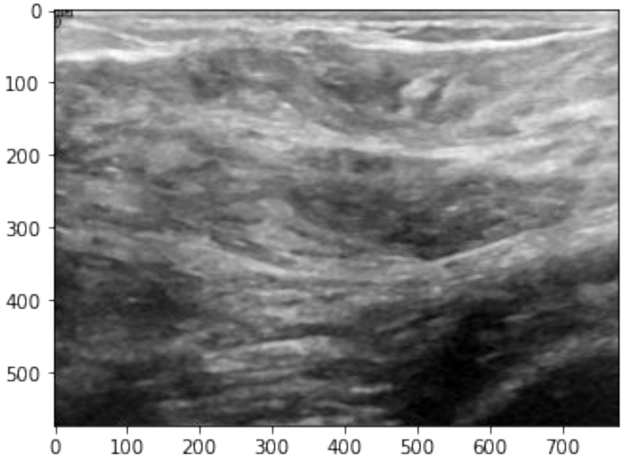
Breast tumor detection using federal learning

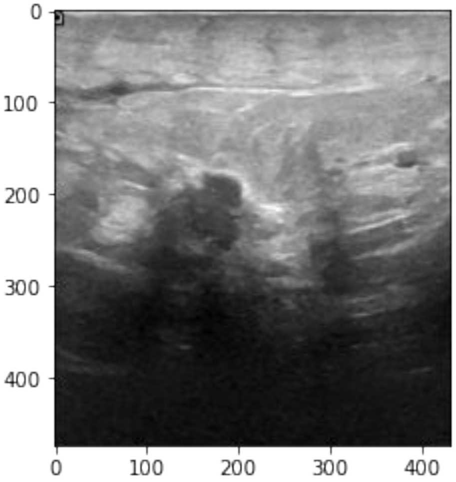
Dataset

The Dataset that we have used is one of the datasets that we have found in kaggle "https://www.kaggle.com/datasets/anaselmasry/datasetbusiwithgt". The datasets in already divided into train and test split. It has test data total of 150 cases from which 84 are labelled as "benign", 42 as "malignant", 24 as "normal". In the training set we have total of 630 casesfrom which 353 labelled as "benign", 168 labelled as "malignant", 109 as "normal". We have excluded out the "benign" labelled caes and went for purely breast cancer detection rather than breast tumor detection.

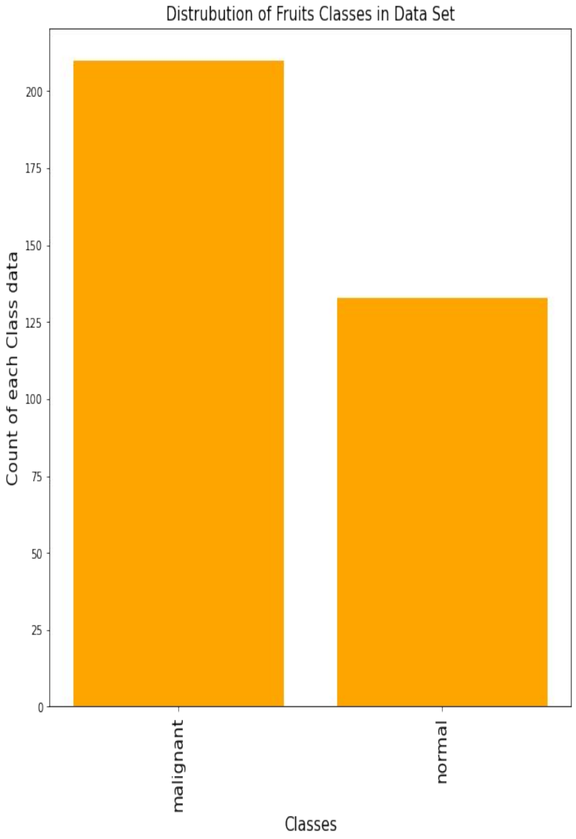
Data



Healthy.



