Department of Computer Science and Engineering (CSE)
IIT Hyderabad

CS6450: Visual Computing

FEDERATED SEMI-SUPERVISED MEDICAL IMAGE CLASSIFICATION VIA INTER-CLIENT RELATION MATCHING

Quande Liu, Hongzheng Yang , Qi Dou, and Pheng-Ann Heng

MICCAI2021

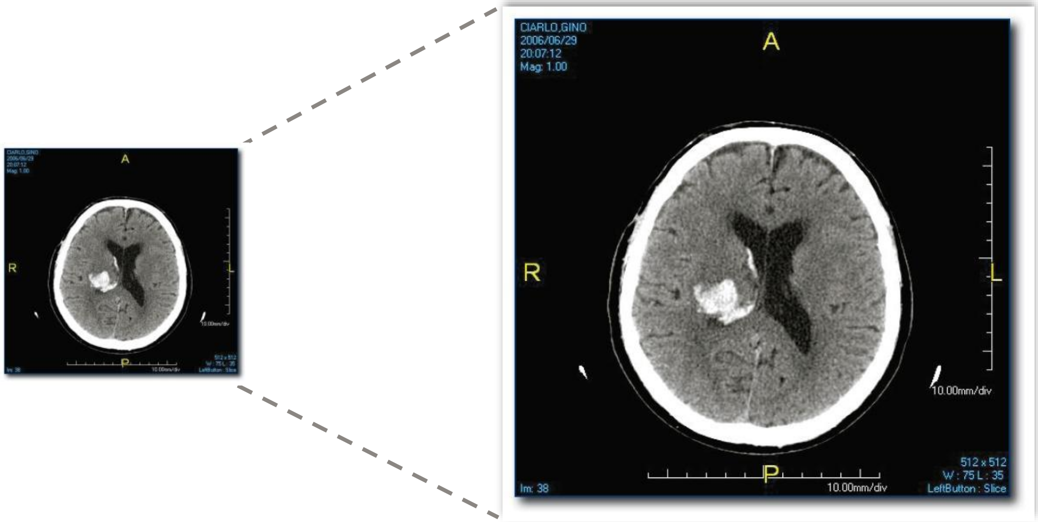
Instructor
Prof. C. Krishna Mohan

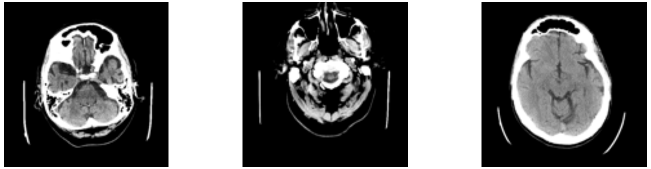
Teaching Assistant
Divya Peketi

Presenter
Kamal Shrestha
cs21mtech16001

OVERVIEW

- Problem & Motivation
- Traditional Approach
- Fed-IRM
- Implementation Details





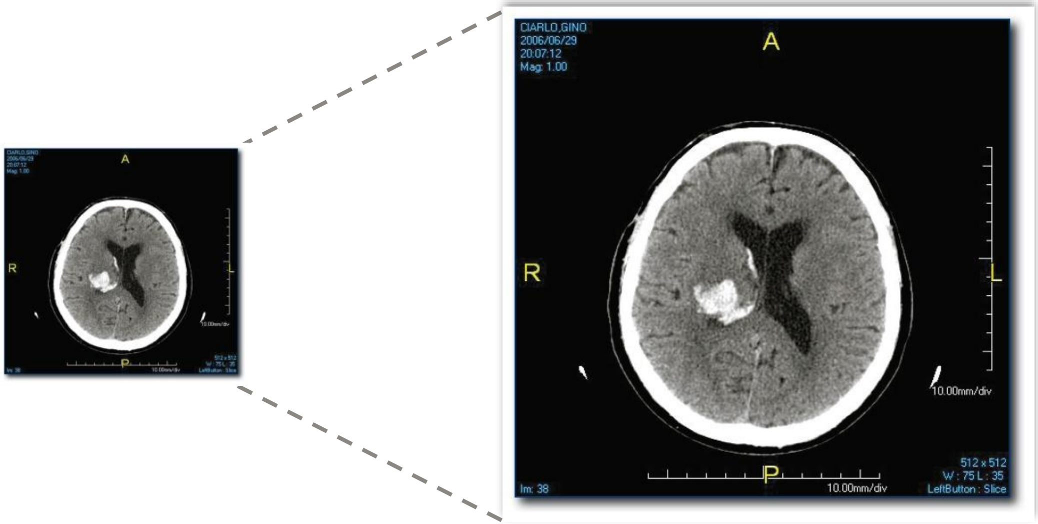
OVERVIEW

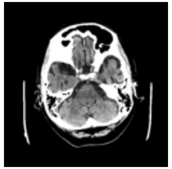
Problem & Motivation

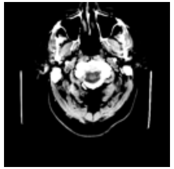
Traditional Approach

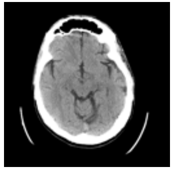
Fed-IRM

Implementation Details

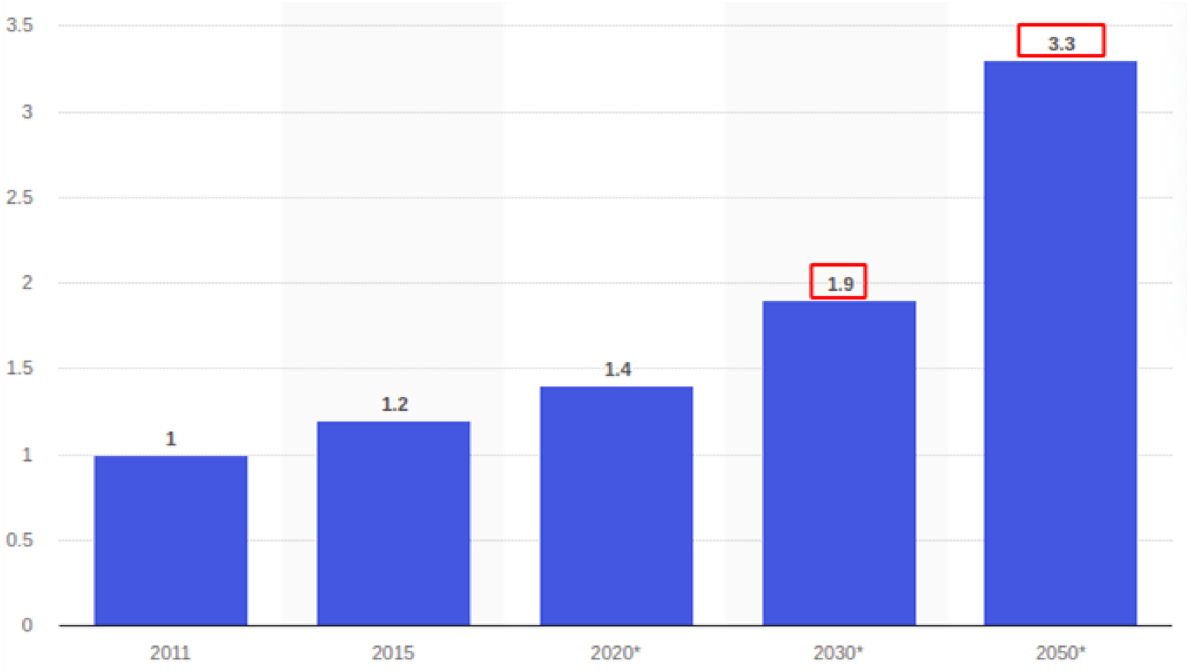






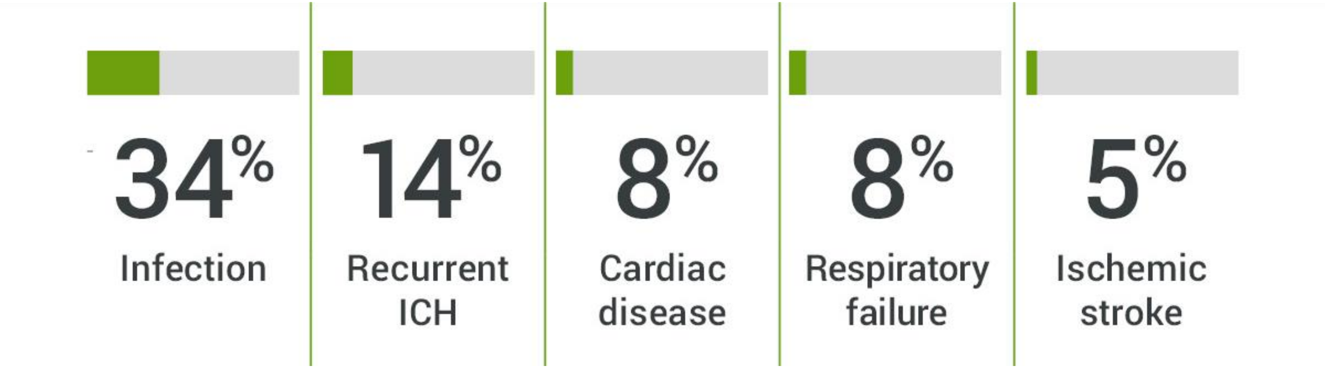


Brain Stroke or Brain Attack kills **more people than HIV, Tuberculosis, and Malaria combined.**



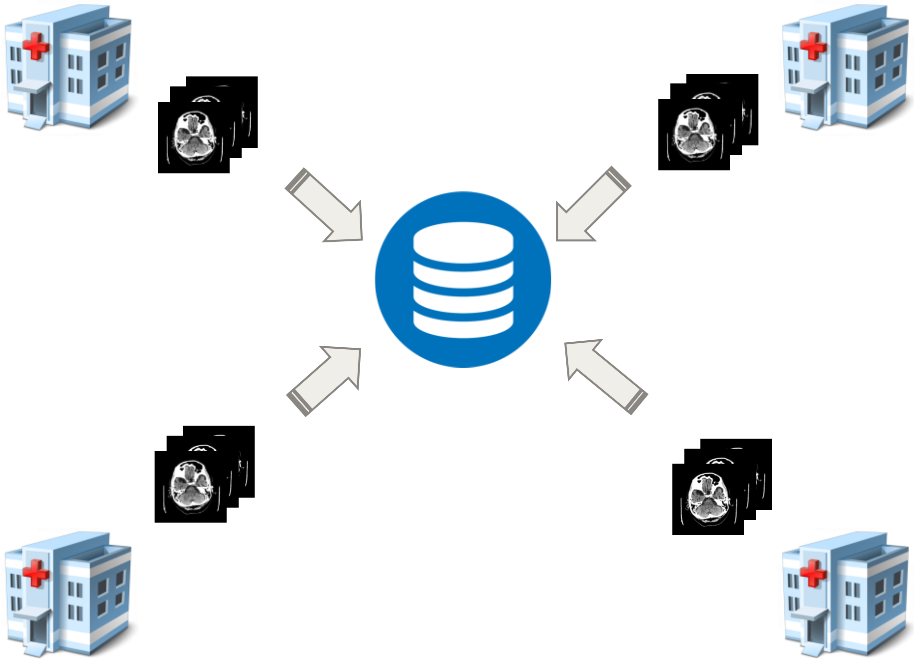
Statista 2022

Brain Stroke kills **more people than HIV, Tuberculosis, and Malaria combined.**



Healio

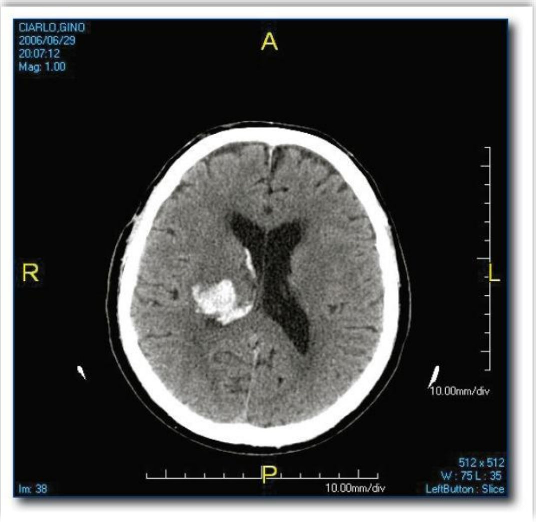
Data collaboration across medical institutions is increasingly desired to **mitigate the scarcity and distribution of medical images.**

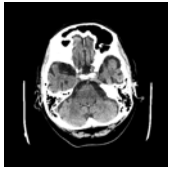


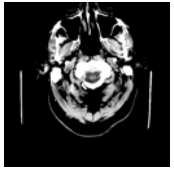
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

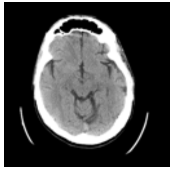
OVERVIEW

- Problem & Motivation
- **Traditional Approach**
- Fed-IRM
- Implementation Details

