Brain Tumor Detection from MRI images

Rohini Chatterjee
Electrical and Computer engineering
Colorado State university
Fort Collins, USA
rohini12@colostate.edu

Muddasir Attar
Electrical and Computer engineering
Colorado State university
Fort Collins, USA
muddasir@colostate.edu

***Abstract*— Medical image data, including brain MRI scans, contain the subjects' personal information and cannot be shared or used without special authorization. This makes it challenging to analyze medical photos from many sites in an aggregated way. A collaborative learning paradigm known as federated learning (FL) can create an objective global model by gathering updates from local models that have been trained using client data while maintaining the confidentiality of the client's local data. This study uses Federated Learning to identify brain tumors from MRI images in order to address the problem of centralized data collecting. The diagnosis and screening of neuropsychiatric illnesses can then be aided, and a robust federated model with improved performance, can be produced for use in the future. Compared to the conventional deep learning approach, the FL approach can accomplish privacy-protected tumor classification from MRI scans without significantly sacrificing accuracy. When we tested the model using federated learning, we achieved a better accuracy for unseen data for Raspberry Pi1 and Raspberry Pi2 model with a significant increase of 9% for the two models.**

Keywords—federated learning, MRI, confidential, brain tumor

I. Introduction

We are proposing a way to analyze Brain Tumor Detection from the MRI images and we have used federated learning to ensure data privacy for this approach. Federated Learning (FL) technology is developing as an effective remedy in such a dynamic environment. Federated Learning is a machine learning framework that allows data scientists to train statistical models using sensitive data from users, without uploading user data to servers. It is a distributed training technique where training and testing occur locally (at the edge) and only the meta-information is sent to a central server that combines multiple model updates (from multiple clients) into a new model. This model will help the doctors to accurately predict brain tumors in MRI images and it will also help to make the treatment faster along with privacy concern of patients. It will also help them to take better decisions to detect brain tumor. We have augmented the dataset to create a larger dataset and then we have trained it. Then we have converted it into tflite. Then we have created two separate Jupiter notebooks treating them as Raspberry Pi 1 and 2. In the global model we have compared the accuracy with the models. The weights were averaged from the models and the accuracy was tested on unseen data without sharing the data.

II. Related Work

We implemented the building blocks of Federated Learning (FL) and trained one from scratch on the Brain Tumor MRI data set on TenserFlow. We have not used the exact approach described in the references. We have not used any predefined Federated Learning approach for our project. We have not used the federated learning code on the TensorFlow website because we got a lot of errors while using but we have implemented the logic of Federated Learning by training the model, sharing the weights to the Raspberry Pi1 and Raspberry Pi 2 from the main model and then we took these weights from the Raspberry Pi1 and 2 and averaged the weights. Then we tested the model using the updated weights in the global model and we achieved better accuracy from the baseline model. We tried using Pytorch initially but we were not able to successfully execute that. The existing codes don't use federated learning in their models.

III. Data

The dataset that we have used is Brain Tumor MRI dataset that is available on Kaggle. The dataset uses 253 images of brain tumors as 'yes' and 'no' according to the presence or absence of the tumor. It is split into 155 images for brain tumor and 98 images for no brain tumor. We split the data into 60% for the main model into training and validation and 20% for Raspberry Pi1 for training, testing and validation and 20% for Raspberry Pi2 for training, testing and validation. Then, we created labels for main model and Raspberry Pi1 and 2 data. they were labeled as "0" and "1" according to the presence or absence of the tumor and saved them as csv files, namely, main_data, rasp1_data and rasp2_data as shown in Figure 1 and Figure 3. CNN model works better with a balanced dataset, but there were more images in the 'Yes' class than in the 'No' class in the dataset. Then, the data was augmented by randomly flipping and rotating the images for the main data as shown in Figure 2.