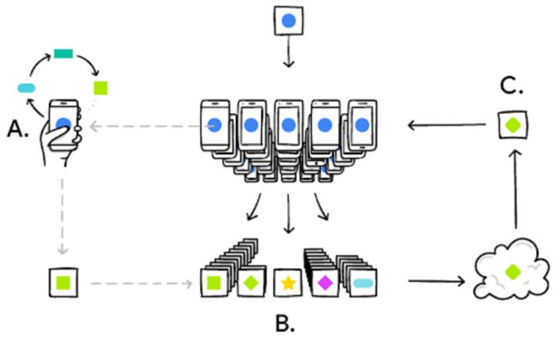
Malignant.



Distribution of data

Federal learning

- So, our centralized machine learning application will have a local copy on all devices, where users can use them according to our need.
- The model will now gradually learn and train itself on the information inputted by the user and become smarter, time to time.
- The devices are then allowed to transfer the training results, from the local copy of the machine learning app, back to the central server.



Creating clients

Creating a list of clients where the x_train and y_train datasets are passed with number of number to be created is 100. We are creating 100 local clients from which the weight of the model will be passed on to the global model. The clients will have randomised data and splits. Each client will be shared data accordingly and train. The data shards are then seperated out into data and label lists.

Global and local model

- So I have used a simple ANN using MLP having.
- An instance of the model is created.
- So use the model as a global model
- I have made a global training loop of 300 iterations. At the first iteration the weight of the global model is served as the weight for all the local models. During each iteration we loop through each client and create a local model and train the data on that local model. Once the model is trained we send the weight of the local model to the global model from there the next local model gets the weight. Before every operation we scale the weights in between 0-1 to reduce the computational expense. Doing this we get and collect all the weights of the local model then average out the weights of all the local models and then fed to the global model. This is done 300 times.

Randomising the data

We randomize the data and send it to every client

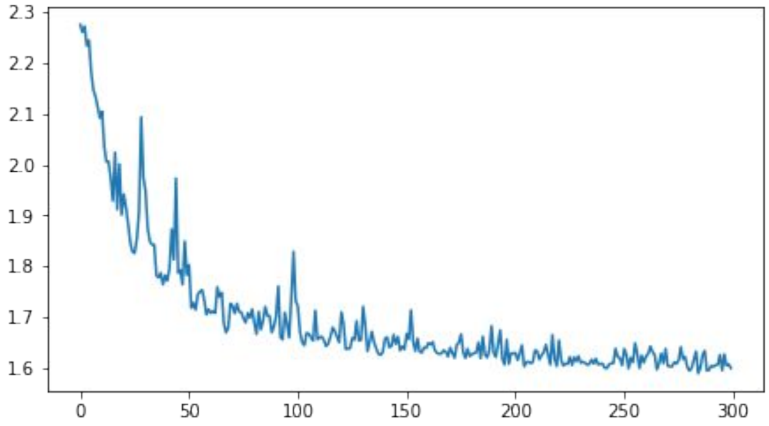
```
#randomize the data
data = list(zip(image_list, label_list))
random.shuffle(data)  # <- IID
```