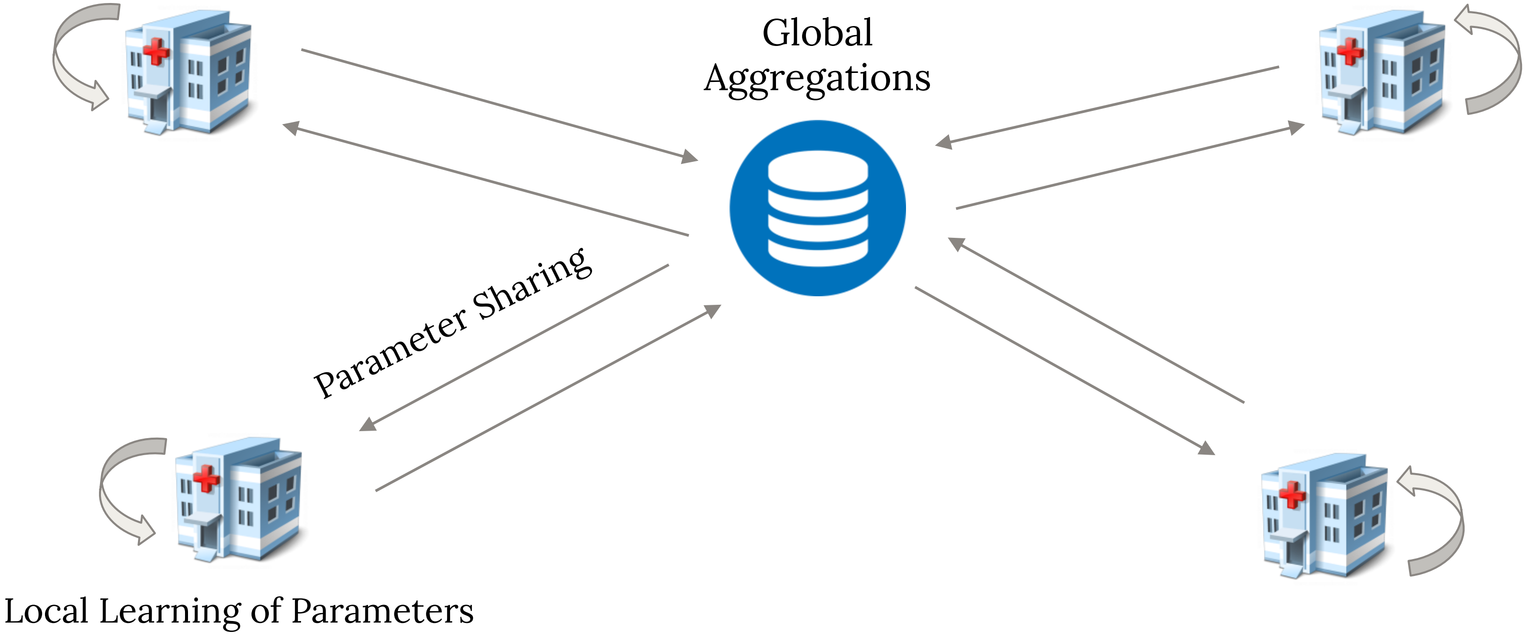






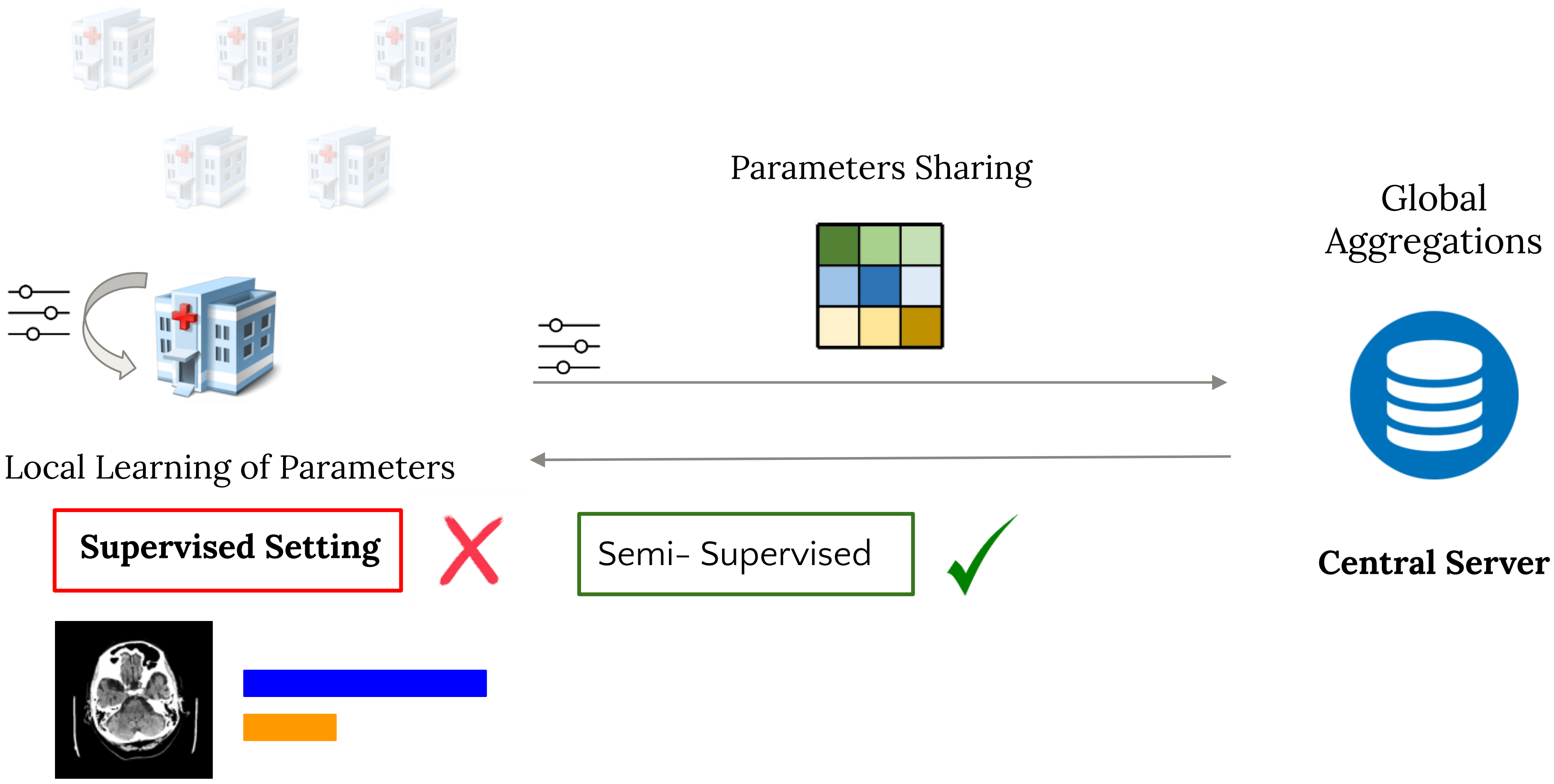


Federated learning (FL) has emerged as **a privacy-preserving solution** to learn models **without exchanging the sensitive health data.**

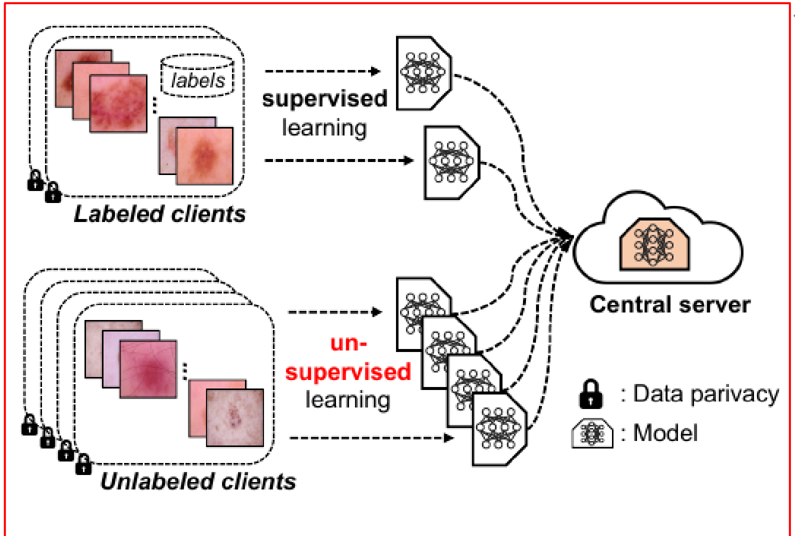


Supervised Setting

Existing FL algorithms typically **only allow the supervised training setting.**

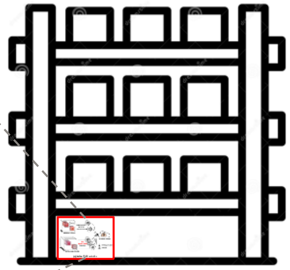


A naive FSSL, solution is to simply integrate the **off-the-rack semi-supervised learning (SSL) methods** onto the federated learning paradigm.