	A	B
1	filename	labels
2	./brain_tumour_dataset/no\23 no.jpg	0
3	./brain_tumour_dataset/no\N21.jpg	0
4	./brain_tumour_dataset/yes\Y86.JPG	1
5	./brain_tumour_dataset/no\N16.jpg	0
6	./brain_tumour_dataset/yes\Y193.JPG	1
7	./brain_tumour_dataset/yes\Y107.jpg	1
8	./brain_tumour_dataset/yes\Y161.JPG	1
9	./brain_tumour_dataset/yes\Y245.jpg	1
10	./brain_tumour_dataset/yes\Y8.jpg	1
11	./brain_tumour_dataset/yes\Y44.JPG	1
12	./brain_tumour_dataset/yes\Y41.jpg	1
13	./brain_tumour_dataset/yes\Y76.jpg	1
14	./brain_tumour_dataset/no\No13.jpg	0
15	./brain_tumour_dataset/yes\Y244.JPG	1
16	./brain_tumour_dataset/no\26 no.jpg	0
17	./brain_tumour_dataset/no\No14.jpg	0
18	./brain_tumour_dataset/no\15 no.jpg	0
19	./brain_tumour_dataset/no\50 no.jpg	0
20	./brain_tumour_dataset/yes\Y53.jpg	1
21	./brain_tumour_dataset/no\29 no.jpg	0
22	./brain_tumour_dataset/no\41 no.jpg	0
23	./brain_tumour_dataset/yes\Y195.JPG	1
24	./brain_tumour_dataset/no\no 9.png	0
25	./brain_tumour_dataset/no\14 no.jpg	0
26	./brain_tumour_dataset/yes\Y28.jpg	1

Fig 1. Labels Assigned

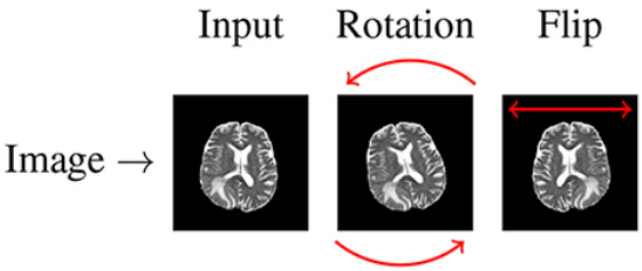

Fig 2. Data Augmentation

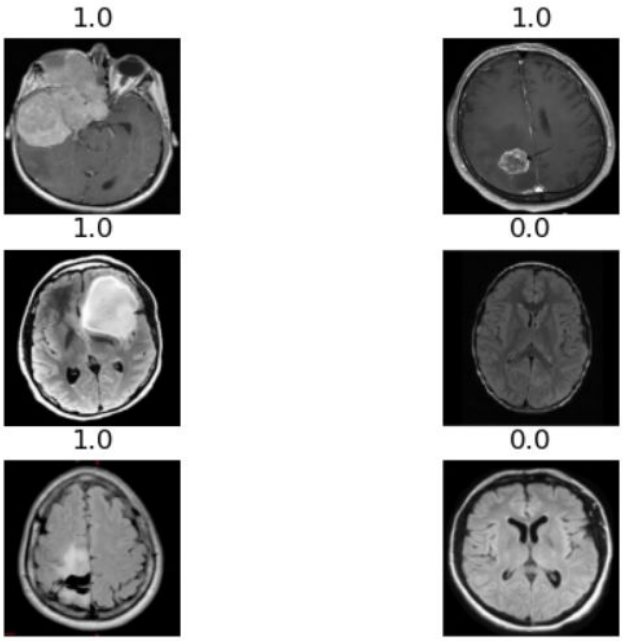

Fig 3. Labels

Using federated learning, we will take an average of the weights after training and feed it to the global model. After the model has been trained, it will be tested using a different dataset than the one we used for the device. The transfer learning will be used in the global model and compare their accuracy over federated learning. In the end, we will use final model which has the best accuracy. We will try to evaluate our model based on accuracy using accuracy metrics and plots and review the size of the model. We will compare the accuracy for the models that are mentioned above.

IV. Technical Approach

In order to produce a larger dataset for the project, we duplicated the current dataset and then we splitted data into main model, Raspberry pi1 and Raspberry Pi2 then we created the labels for the splitted data as '0' for no tumor and '1' for yes tumor images. Then further we divided the data into train and validation data for main model and train, test and validation for Raspberry pi 1 and Raspberry pi2 then we saved them as csv file for future use in models. At the first we trained the main model on main data and then we used the weights generated after training for Raspberry pi1 and raspberry pi2 training then the weights generated after raspberry pi1 and raspberry pi2 were taken and then we averaged the two weights and saved them as global weights. Then using this global updated weights we tested the some unseen data on Raspberry pi1 and Raspberry pi 2. To do this, transfer learning has been used. We have evaluated our model based on accuracy plots.. We considered using GAN for data augmentation but as the dataset was very small, it was not performing well so we went ahead with normal augmentation using data generator. Initially, we used Federated learning by Tensorflow on Tensorflow website but due to some version mismatch, we were not able to do so. Then we also considered working on Pytorch but we couldn't execute it properly. We switched to working with normal Convolution Neural Networks. Here, we tried using different pretrained models as ResNet50, VGG and, ImageNet but we couldn't achieve desirable accuracy so we have defined our own model using Convolutional layers.