Results

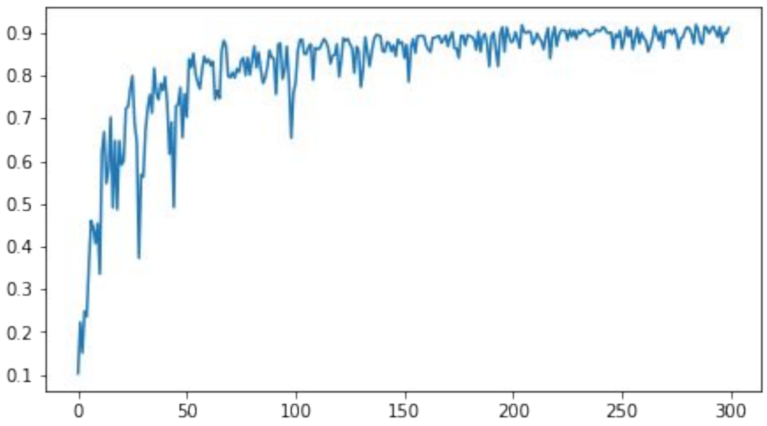
After training the model over 300 iterations and federated learning process of locally and globally training the model but only taking the weights of of the trained model, we got a very good accuracy of global accuracy being 91.119 and global loss being 1.59965.

```
comm_round: 261 | global_acc: 88.214% | global_loss: 1.6277244091033936
comm_round: 262 | global_acc: 85.619% | global_loss: 1.6424388885498047
comm_round: 263 | global_acc: 86.881% | global_loss: 1.6318743228912354
comm_round: 264 | global_acc: 88.548% | global_loss: 1.625700831413269
comm_round: 265 | global_acc: 91.571% | global_loss: 1.596787452697754
comm_round: 266 | global_acc: 89.952% | global_loss: 1.608684778213501
comm_round: 267 | global_acc: 88.214% | global_loss: 1.6285958290100098
comm_round: 268 | global_acc: 90.238% | global_loss: 1.609269618988037
comm_round: 269 | global_acc: 86.500% | global_loss: 1.6376686096191406
comm_round: 270 | global_acc: 90.452% | global_loss: 1.6039811372756958
comm_round: 271 | global_acc: 90.286% | global_loss: 1.6033092737197876
comm_round: 272 | global_acc: 90.714% | global_loss: 1.6026108264923096
comm_round: 273 | global_acc: 89.643% | global_loss: 1.6113004684448242
comm_round: 274 | global_acc: 90.929% | global_loss: 1.6083978414535522
comm_round: 275 | global_acc: 89.524% | global_loss: 1.6150033473968506
comm_round: 276 | global_acc: 86.286% | global_loss: 1.641101360321045
comm_round: 277 | global_acc: 88.714% | global_loss: 1.6173404455184937
comm_round: 278 | global_acc: 89.000% | global_loss: 1.6218324899673462
comm_round: 279 | global_acc: 90.548% | global_loss: 1.6015233993530273
comm_round: 280 | global_acc: 91.381% | global_loss: 1.5945498943328857
comm_round: 281 | global_acc: 90.952% | global_loss: 1.6001704931259155
comm_round: 282 | global_acc: 89.738% | global_loss: 1.6162526607513428
comm_round: 283 | global_acc: 87.500% | global_loss: 1.6324450969696045
comm_round: 284 | global_acc: 91.929% | global_loss: 1.5894263982772827
comm_round: 285 | global_acc: 90.738% | global_loss: 1.5999544858932495
comm_round: 286 | global_acc: 87.857% | global_loss: 1.625375747680664
comm_round: 287 | global_acc: 87.429% | global_loss: 1.6339106559753418
comm_round: 288 | global_acc: 91.571% | global_loss: 1.5947693586349487
comm_round: 289 | global_acc: 91.095% | global_loss: 1.596139907836914
comm_round: 290 | global_acc: 89.905% | global_loss: 1.6045095920562744
comm_round: 291 | global_acc: 90.881% | global_loss: 1.6024580001831055
comm_round: 292 | global_acc: 91.524% | global_loss: 1.605469822883606
comm_round: 293 | global_acc: 90.262% | global_loss: 1.6065346002578735
comm_round: 294 | global_acc: 89.095% | global_loss: 1.625815749168396
comm_round: 295 | global_acc: 91.429% | global_loss: 1.5952544212341309
comm_round: 296 | global_acc: 87.690% | global_loss: 1.6267505884170532
comm_round: 297 | global_acc: 90.024% | global_loss: 1.604494571685791
comm_round: 298 | global_acc: 89.833% | global_loss: 1.607370138168335
comm_round: 299 | global_acc: 91.119% | global_loss: 1.5996589660644531
```

Global accuracy and global loss



Globall loss.



Global accuracy

Conclusion

- Federated Learning seems to have a lot of potentials. Not only it secures user sensitive information, but also aggregates results and identifies common patterns from a lot of users, which makes the model robust, day by day.
- It trains itself as per its user data, keeps it secure, and then comes back as a smarter guy, which is again ready to test itself from its own user! Training and testing became smarter!
- Be it training, testing, or information privacy, Federated Learning created a new era of secured AI.
- Federated Learning is still in its early stages and faces numerous challenges with its design and deployment.