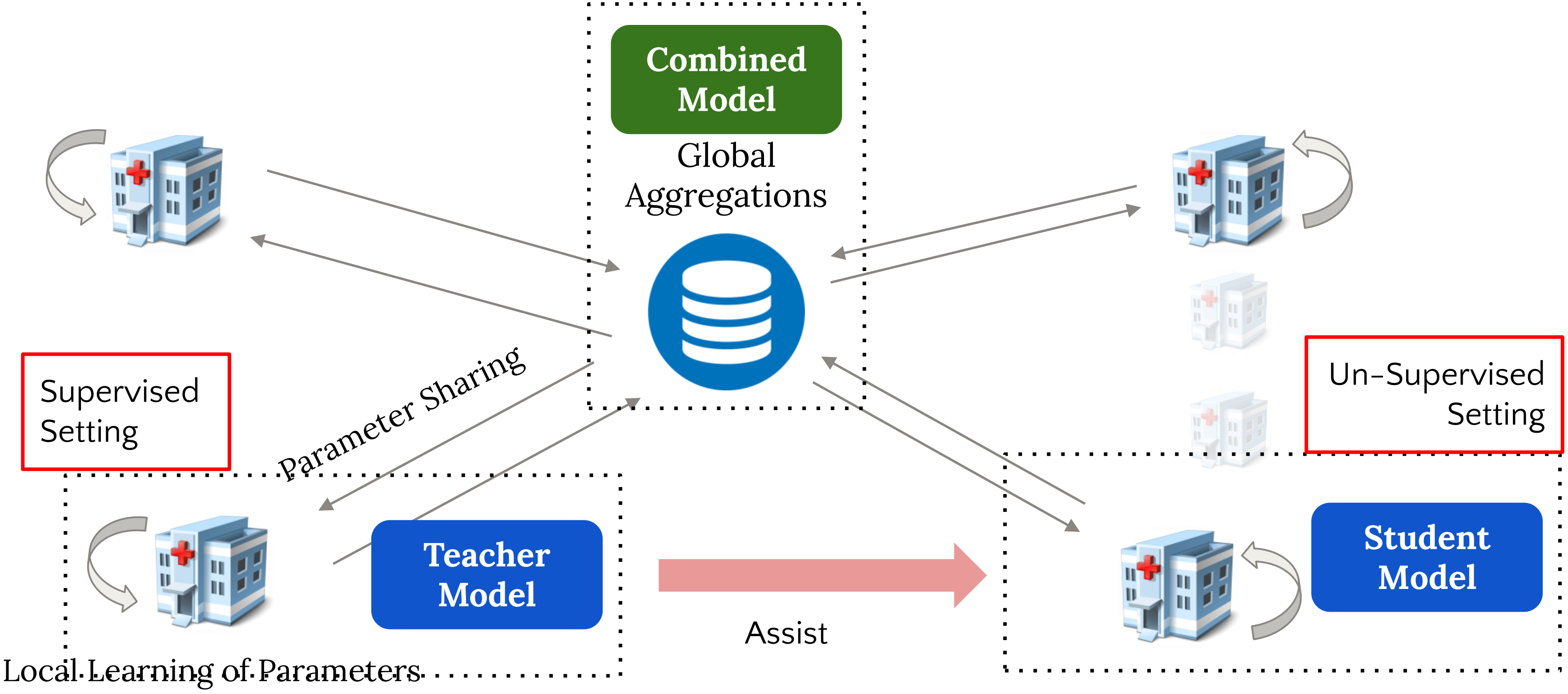


(a) Naive FSSL solution

RACK

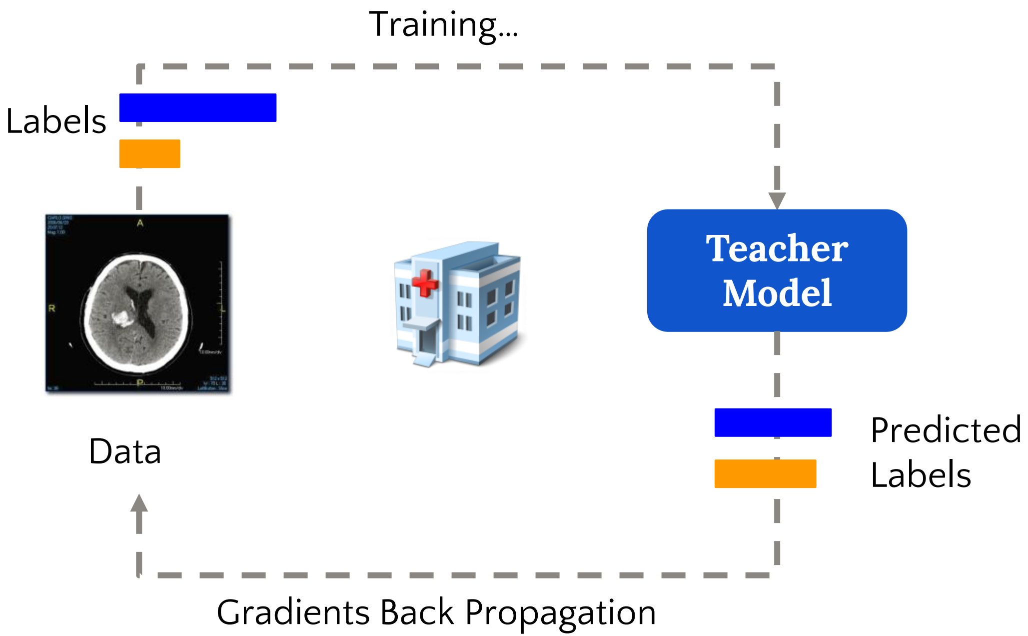


Federated Learning in a Semi Supervised Scenario

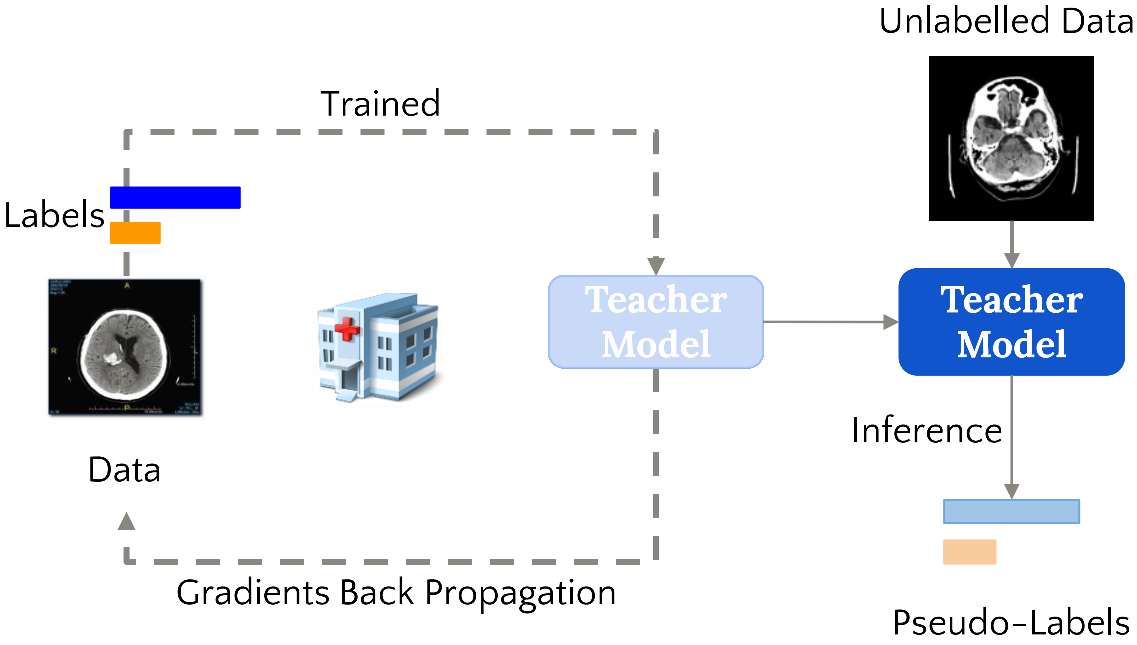


Supervised Clients

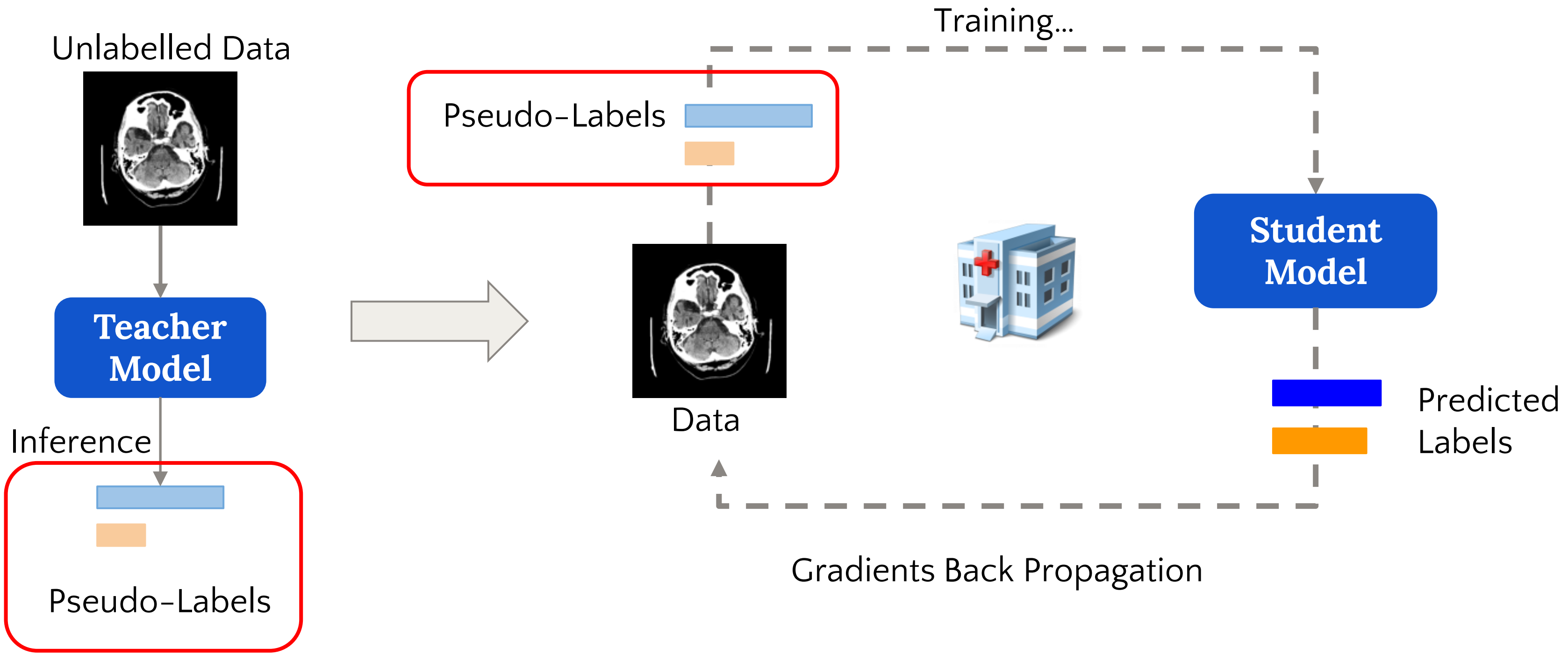
Semi Supervised Scenario: **Teacher Model (Common Supervised Setting)**



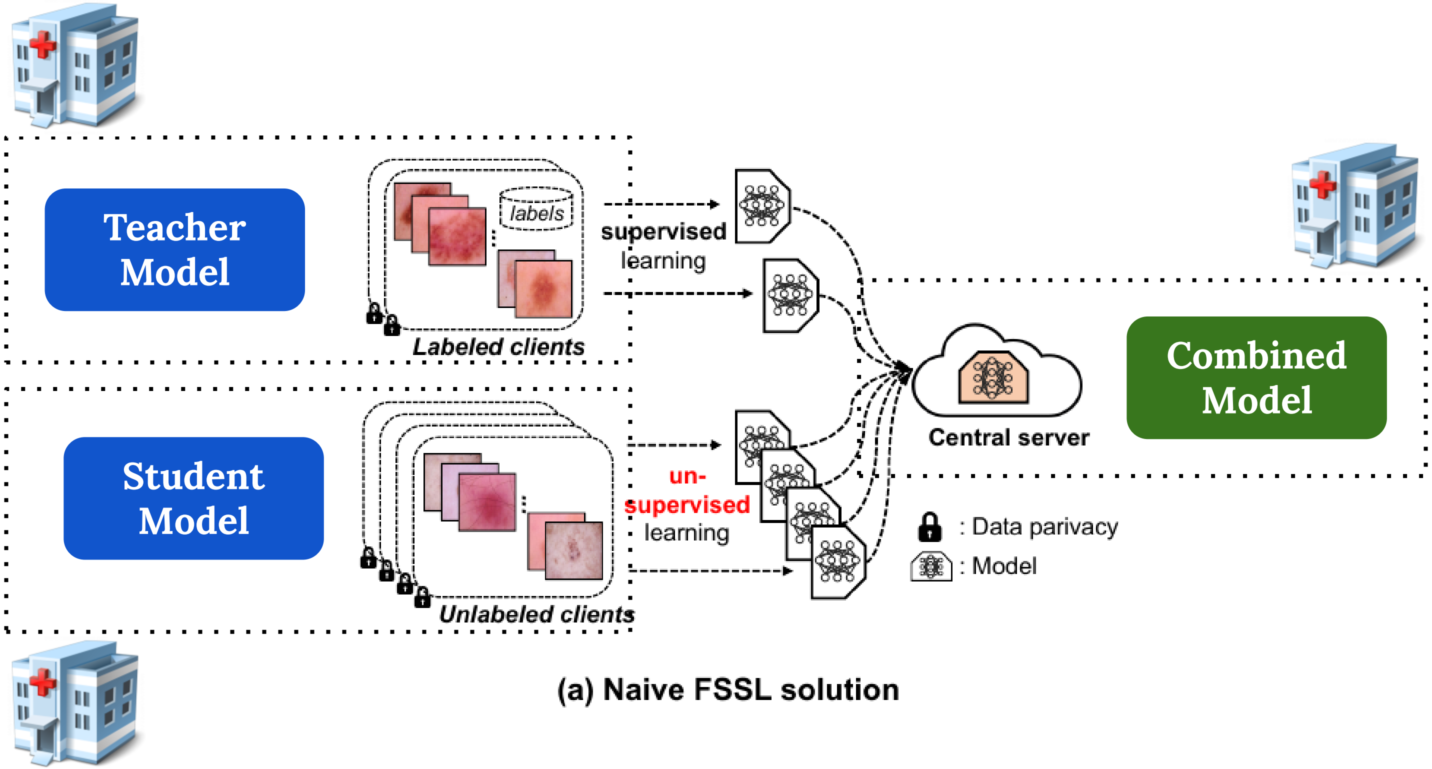
Semi Supervised Scenario: **Student Model**