Data Augmentation: Data augmentation is a process that artificially generates more images while the primary input data remain the same. It produces more variations of the training data that make the model training more robust and complete. In this we have horizontally flipped the image and random rotation by 20%.

Transfer Learning: Transfer learning is the process of using features discovered while solving one problem to solve another that is somewhat related. Transfer learning is typically used for problems where the dataset contains insufficient data to fully train a model from scratch.

Federated Learning: By separating the capacity to do machine learning from the requirement to put the training data in the cloud, federated learning enables mobile devices/embedded platforms to cooperatively develop a shared prediction model while maintaining all of the training data on the device. By bringing model training to the device as well, this extends beyond the usage of local models that make predictions on mobile devices.

Convolutional neural network: A convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), most commonly applied to analyze visual imagery. CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters

that slide along input features and provide translation-equivariant responses known as feature maps.

Pruning and Quantization: Pruning is a strategy that focuses on getting rid of part of the model weights to shrink the model and lower the amount of inference needed. It has been demonstrated that pruning can significantly increase efficiency while reducing the performance reduction (prediction quality).

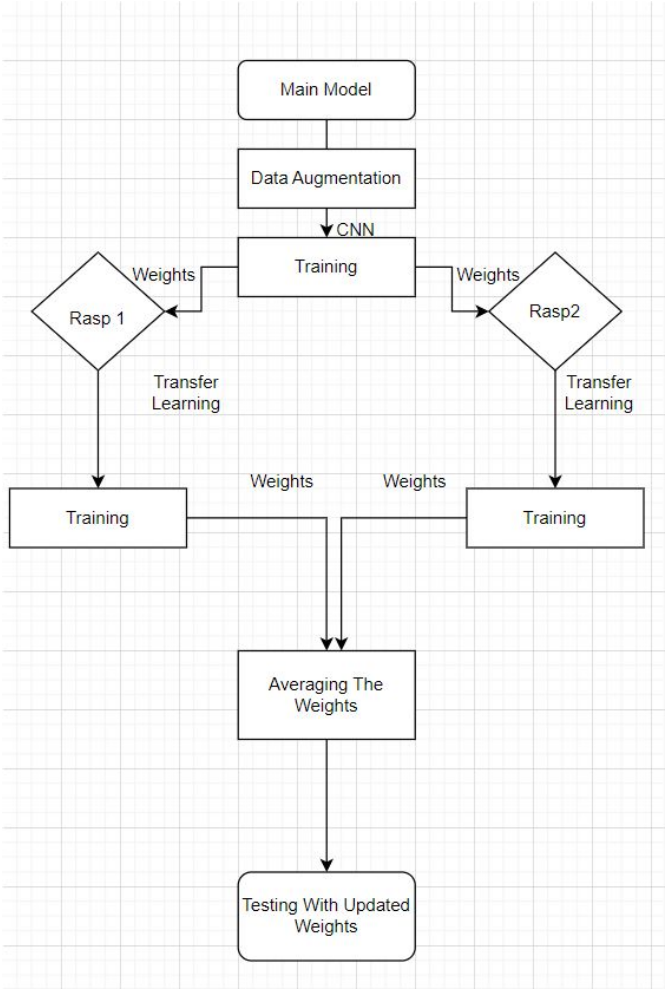

Fig 4. Federated Learning Algorithm

A. CNN Architectures

For our main model, we have rescaled the input image. Then we have used CNN with three convolution layers with Maxpooling2D that downsamples the input along spatial dimensions. For the first convolution layer, we have used 8 filters across three channels. For the second convolution layer, we have used 16 filters across three channels. For the third convolution layer, we have used 32 filters across three channels. We have tried using average pooling with the convolution layers which resulted in a bad accuracy so we again switched to Maxpooling2D. We have used 'relu' as an activation function. We have used Flatten layer to flatten the input dimensions and two dense layers with 32 and 1 neurons, respectively. As we have two classes in our dataset which is 'yes' and 'no' so the last dense layer has 1 neuron. We tried working with different optimizers such as SGD and Adam, where we observed that we got a better accuracy using Adam. For loss, we have used BinaryCrossentropy as there is a classification problem between 2 categories. Figure 5 shows the general layer plot of the CNN model.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 224, 224, 3)       0

