of Applied Artificial Intelligence at the University of Oldenburg. His research spans intelligent user interfaces, natural language processing, information retrieval, dialogue systems, and explainable machine learning, with applications in healthcare and Industry 4.0. He has published over 200 peer-reviewed papers, received the 2011 German High-Tech Champion Award, and leads multiple national and European research projects.

References

1 J. A. Fails and D. R. Olsen, "Interactive machine learning," in *Proceedings of the 8th International Conference on Intelligent User Interfaces*, *IUI '03*, 39–45, Association for Computing Machinery, (New York, NY, USA) (2003).

2 T. Tseng, J. King Chen, M. Abdelrahman, *et al.*, "Collaborative machine learning model building with families using co-ml," in *Proceedings of the 22nd Annual ACM Interaction Design and Children Conference*, *IDC '23*, 40–51, Association for Computing Machinery, (New York, NY, USA) (2023).

3 B. McMahan, E. Moore, D. Ramage, *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, Eds., *Proceedings of Machine Learning Research* **54**, 1273–1282, PMLR (2017).

4 H. Lee, Y. J. Chai, H. Joo, *et al.*, "Federated learning for thyroid ultrasound image analysis to protect personal information: Validation study in a real health care environment," *JMIR Med Inform* **9**, e25869 (2021).

5 S. Haggenmüller, M. Schmitt, E. Krieghoff-Henning, *et al.*, "Federated Learning for Decentralized Artificial Intelligence in Melanoma Diagnostics," *JAMA Dermatology* **160**, 303–311 (2024).

6 M. Islam, M. T. Reza, M. Kaosar, *et al.*, "Effectiveness of federated learning and cnn ensemble architectures for identifying brain tumors using mri images," *Neural Processing Letters* **55** (2022).

7 M. Moshawrab, M. Adda, A. Bouzouane, *et al.*, "Reviewing federated machine learning and its use in diseases prediction," *Sensors* **23**(4) (2023).

8 H. Guan, P.-T. Yap, A. Bozoki, *et al.*, "Federated learning for medical image analysis: A survey," *Pattern Recognition* **151**, 110424 (2024).

9 C. Xie, S. Koyejo, and I. Gupta, "Asynchronous Federated Optimization," *arXiv e-prints* (2019).

10 J. Nguyen, K. Malik, H. Zhan, *et al.*, "Federated learning with buffered asynchronous aggregation," in *Proceedings of The 25th International Conference on Artificial Intelligence*

and Statistics, G. Camps-Valls, F. J. R. Ruiz, and I. Valera, Eds., *Proceedings of Machine Learning Research* **151**, 3581–3607, PMLR (2022).

11 D. Huang, E. A. Swanson, C. P. Lin, *et al.*, "Optical coherence tomography," *Science* **254**(5035), 1178–1181 (1991).

12 J. Lo, T. T. Yu, D. Ma, *et al.*, "Federated learning for microvasculature segmentation and diabetic retinopathy classification of oct data," *Ophthalmology Science* **1**(4), 100069 (2021).

13 A. R. Ran, X. Wang, P. P. Chan, *et al.*, "Developing a privacy-preserving deep learning model for glaucoma detection: a multicentre study with federated learning," *British Journal of Ophthalmology* (2023).

14 S. Gholami, J. Lim, T. Leng, *et al.*, "Federated learning for diagnosis of age-related macular degeneration," *Frontiers in Medicine* **10** (2023).

15 M. Kulyabin, A. Zhdanov, A. Nikiforova, *et al.*, "OCTDL: Optical Coherence Tomography Dataset for Image-Based Deep Learning Methods," *Scientific Data* **11**, 365 (2024).

16 Y. Ma, D. Xiang, S. Zheng, *et al.*, "Moving deep learning into web browser: How far can we go?," in *The World Wide Web Conference*, *WWW '19*, 1234–1244, Association for Computing Machinery, (New York, NY, USA) (2019).

17 K. Ninomiya, J. Blandy, B. Jones, *et al.*, "WebGPU," W3C Working Draft, W3C (2024).

18 M. Carney, B. Webster, I. Alvarado, *et al.*, "Teachable machine: Approachable web-based tool for exploring machine learning classification," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, *CHI EA '20*, 1–8, Association for Computing Machinery, (New York, NY, USA) (2020).

19 Z. Lian, Q. Yang, Q. Zeng, *et al.*, "Webfed: Cross-platform federated learning framework based on web browser with local differential privacy," in *ICC 2022 - IEEE International Conference on Communications*, 2071–2076 (2022).

20 T. Maurer, A. Mohamed Selim, H. M. T. Alam, *et al.*, "InFL-UX: A Toolkit for Web-Based Interactive Federated Learning," in *Companion Proceedings of the 17th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, *EICS '25 Companion*, 65–67, Association for Computing Machinery, (New York, NY, USA) (2025).