Semi Supervised Scenario: **Student Model**

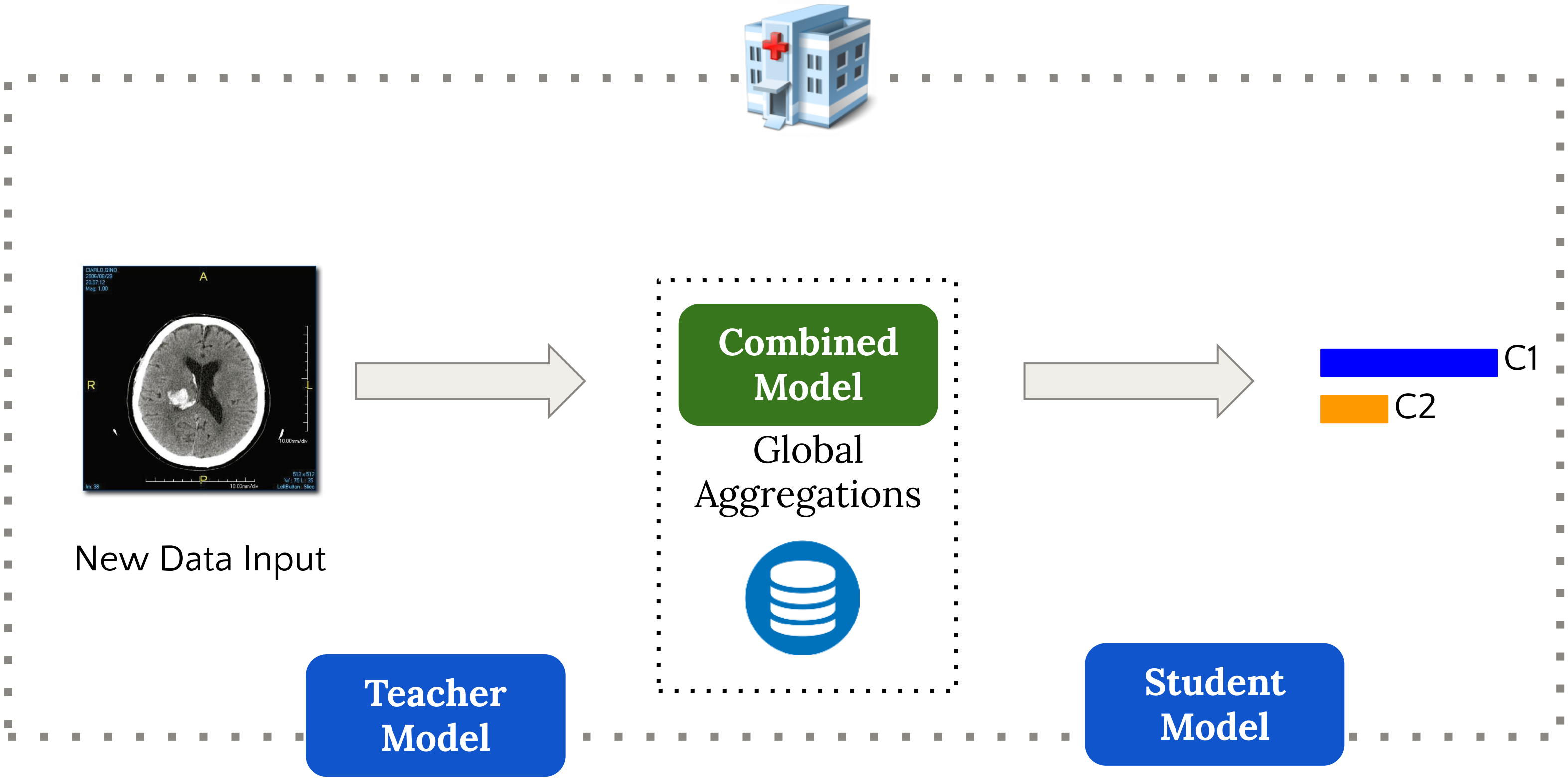


Global Model from Teacher and Student Model

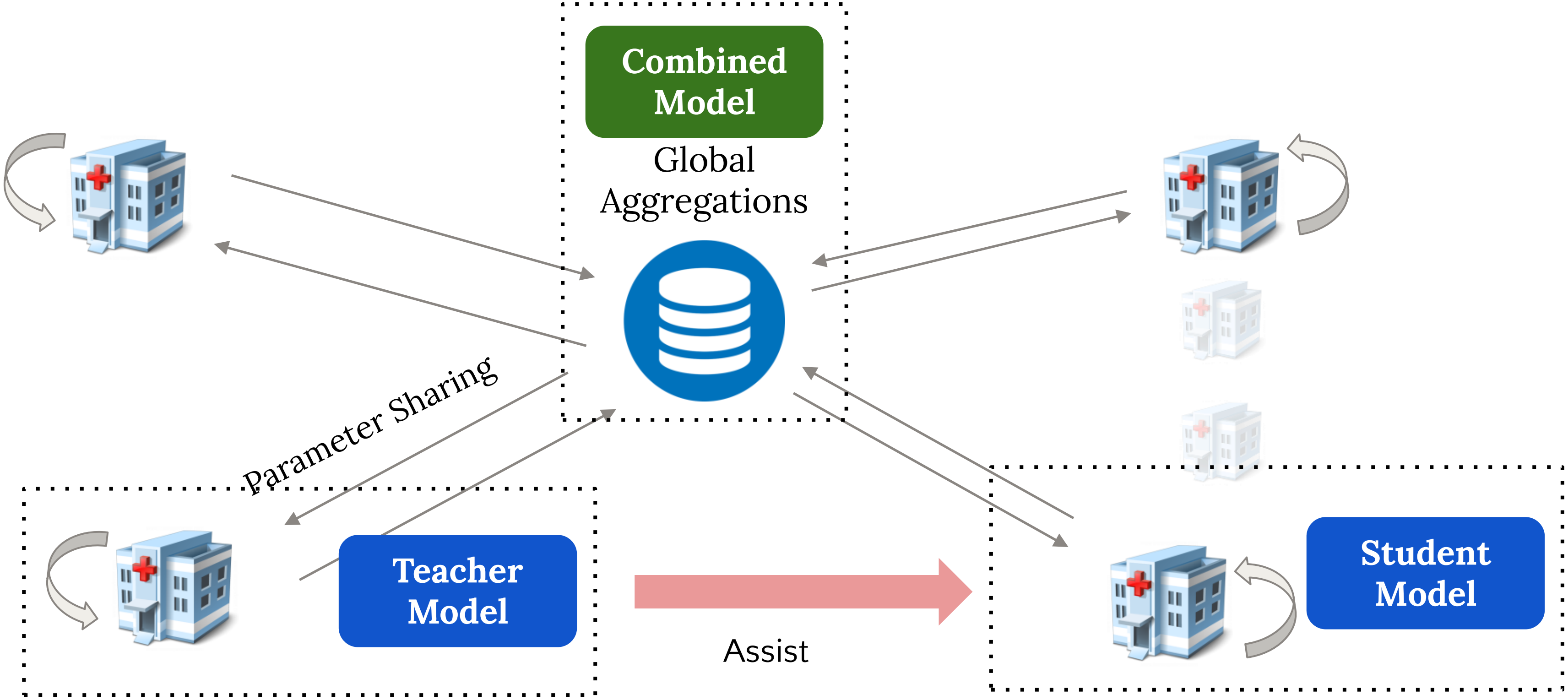


(a) Naive FSSL solution

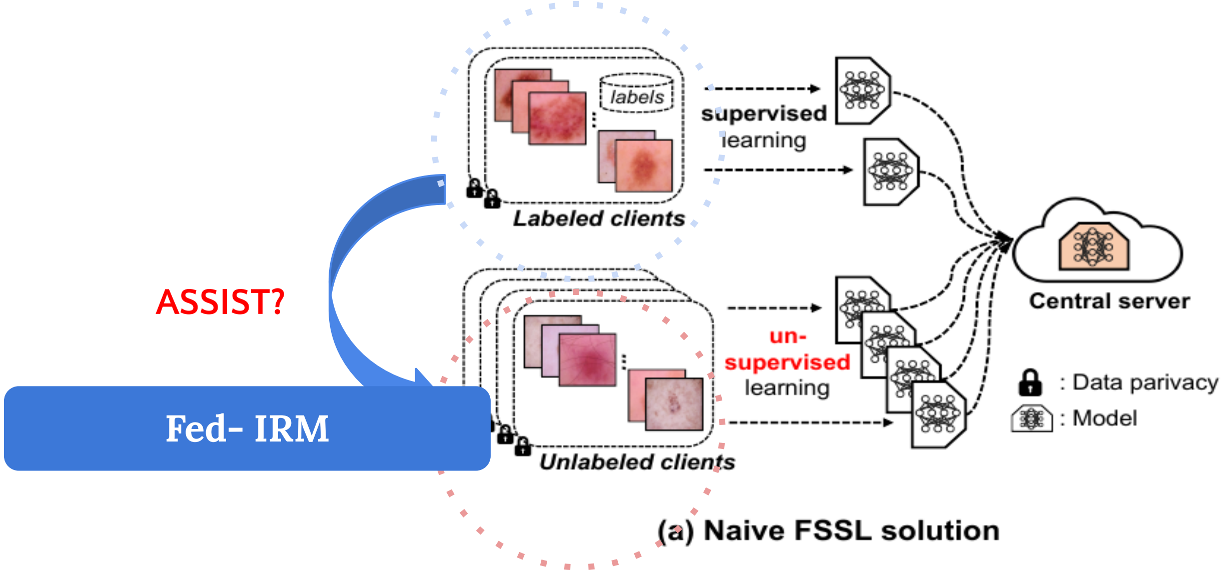
Federated Learning in a Semi Supervised Scenario



Federated Learning in a Semi Supervised Scenario



How to build the **interaction** between the learning at labeled and unlabeled clients, given the challenging constraint of data decentralization?



(a) Naive FSSL solution

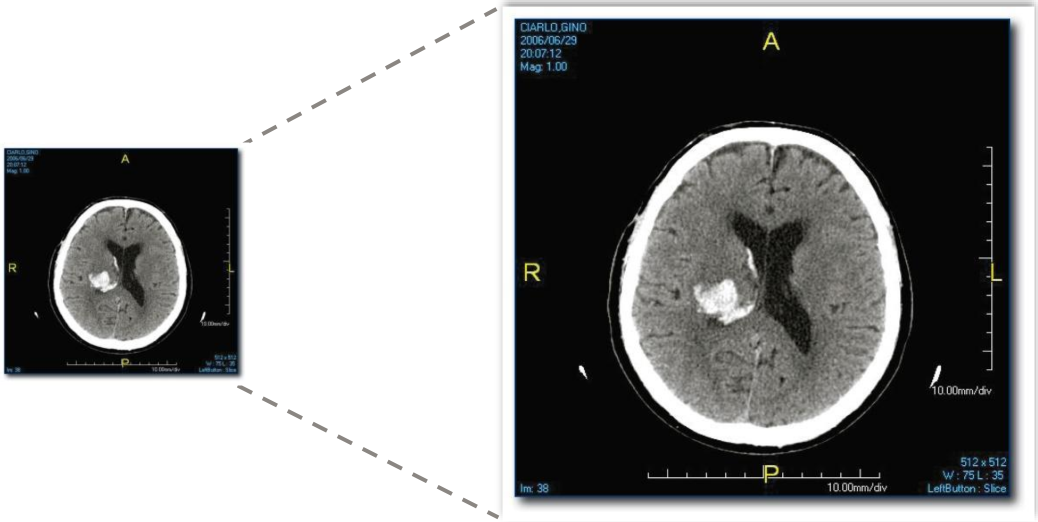
OVERVIEW

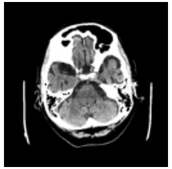
Problem & Motivation

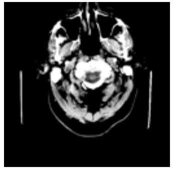
Traditional Approach

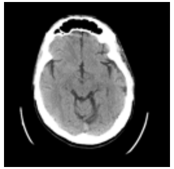
Fed- IRM

Implementation Details

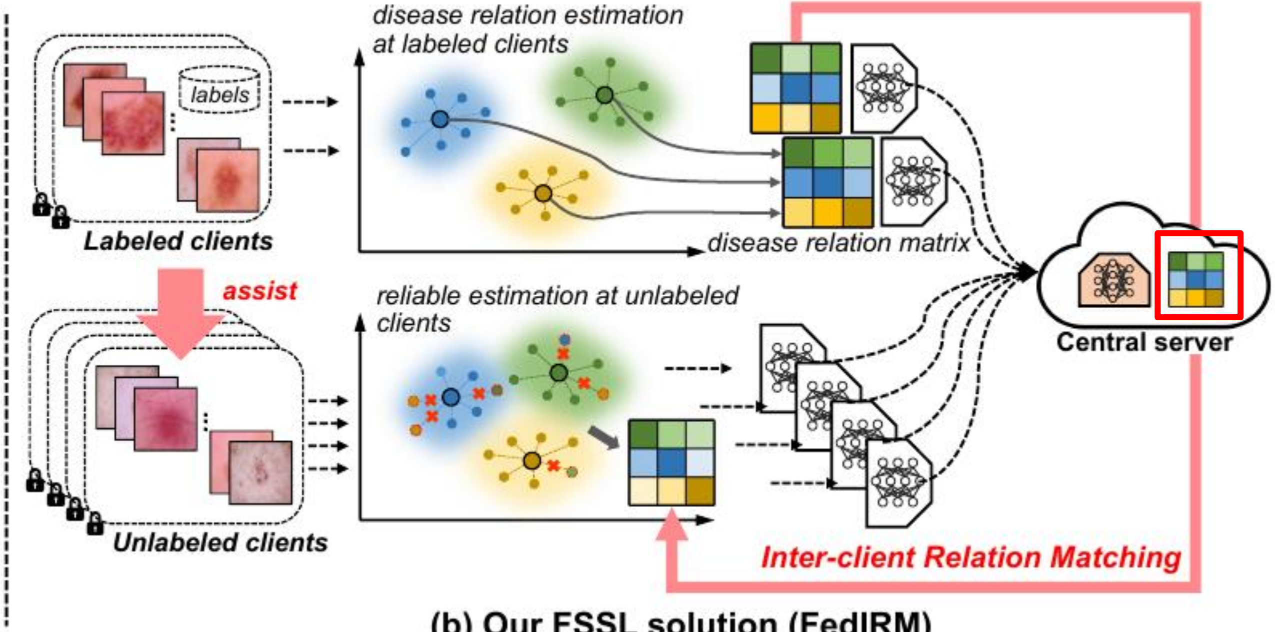






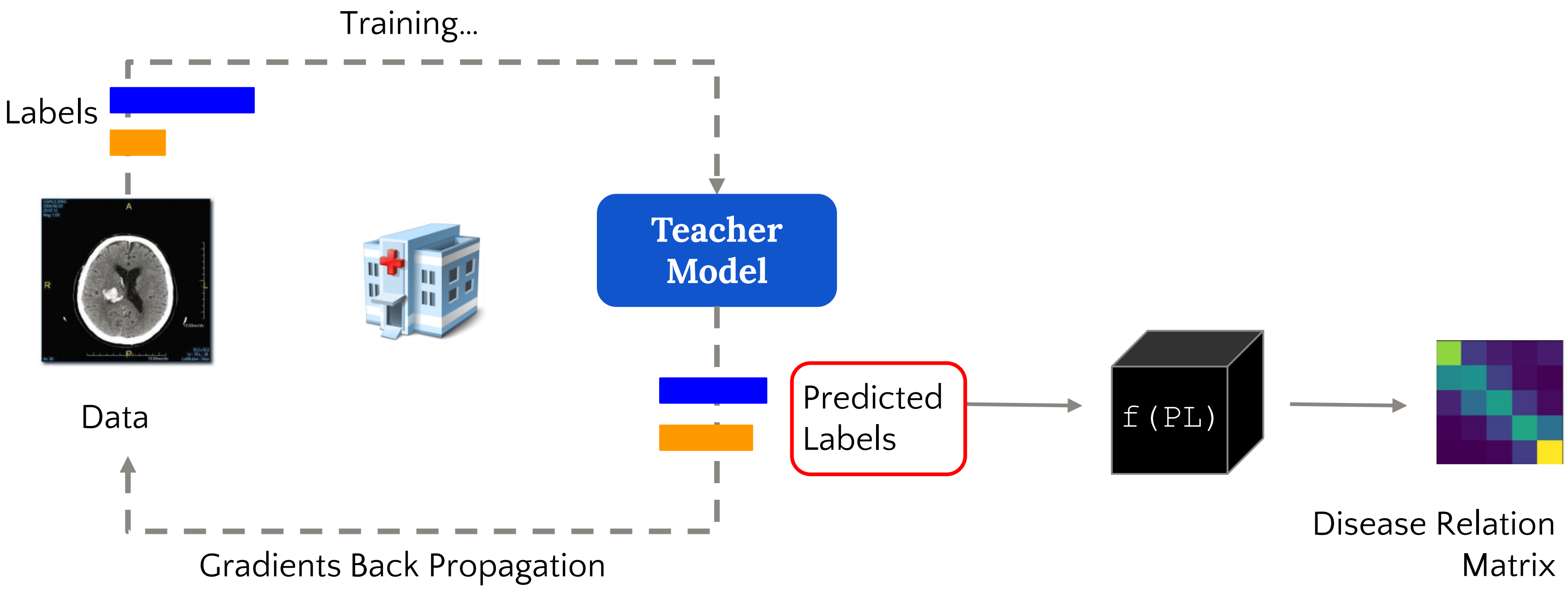


Inter-client Relation Matching scheme regularizes the unlabeled clients to capture **similar disease relationships as labeled clients** for preserving the discriminative task knowledge.