cnn1 (Conv2D)                (None, 222, 222, 8)       224

max_pooling2d (MaxPooling2D  (None, 111, 111, 8)       0
)

cnn2 (Conv2D)                (None, 109, 109, 16)      1168

max_pooling2d_1 (MaxPooling  (None, 54, 54, 16)        0
2D)

cnn3 (Conv2D)                (None, 52, 52, 32)        4640

max_pooling2d_2 (MaxPooling  (None, 26, 26, 32)        0
2D)

flatten (Flatten)            (None, 21632)             0

dense1 (Dense)               (None, 32)                692256

dense2 (Dense)               (None, 1)                 33

=================================================================
Total params: 698,321
Trainable params: 698,321
Non-trainable params: 0
```

Fig 5. CNN Architecture of Main Model

For Raspberry Pi1 and Raspberry Pi2, we have used transfer learning and we have used the weights from the main model using load_weights and for the model, we have added extra convolution and dense layers in the model. We have used trainable as false for other convolution layers that we had in our main model.

V. EXPERIMENTS

The model was built using Keras and Tensorflow 2.0 and used GPU for this experimental setup.

$$Weights_{Global} = \frac{Weights_{Raspbeery\ Pi1} + Weights_{Raspberri\ Pi2}}{2}$$

Here, the $Weights_{global}$ define the weight of the global model and $Weights_{RaspberryPi1}$ and $Weights_{RaspberryPi2}$ define the weight of the Raspberry Pi1 and Raspberry Pi 2 accordingly.

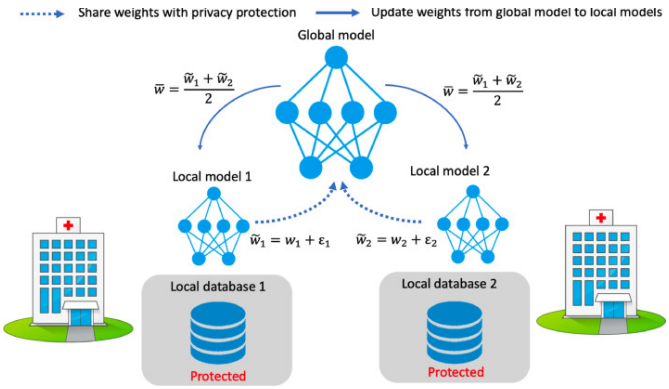

Fig 6. Weights Averaging

In our global model, we have defined two models, mainly named, 'simple' and 'extra _dense_CNN'. We have defined model as getmodel() so that we can call the models which we need for the main model, RaspberryPi1 and RaspberryPi2, in model.fit(). Also, local epochs will be taken care of by epoch argument in model.fit(). Starting with our notebook, we have defined data augmentation and then our sequential models for 'simple' and 'extra _dense_CNN'. Then we have called our simple model for main model and then the weights generated for this model are used for Raspberry Pi1 and Raspberry Pi2 using get_weights(). In our proposed system, for Raspberry Pi1 and Raspberry Pi2, we have called model 'extra _dense_CNN' using getmodel(). The weights generated from Raspberry Pi1 and Raspberry Pi2 are defined as wt_rasp1 and wt_rasp2 using get_weights(). Then we have performed the average on wt_rasp1 and wt_rasp2 and updated the weights in the main model using set_weights(). Then we have tested the accuracy of Raspberry Pi1 and Raspberry Pi2 using the updated weights and compared it with accuracy of the Raspberry Pi1 and Raspberry Pi2 without the updated weights. We tried using more epochs but since our data is small, our model was overfitting so we reduced the number of epochs to 10. The drawbacks of our proposed system is sometimes the accuracy might decrease for federated model. The model is also overfitting because of small dataset. The other main drawback is when a client is having data that consists of more number of 'no tumor' data might affect the accuracy of the main model when the weights are shared. This uneven class distribution adds more challenges to the learning procedure. As per federated learning, it requires computation to be done on the client's or owner's device that holds the data. For some devices which have less computational capacity, this may not be possible.

Our proposed system has not used ensemble architecture for federated learning like it has been usen in the published papers. In the published papers, the authors have imported VGG16, VGG19, Inception V3, ResNet50, DenseNet121, and Xception CNN models using TensorFlow and Keras libraries and have used the average of all the models for the main model but we haven't used any predefined models. We have defined our own CNN model.