21 K. A. Bonawitz, V. Ivanov, B. Kreuter, *et al.*, "Practical secure aggregation for federated learning on user-held data," in *NIPS Workshop on Private Multi-Party Machine Learning*, (2016).

22 R. Islamov, M. Safaryan, and D. Alistarh, "AsGrad: A sharp unified analysis of asynchronous-SGD algorithms," in *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, S. Dasgupta, S. Mandt, and Y. Li, Eds., *Proceedings of Machine Learning Research* **238**, 649–657, PMLR (2024).

23 N. Su and B. Li, "How asynchronous can federated learning be?," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, 1–11 (2022).

24 T. Zhang, L. Gao, S. Lee, *et al.*, "Timelyfl: Heterogeneity-aware asynchronous federated learning with adaptive partial training," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 5064–5073 (2023).

25 L. Leconte, V. M. Nguyen, and E. Moulines, "Favano: Federated averaging with asynchronous nodes," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5665–5669 (2024).

26 J. Liu, J. Jia, T. Che, *et al.*, "Fedasmu: Efficient asynchronous federated learning with dynamic staleness-aware model update," *Proceedings of the AAAI Conference on Artificial Intelligence* **38**, 13900–13908 (2024).

27 R. Vidal and L. Kameni, "A general theory for federated optimization with asynchronous and heterogeneous clients updates," *J. Mach. Learn. Res.* **24** (2024).

28 S. Truex, L. Liu, K.-H. Chow, *et al.*, *LDP-Fed: federated learning with local differential privacy*, 61–66. Association for Computing Machinery, New York, NY, USA (2020).

29 M. Abadi, A. Agarwal, P. Barham, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015). Software available from tensorflow.org.

30 D. Smilkov, N. Thorat, Y. Assogba, *et al.*, "Tensorflow.js: Machine learning for the web and beyond," *CoRR* **abs/1901.05350** (2019).

31 A. G. Howard, M. Zhu, B. Chen, *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR* **abs/1704.04861** (2017).

32 J. Ángel Morell and E. Alba, "Dynamic and adaptive fault-tolerant asynchronous federated learning using volunteer edge devices," *Future Generation Computer Systems* **133**, 53–67 (2022).

33 K. He, X. Zhang, S. Ren, *et al.*, "Deep residual learning for image recognition," *CoRR* **abs/1512.03385** (2015).

34 A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929* (2020).

35 J. Deng, W. Dong, R. Socher, *et al.*, "ImageNet: A large-scale hierarchical image database,"

in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255 (2009). ISSN: 1063-6919.

36 B. Wu, C. Xu, X. Dai, *et al.*, "Visual Transformers: Token-based Image Representation and Processing for Computer Vision," (2020). arXiv:2006.03677 [cs].

37 A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, (2021).

38 O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision* **115**, 211–252 (2015).

39 J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd ed. (1988).

40 A. Rossberg, "WebAssembly Core Specification," tech. rep., W3C (2022). Version Number: 2.0.

41 S. Becker, A. Alabbas, and J. Bell, "Indexed database api 3.0," W3C Working Draft, W3C (2025).

42 M. Grinberg, *Flask web development: developing web applications with python*, " O'Reilly Media, Inc." (2018).

43 B. McMahan, E. Moore, D. Ramage, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 1273–1282, PMLR (2017).

44 Y. Zhao, M. Li, L. Lai, *et al.*, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582* (2018).

45 S. P. Karimireddy, S. Kale, M. Mohri, *et al.*, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*, 5132–5143, PMLR (2020).

46 Z. Wang, Z. Zhang, Y. Tian, *et al.*, "Asynchronous federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications* **21**(9), 6961–6978 (2022).

47 X. Li, M. Jiang, X. Zhang, *et al.*, "Fed{bn}: Federated learning on non-{iid} features via local batch normalization," in *International Conference on Learning Representations*, (2021).

48 M. A. Kadir, H. M. T. Alam, D. Srivastav, *et al.*, "Partial image active annotation (piaa): An efficient active learning technique using edge information in limited data scenarios," *KI-Künstliche Intelligenz* , 1–12 (2024).

49 M. A. Kadir, H. M. T. Alam, and D. Sonntag, "Edgeal: an edge estimation based active learning approach for oct segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 79–89, Springer (2023).

50 R. R. Selvaraju, M. Cogswell, A. Das, *et al.*, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 618–626 (2017).

51 H. M. T. Alam, D. Srivastav, M. A. Kadir, *et al.*, "Towards interpretable radiology report generation via concept bottlenecks using a multi-agentic rag," in *European Conference on Information Retrieval*, 201–209, Springer (2025).

52 H. M. T. Alam, D. Srivastav, A. Mohamed Selim, *et al.*, "Cbm-rag: Demonstrating enhanced interpretability in radiology report generation with multi-agent rag and concept bottleneck

models," in *Companion Proceedings of the 17th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 59–61 (2025).

List of figure captions

List of Figures