(b) Our FSSL solution (FedIRM)

Disease Relation Matrix in Teacher Model



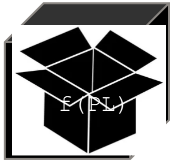
Disease Relation Matrix in Teacher Model

Teacher Model

Pre-Softmax Layer Outputs
= Logits

$\hat{f}_{\theta^l}(x_i^l)$

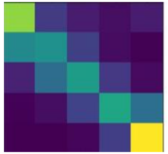
Predicted
Labels



$$\mathbf{v}_c^l = \frac{1}{N_c^l}\sum_{i=1}^{N^l} \mathbb{1}_{[y_i^l=c]}\,\hat{f}_{\theta^l}(x_i^l)$$

$\mathbf{v}_c^l$

Per-Class Mean Feature
Vectors

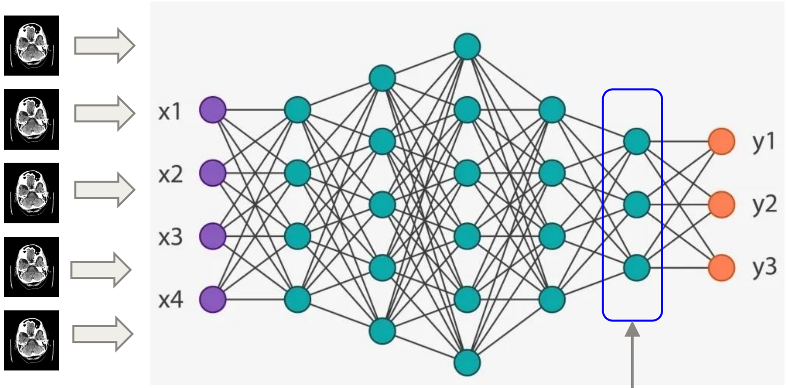


Disease Relation
Matrix

Disease Relation Matrix in Teacher Model

$$\mathbf{v}_c^l = \frac{1}{N_c^l}\sum_{i=1}^{N^l} \mathbb{1}_{[y_i^l=c]} \boxed{\hat{f}_{\theta^l}(x_i^l)}$$

Classification Architecture

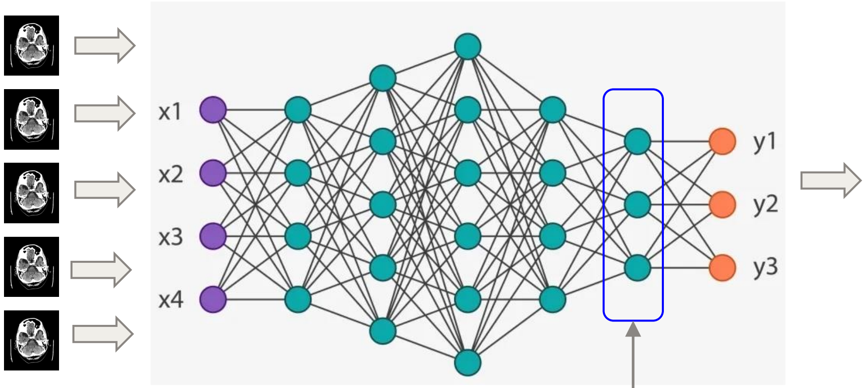


C1	C2	C3
0.19	0.14	**0.65**
0.8	0.6	0.3
0.35	0.1	**0.4**
0.50	0.2	0.46
0.18	0.79	**0.8**

Pre-Softmax Layer Outputs = Logits

Disease Relation Matrix in Teacher Model

$$\boxed{\mathbf{v}_c^l} = \boxed{\frac{1}{N_c^l}\sum_{i=1}^{N^l}} \boxed{\mathbb{1}_{[y_i^l = c]}} \boxed{\hat{f}_{\theta^l}(x_i^l)}$$



Pre-Softmax Layer Outputs = Logits

C1	C2	C3
0.1 9	0.14	**0.65**
0.8	0.6	0.3
0.35	0.1	**0.4**
0.50	0.2	0.46
0.18	0.79	**0.8**

V_{C3}

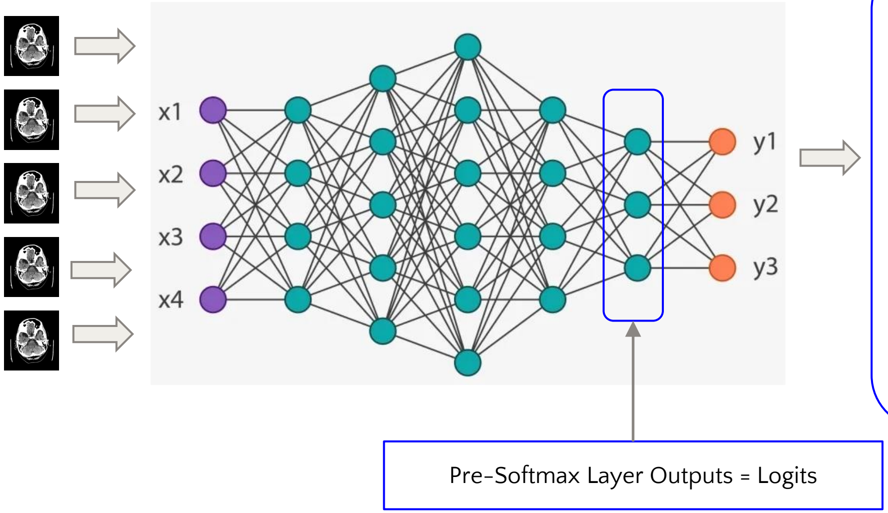
0.19	0.14	**0.65**
0.35	0.1	**0.4**
0.18	0.79	**0.8**

$\boxed{V_{C3}}$

0.11	0.19	**0.39**

Disease Relation Matrix in Teacher Model

$$\boxed{\mathbf{v}_c^l} = \boxed{\frac{1}{N_c^l}\sum_{i=1}^{N^l}} \boxed{\mathbb{1}_{[y_i^l=c]}} \boxed{\hat{f}_{\theta^l}(x_i^l)}$$



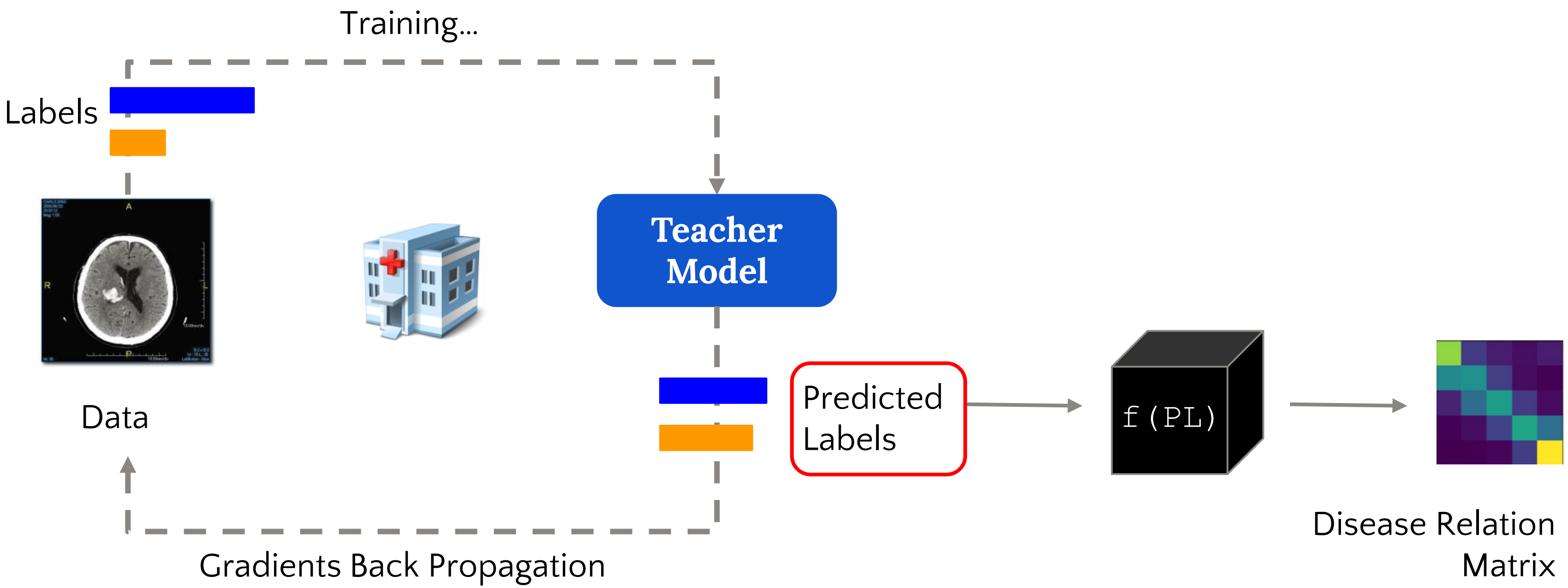
C1	C2	C3
0.1	0.15	**0.75**
0.8	0.1	0.1
0.3	0.3	**0.4**
0.5	0.25	0.25
0.15	0.5	**0.8**

V_{C1}	**0.60**	0.20	0.13
V_{C2}	0.11	**0.79**	0.12
V_{C3}	0.11	0.19	**0.39**

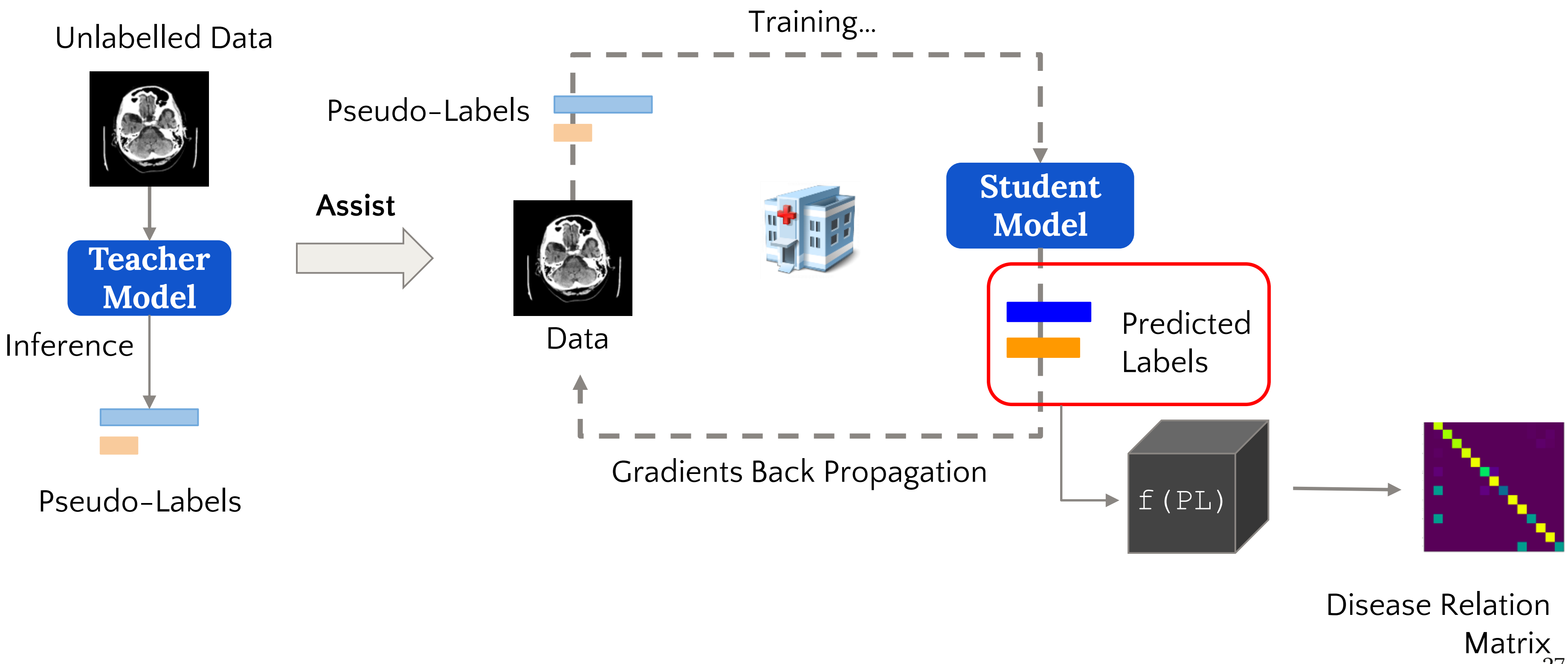
=

Soft Confusion Matrix = Disease Relation Matrix

Disease Relation Matrix in Teacher Model



Disease Relation Matrix in **Student Model**



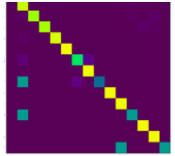
Disease Relation Matrix in **Student Model**