To validate the FL model's model efficiency, the training and validation accuracy was measured which is plotted in Figures.

For the main model, we have achieved training accuracy of around 88%. Figure 7 illustrates the training and validation loss and accuracy for the main model.

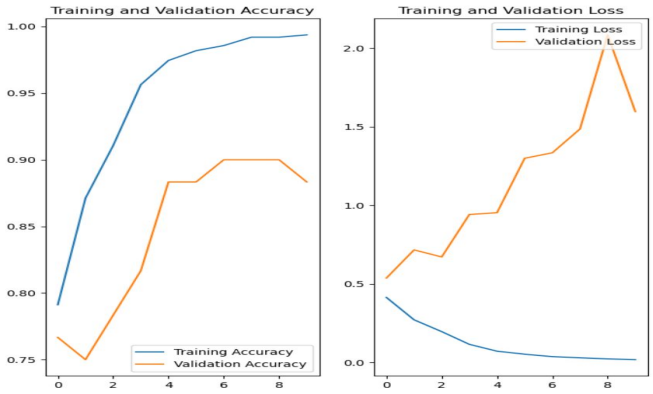

Fig 7. Training and Validation Accuracy on main model

For Raspberry Pi1, we have achieved training accuracy of around 89% on unseen data. Figure 7 illustrates the training and validation loss and accuracy for the Raspberry Pi1.

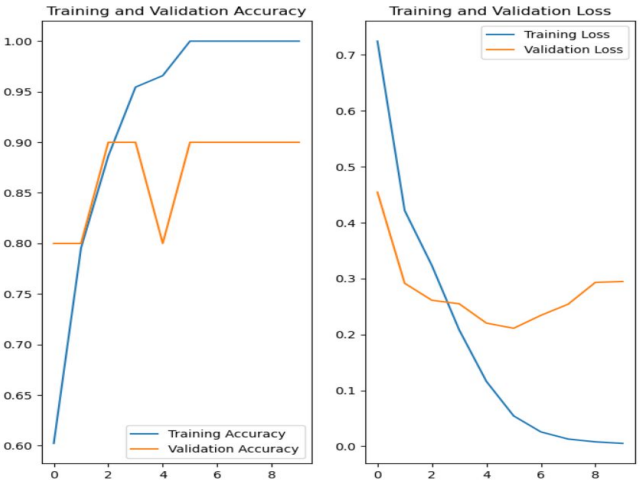

Fig 8.. Training and Validation Accuracy on Raspberry Pi1

For Raspberry Pi2, we have achieved training accuracy of around 80% on unseen data. Figure 9 illustrates the training and validation loss and accuracy for the Raspberry Pi2.

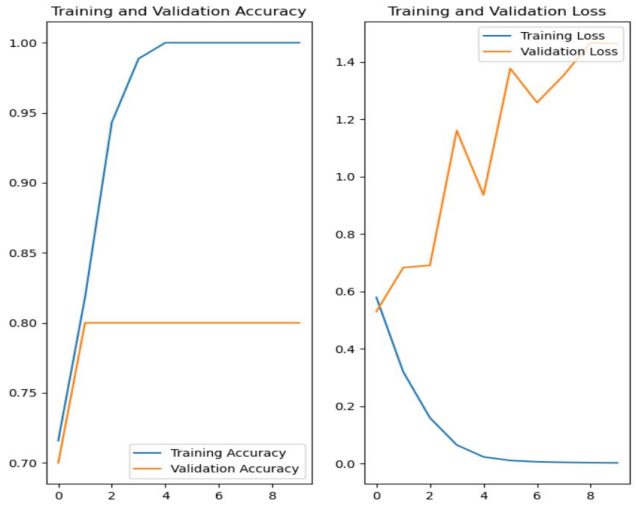

Fig 9. Training and Validation Accuracy on Raspberry Pi

VI. Conclusion

Federated Learning is a machine learning framework that allows data scientists to train statistical models using sensitive data from users, without uploading user data to servers. It is a distributed training technique where training and testing occur locally (at the edge) and only the meta-information is sent to a central server that combines multiple model updates (from multiple clients) into a new model.

According to federated learning, we have trained the main model on Brain MRI dataset. We have got the training accuracy of around 88%. Then we used the weights from the main model and transferred it to the Raspberry Pi1. We trained Raspberry Pi1 model and we got training accuracy of 89% on unseen data. We trained Raspberry Pi2 and we got training accuracy of 80% on unseen data. When we tested the Raspberry Pi1 on without updating the weights, we achieved an accuracy of 80% and when we tested Raspberry Pi1 after updating the weights, an accuracy of 89% was achieved. When we tested the Raspberry Pi2 model on without updating the weights, we achieved an accuracy of 80% and when we tested Raspberry Pi1 after updating the weights, an accuracy of 89% was achieved. When we tested the model using federated learning, we achieved a better accuracy for unseen data for Raspberry Pi1 and Raspberry Pi2 with a significant increase of 9% for the two models. The accuracy for both the Raspberry Pi1 and Raspberry Pi2 changes every time when we train or test.

The algorithm maintains the clients' data privacy as the global model does not receive any clients' data besides the updated weight of local models. Statistical models are used to power apps like next-word prediction, facial recognition, and voice recognition by studying user behavior across a large pool of mobile phones. Users can choose not to share their data in order to maintain their privacy or to save data or battery life on their phone. Without disclosing private information or degrading the user experience, federated learning can produce precise smartphone predictions. Thus, a successful FL implementation could have a significant impact on the ability to practice precision medicine on a large scale, resulting in models that produce objective judgments, accurately reflect the physiology of an individual, are sensitive to rare diseases, and respect governance and privacy concerns. Federated Learning is a promising method to obtain strong, accurate, safe, robust, and unbiased models because all ML methods greatly benefit from having access to data that roughly approximates the true global distribution. FL effectively addresses concerns regarding the egress of sensitive medical data by allowing multiple parties to train cooperatively without the need to exchange or centralize data sets. As a result, it might create fresh opportunities for business and research and could enhance patient care all over the world.