Student Model

Predicted Labels

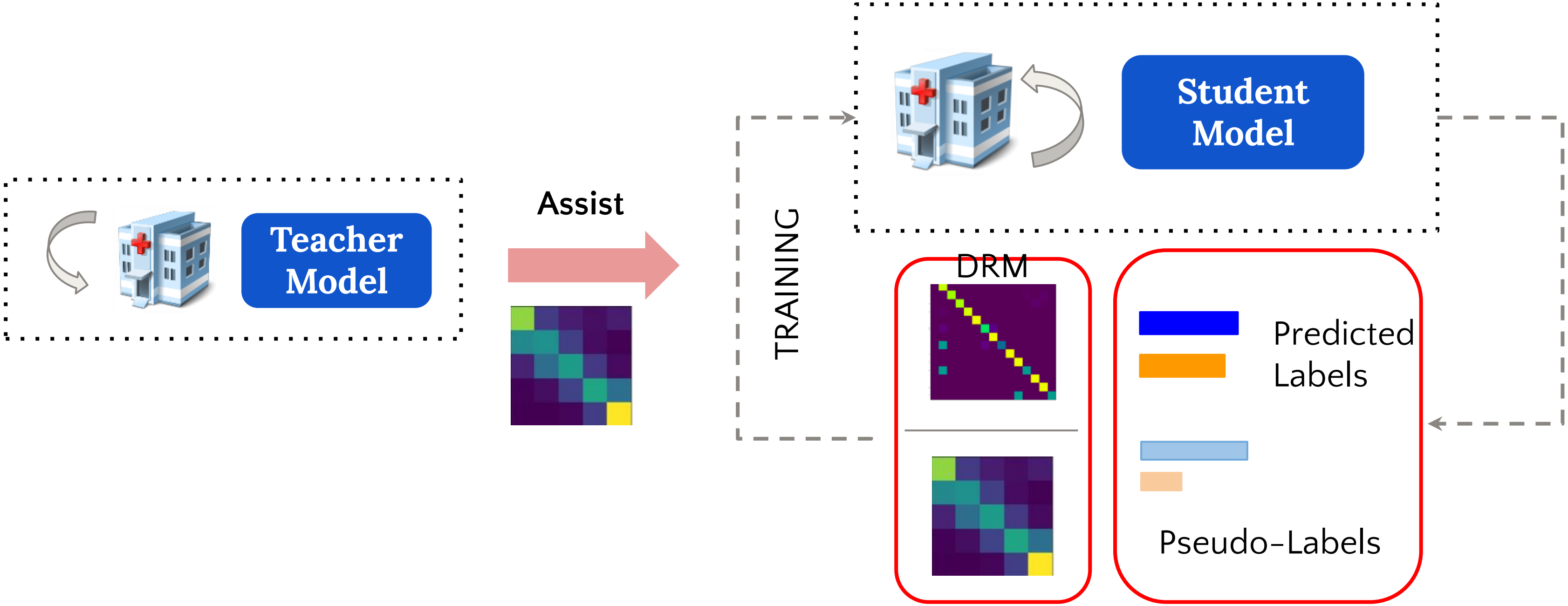
f(PL)

$$\mathbf{v}_c^u = \frac{\sum_{i=1}^{B} \mathbb{1}_{[(\mathbf{y}_i=c)\cdot(\mathbf{w}_i^u<h)]} \cdot \mathbf{p}_i^u}{\sum_{i=1}^{B} \mathbb{1}_{[(\mathbf{y}_i=c)\cdot(\mathbf{w}_i^u<h)]}}$$



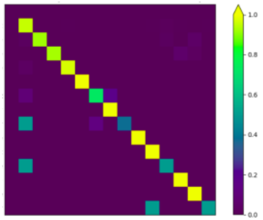
Disease Relation Matrix

Calculation of **Training Loss** in Fed-IRM

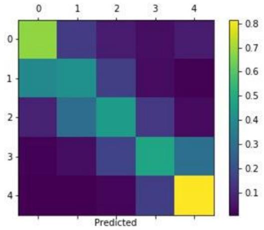


Overall Training Loss = KL Divergence Loss + Cross Entropy Loss

Finally, the inter-client relation matching loss is designed by **minimizing the KL divergence** between disease relation matrix from labelled and unlabelled clients.



DRM, labelled clients



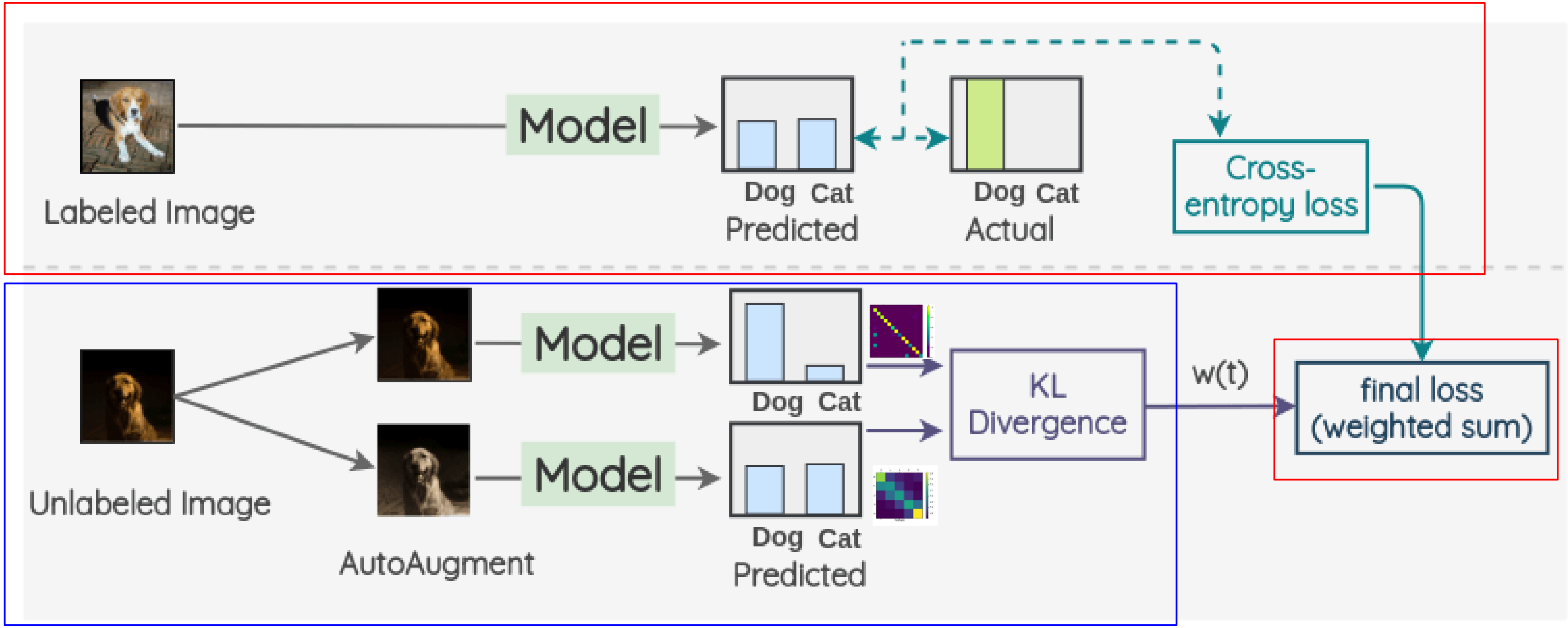
DRM, Un-labelled clients

$$\mathcal{L}_{\mathrm{IRM}} = \frac{1}{C}\sum_{c=1}^{C}(\mathcal{L}_{\mathrm{KL}}(\mathcal{M}_c||\mathcal{M}_c^u) + \mathcal{L}_{\mathrm{KL}}(\mathcal{M}_c^u||\mathcal{M}_c)),$$

$$\text{with } \mathcal{L}_{\mathrm{KL}}(\mathcal{M}_c||\mathcal{M}_c^u)) = \sum_j \mathcal{M}_{c(j)} \log \frac{\mathcal{M}_{c(j)}}{\mathcal{M}_{c(j)}^u}$$

The **local learning objectives** at labeled and unlabeled clients are respectively expressed as:

$$\boxed{\mathcal{L}^l = \mathcal{L}_{ce}(\mathcal{D}^l, \theta^l)} \text{ and } \boxed{\mathcal{L}^u = \lambda(\omega)(\mathcal{L}_c + \boxed{\mathcal{L}_{\mathrm{IRM}})}}$$



Reflection!

Any questions till here?

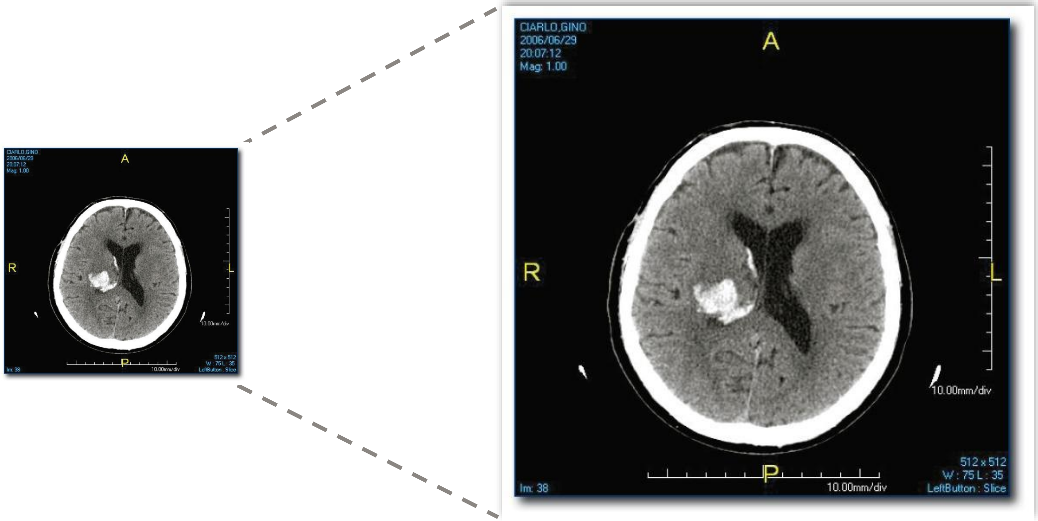
OVERVIEW

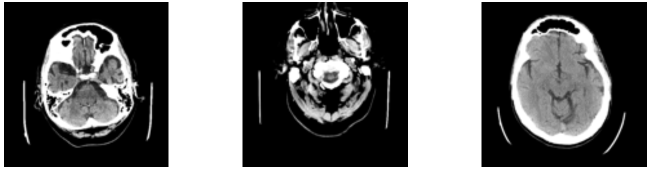
Problem & Motivation

Traditional Approach

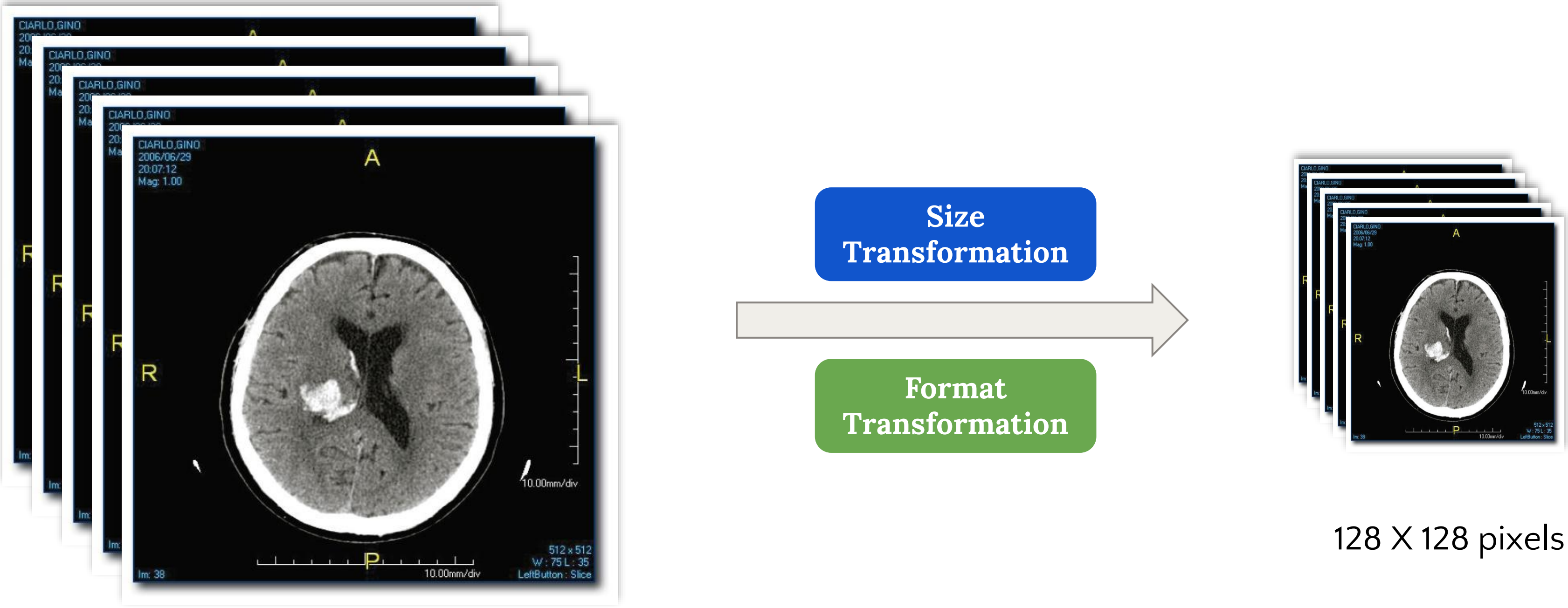
Fed-IRM

Implementations





Dataset

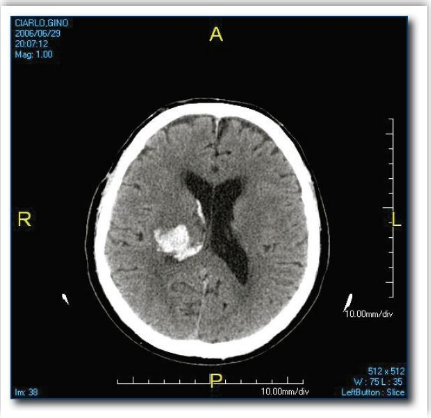


128 X 128 pixels

674, 626 Brain MRI images	185 GB (0.25 MB per image)	DICOM format (Int. Standard)