However, FL already has an effect on almost all stakeholders and the entire treatment cycle. Examples include better medical image analysis giving clinicians better diagnostic tools, over true precision medicine by helping to find similar patients, and collaborative and accelerated drug discovery cutting down cost and time-to-market for pharmaceutical companies. FL will undoubtedly continue to be a topic of active research over the following ten years because not all technical questions have yet been resolved. Despite this, we firmly believe that it has a very promising future in terms of its potential impact on precision medicine and ultimately improving medical care.

In the creation of smart cities, federated learning (FL) is crucial. Big data and artificial intelligence have given rise to problems with data privacy and protection that FL can address. The most recent advancements in FL and its applications in various fields are examined in this essay. The most recent research on the use of FL for various fields in smart cities is discussed with a thorough investigation.

The data-driven medical application has emerged as a promising tool for developing trustworthy and scalable diagnostic and prognostic models from medical data with the introduction of the IoT, AI, and ML/DL algorithms. In recent years, this has drawn a lot of interest from both academia and industry. There is no doubt that this has raised the standard of healthcare delivery. However, these AI-based medical applications are still not widely used as a result of their

challenges in meeting stringent security, privacy, and service quality requirements (such as low latency). In addition, medical data are frequently fragmented and private, which makes it difficult to produce reliable results across populations. Complex machine-learning models can now be distributedly trained thanks to recent advancements in federated learning (FL).

VII. References

[1] Effectiveness of Federated Learning and CNN Ensemble Architectures for Identifying Brain Tumors Using MRI Images (Moinul Islam· Md. Tanzim Reza · Mohammed Kaosar · Mohammad Zavid Parvez)

[2] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange A Federated Deep Learning Framework for 3D Brain MRI Images (Zhipeng Fan; Jianpo Su; Kai Gao; Dewen Hu; Ling-Li Zeng)

[3] Decentralized Federated Learning for Healthcare Networks: A Case Study on Tumor Segmentation(Bernardo Camajori Tedeschini; Stefano Savazzi; Roman Stoklasa; Luca Barbieri; Ioannis Stathopoulos)

[4] Effectiveness of Federated Learning and CNN Ensemble Architectures for Identifying Brain Tumors Using MRI Images (Moinul Islam· Md. Tanzim Reza · Mohammed Kaosar · Mohammad Zavid Parvez)

[5] Effectiveness of Federated Learning and CNN Ensemble Architectures for Identifying Brain Tumors Using MRI Images (Moinul Islam· Md.
Tanzim Reza · Mohammed Kaosar · Mohammad Zavid Parvez)

[6] A Comprehensive Survey on Federated Learning: Concept and Applications (Dhurgham Hassan Mahlool, Mohammed Hamzah Abed)

[7] Federated Learning: Collaborative Machine Learning without Centralized Training Data

[8] https://www.tensorflow.org/tutorials/images/classification

[9] https://en.wikipedia.org/wiki/Federated_learning

[10] https://www.nature.com/articles/s41746-020-00323-1

[11] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

.