674, 626 Brain MRI images	3 GB	PNG format

Types of Intracranial Hemorrhage



674, 626
Brain MRI images

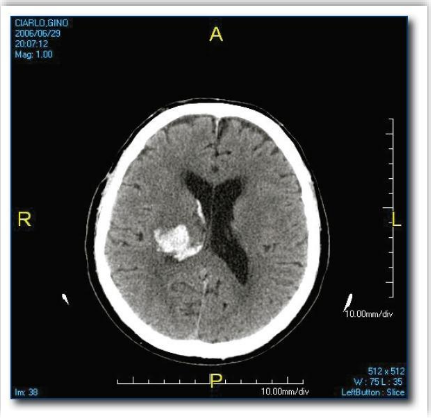
Classification Setup

1. Epidural
2. Parenchymal
3. Intraventricular
4. Subarachnoid
5. Subdural

Intraparenchymal	Intraventricular
Inside of the brain	Inside of the ventricle
	

Subarachnoid	Subdural	Epidural
Between the arachnoid and the pia mater	Between the Dura and the arachnoid	Between the dura and the skull
		

Classification Problem Setup



674, 626
Brain MRI images

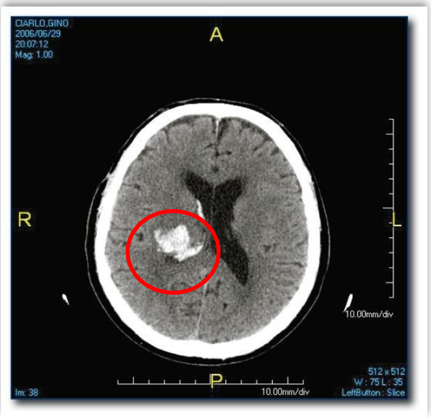
Classification Setup

1. Epidural
2. Parenchymal
3. Intraventricular
4. Subarachnoid
5. Subdural

```
ID_4419f8ae9_epidural,0
ID_4419f8ae9_intraparenchymal,0
ID_4419f8ae9_intraventricular,0
ID_4419f8ae9_subarachnoid,0
ID_4419f8ae9_subdural,0
ID_bcf1add28_epidural,0
ID_bcf1add28_intraparenchymal,0
ID_bcf1add28_intraventricular,1
ID_bcf1add28_subarachnoid,0
ID_bcf1add28_subdural,0
ID_aeda0804d_epidural,0
ID_aeda0804d_intraparenchymal,0
ID_aeda0804d_intraventricular,0
ID_aeda0804d_subarachnoid,0
ID_aeda0804d_subdural,1
```

Labels: Comma Separated Values

Classification Problem Setup



674, 626
Brain MRI images

Classification Setup

1	Epidural	0
2	Parenchymal	0
3	Intraventricular	1
4	Subarachnoid	0
5	Subdural	0

674, 626 X 5
=
3, 373, 130
Data instances

Working Directory, List of all modules

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```

Working Directory

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```

```python
def args_parser():
    parser = argparse.ArgumentParser()
    parser.add_argument("--batch_size", type=int, default=48, help="batch_size per gpu")
    parser.add_argument("--drop_rate", type=int, default=0.2, help="dropout rate")
    parser.add_argument("--ema_consistency", type=int, default=1, help="whether train baseline model")
    parser.add_argument("--base_lr", type=float, default=1e-3, help="maximum epoch number to train")
    parser.add_argument("--deterministic", type=int, default=1, help="whether use deterministic training")
    parser.add_argument("--seed", type=int, default=1337, help="random seed")
    parser.add_argument("--gpu", type=str, default="0,1", help="GPU to use")
    parser.add_argument("--local_ep", type=int, default=2, help="local epoch")
    parser.add_argument("--num_users", type=int, default=10, help="local epoch")
    parser.add_argument("--rounds", type=int, default=200, help="local epoch")

    ### tune
    parser.add_argument("--resume", type=str, default=None, help="model to resume")
    parser.add_argument("--start_epoch", type=int, default=0, help="start_epoch")
    parser.add_argument("--global_step", type=int, default=0, help="global_step")
    ### costs
    parser.add_argument("--label_uncertainty", type=str, default="U-Ones", help="label type")
    parser.add_argument("--ema_decay", type=float, default=0.99, help="ema_decay")
    parser.add_argument("--consistency", type=float, default=1, help="consistency")
    parser.add_argument("--consistency_rampup", type=float, default=30, help="consistency_rampup")
    args = parser.parse_args()
    return args
```

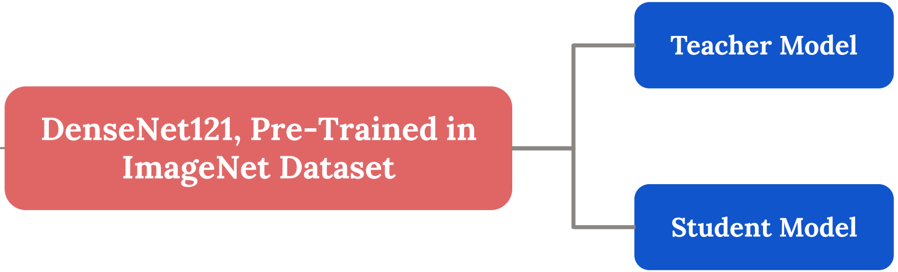
Working Directory

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```

Custom Pytorch Data Loaders

Working Directory

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```



Working Directory

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```

Custom Loading Saved Checkpoints for DenseNet121

Working Directory

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```

Utility Functions to calculate loss and evaluation metrics

Working Directory

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```

Modules to train in Supervised and Semi-Supervised Clients

Working Directory

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```

Global Parameters Aggregations

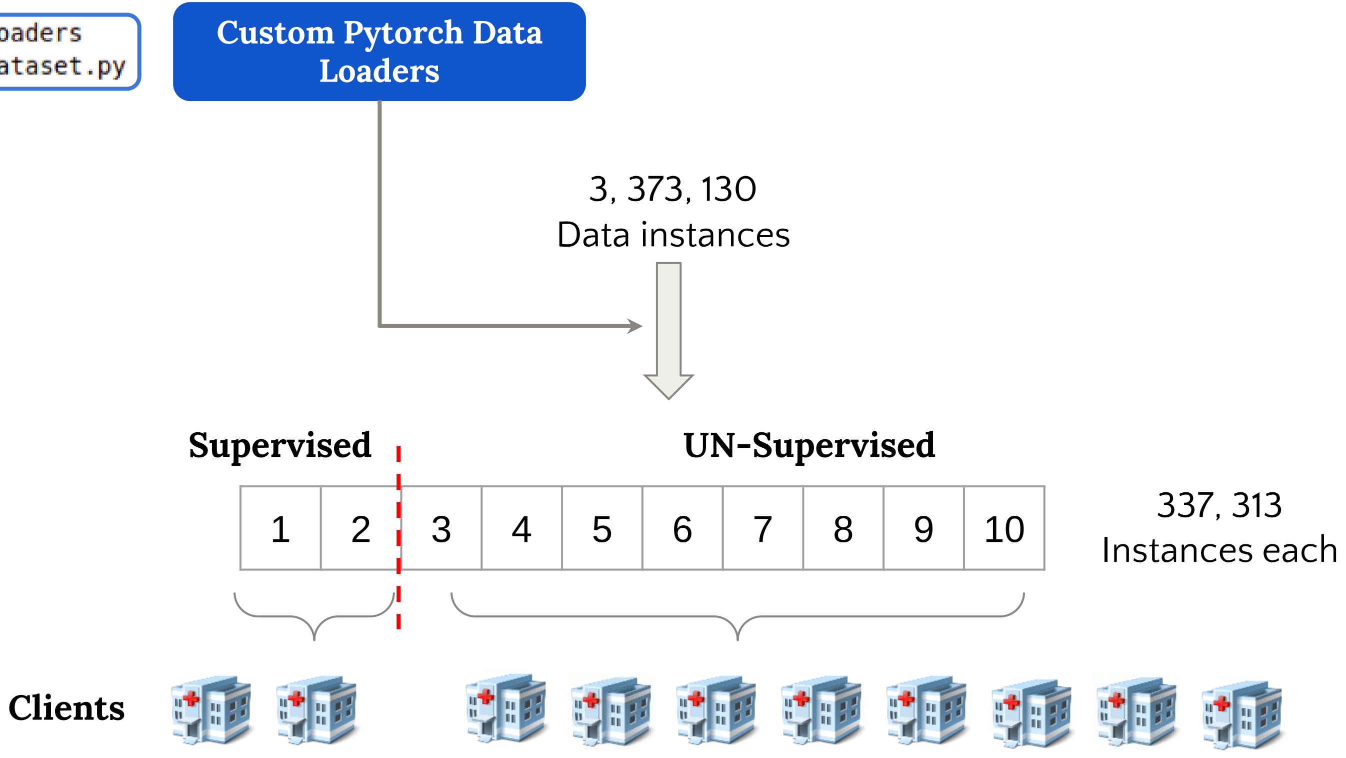
Working Directory

```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
├── models
│   └── hub
│       └── checkpoints
│           └── densenet121.pth
│
├── networks
│   ├── densenet.py
│   └── models.py
│
├── confuse_matrix.py
│
├── utils
│   ├── losses.py
│   └── metrics.py
│
├── local_supervised.py
├── local_unsupervised.py
│
├── FedAvg.py
│
├── train_main.py
└── test.py
```

Code Entrypoint: Trainer and Tester Modules

Simulating Semi Supervised Learning

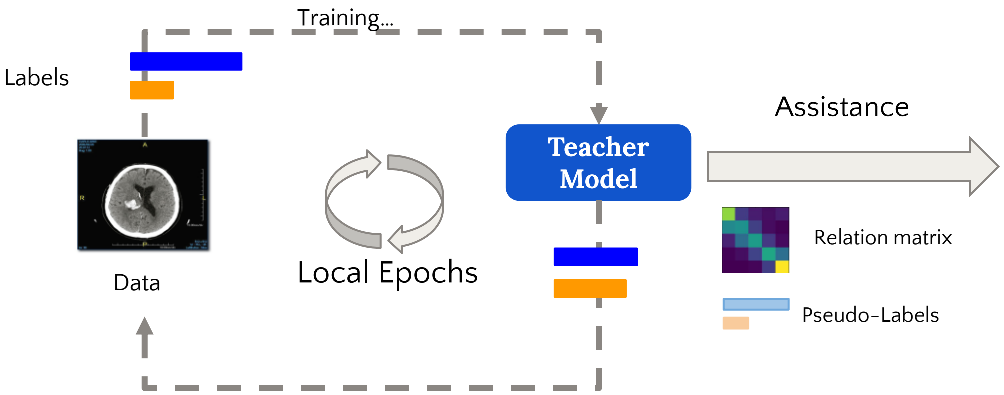
```
├── options.py
│
├── dataloaders
│   └── dataset.py
│
```



Training Supervised Clients

Clients

1
2
3
4
5
6
7
8
9
10



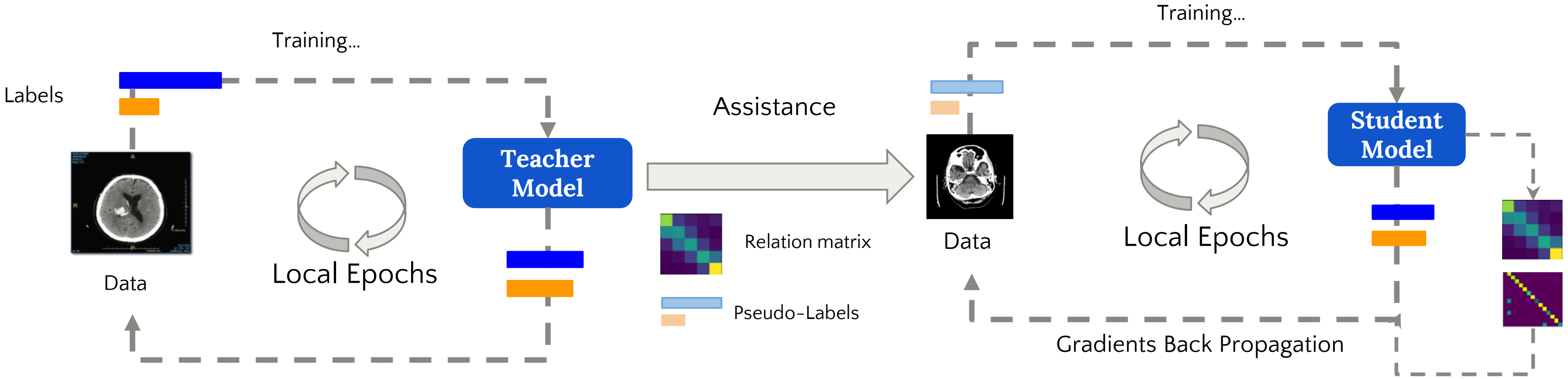
Supervised Training

```python
for epoch in range(args.local_ep):
    batch_loss = []

    iter_max = len(self.ldr_train)

    for i, (_, _, (image_batch, ema_image_batch), label_batch) in enumerate(self.ldr_train):

        image_batch, ema_image_batch, label_batch = (
            image_batch.cuda(),
            ema_image_batch.cuda(),
            label_batch.cuda(),
        )

        inputs = image_batch

        _, outputs = net(inputs)

        with torch.no_grad():
            self.confuse_matrix = self.confuse_matrix + get_confuse_matrix(outputs, label_batch)

        loss_classification = loss_fn(outputs, label_batch.long())

        loss = loss_classification

        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
        batch_loss.append(loss.item())
        self.iter_num = self.iter_num + 1

    self.epoch = self.epoch + 1
    epoch_loss.append(np.array(batch_loss).mean())
```

Training Semi-Supervised Clients

Clients

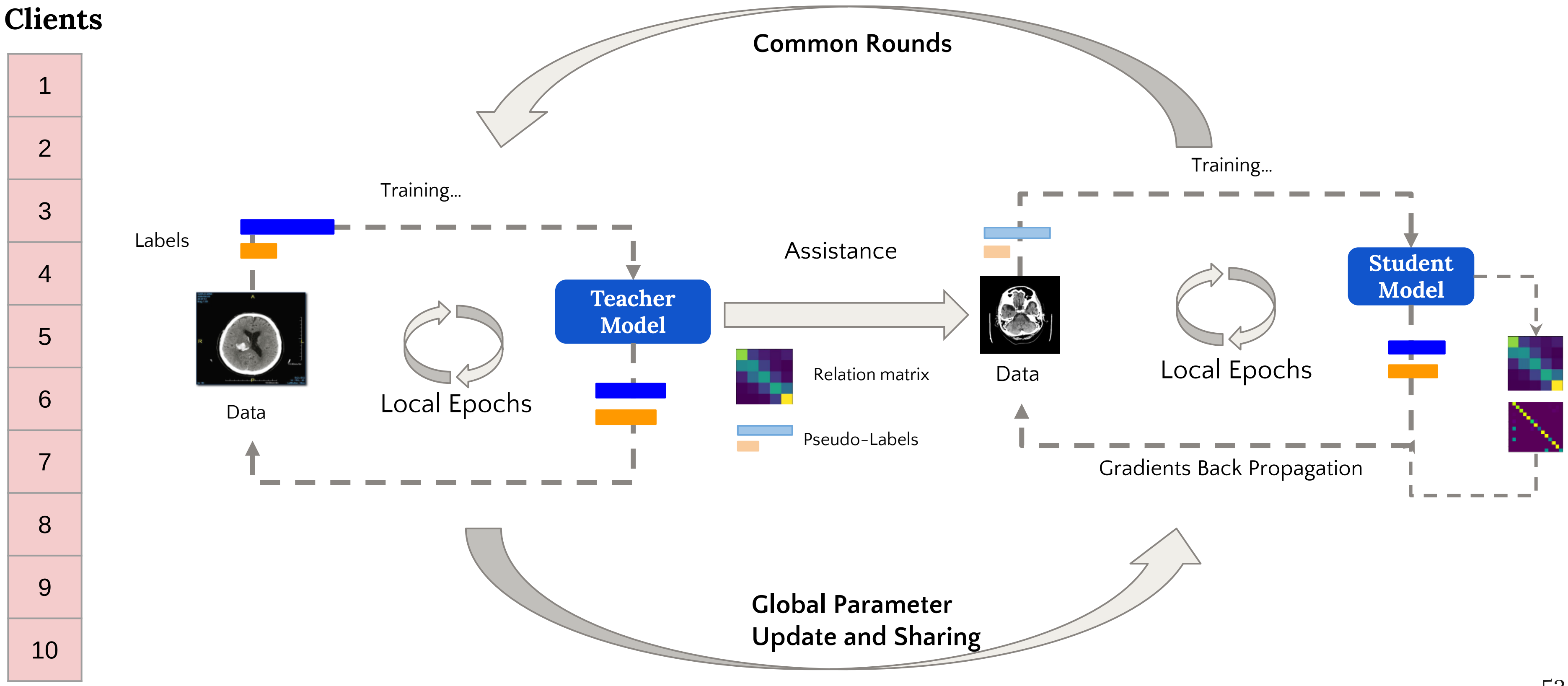
1
2
3
4
5
6
7
8
9
10



Semi-Supervised Training Sample Code

```
for i, (_, _, (image_batch, ema_image_batch), label_batch) in enumerate(self.ldr_train):

    image_batch, ema_image_batch, label_batch = image_batch.cuda(), ema_image_batch.cuda(), label_batch.cuda()
    ema_inputs = ema_image_batch
    inputs = image_batch
    _, outputs = net(inputs)

    T = 10
    with torch.no_grad():
        _, logits_sum = net(inputs)

        for i in range(T):
            _, logits = net(inputs)
            logits_sum = logits_sum + logits

        logits = logits_sum / (T + 1)
        preds = F.softmax(logits, dim=1)

    with torch.no_grad():
        activations = F.softmax(outputs, dim=1)
        confidence, _ = torch.max(activations, dim=1)

    pseudo_labels = F.one_hot(pseudo_labels, num_classes=5)
    source_matrix = get_confuse_matrix(outputs[label_batch], pseudo_labels)
    consistency_weight = get_current_consistency_weight(self.epoch)
    consistency_dist = torch.sum(losses.softmax_mse_loss(outputs, preds)) / args.batch_size
    consistency_loss = consistency_dist

    loss = 15 * consistency_weight * consistency_loss + 15 * consistency_weight * torch.sum(kd_loss(source_matrix, target_matrix))

    self.optimizer.zero_grad()
    loss.backward()
    self.optimizer.step()
    update_ema_variables(net, self.ema_model, args.ema_decay, self.iter_num)
    batch_loss.append(loss.item())

    self.iter_num = self.iter_num + 1
```

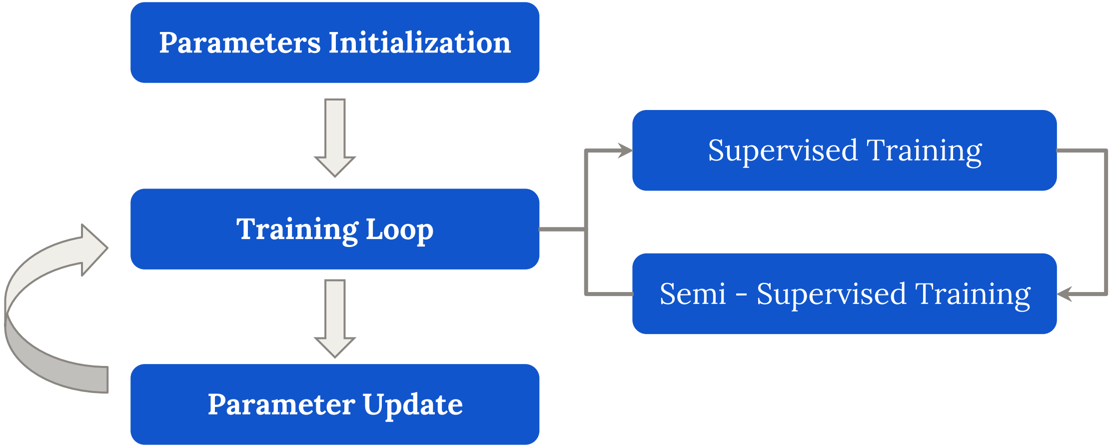
Parameters Update



Parameter Initialization

```
dict_users = split(train_dataset, args.num_users)

net_glob = DenseNet121(out_size=5, mode=args.label_uncertainty, drop_rate=args.drop_rate)

net_glob.train()

w_glob = net_glob.state_dict()
w_locals = []
trainer_locals = []
net_locals = []
optim_locals = []

for i in supervised_user_id:
    trainer_locals.append(SupervisedLocalUpdate(args, train_dataset, dict_users[i]))

    w_locals.append(copy.deepcopy(w_glob))

    net_locals.append(copy.deepcopy(net_glob).cuda())

    optimizer = torch.optim.Adam(
        net_locals[i].parameters(),
        lr=args.base_lr,
        betas=(0.9, 0.999),
        weight_decay=5e-4,
    )
    optim_locals.append(copy.deepcopy(optimizer.state_dict()))

for i in unsupervised_user_id:
    trainer_locals.append(UnsupervisedLocalUpdate(args, train_dataset, dict_users[i]))
```

Parameter Update




```python
with torch.no_grad():
    w_glob = FedAvg(w_locals)

net_glob.load_state_dict(w_glob)

for i in supervised_user_id:
    net_locals[i].load_state_dict(w_glob)

if com_round * args.local_ep > 20:
    for i in unsupervised_user_id:
        net_locals[i].load_state_dict(w_glob)
```

Revised Hyperparameters

Hyperparameter	Original Value	Revised Value
Training Instances	80% = 2, 698, 504	2% = 6, 500
Test Instances	20% = 674, 626	1000
Batch Size	48	5
Local Epochs	5	1
Common Rounds	200	100
GPU	3 GPUs of Titan XP	100% utilization of Tesla P100-SXM2-16GB
Overall Training Time	24 hours + (Parallel Processing)	7 hours 23 minutes (Single GPU)

Results

Hyperparameter	Original Value	Revised Value
Training Instances	80% = 2, 698, 504	2% = 6, 500
Test Instances	20% = 674, 626	1000
Batch Size	48	5
Local Epochs	1	1
Common Rounds	100	100
GPU	3 GPUs of Titan XP	Tesla P100-SXM2-16GB
Overall Training Time	24 hours + (Parallel Processing)	7 hours 23 minutes (Single GPU)
Evaluation Metrics	Accuracy = **92.89 ± 0.25** F1 Score = **55.81 ± 1.49** AUROC = **92.46 ± 0.45**	Accuracy **= 0.663200** F1 Score **= 0.607859** AUROC **= 0.580663**

Results

```
Begin: com_round = 94

[14:04:15.974] Supervised Client: 0
[14:04:15.974] --
[14:04:31.551] Supervised Client: 1
[14:04:31.551] --
[14:04:47.034] Semi-Supervised Client: 2
[14:04:47.034] --
[14:05:18.369] Semi-Supervised Client: 3
[14:05:18.369] --
[14:05:50.078] Semi-Supervised Client: 4
[14:05:50.078] --
[14:06:21.786] Semi-Supervised Client: 5
[14:06:21.786] --
[14:06:52.452] Semi-Supervised Client: 6
[14:06:52.452] --
[14:07:23.026] Semi-Supervised Client: 7
[14:07:23.026] --
[14:07:53.785] Semi-Supervised Client: 8
[14:07:53.786] --
[14:08:24.390] Semi-Supervised Client: 9
[14:08:24.390] --
[14:08:55.883] Loss Avg: 0.158969905116299, Common Round: 94, LR: 0.001
[14:09:00.894] TEST AUROC: 0.561466, TEST Accus: 0.630000, F1: 0.571264
[14:09:00.895]
Begin: com_round = 95

[14:09:00.895] Supervised Client: 0
```

```
Begin: com_round = 97

[14:33:08.663] Semi-Supervised Client: 2
[14:33:08.663] --
[14:33:40.522] Semi-Supervised Client: 3
[14:33:40.522] --
[14:34:11.544] Semi-Supervised Client: 4
[14:34:11.544] --
[14:34:42.342] Semi-Supervised Client: 5
[14:34:42.342] --
[14:35:13.087] Semi-Supervised Client: 6
[14:35:13.087] --
[14:35:43.846] Semi-Supervised Client: 7
[14:35:43.846] --
[14:36:14.373] Semi-Supervised Client: 8
[14:36:14.373] --
[14:36:45.141] Semi-Supervised Client: 9
[14:36:45.141] --
[14:37:18.142] Loss Avg: 0.0032918424397146077, Common Round: 100, LR: 0.001
[14:37:23.170] TEST AUROC: 0.580663, TEST Accus: 0.663200, F1: 0.607859[15:35:01.056]
```

Evaluation Metrics	Accuracy **= 0.663200**	F1 Score **= 0.607859**	AUROC **= 0.580663**

Experimentations on Available Model

<table>
<tr><th>Hyperparameters</th><th>Models</th><th>Accuracy</th><th>F1 Score</th><th>AUROC</th><th>Training Time (Single GPU)</th></tr>
<tr><td rowspan="4">Training Instances: 6500
Test Instances: 1000
Batch Size: 5
Local Epochs: 1
Common Rounds: 100
GPU: Tesla P100-SXM2-16GB</td><td>Revised Baseline Model</td><td>0.663200</td><td>0.607859</td><td>0.580663</td><td>7 hours 23 mins</td></tr>
<tr><td>DRM Distribution
Wasserstein Distance(WD) instead of KL Divergence</td><td>0.682241</td><td>0.634432</td><td>0.537658</td><td>7 hours 52 mins</td></tr>
<tr><td>Self Attention Layer on top of DenseNet121</td><td>0.710034</td><td>0.681145</td><td>0.635578</td><td>10 hours 44 mins</td></tr>
<tr><td>Future Scope:
Segmentation instead of Discrimination Task</td><td colspan="4">3D UNet for Segmentation tasks in MSD Dataset in semi supervised setting</td></tr>
</table>

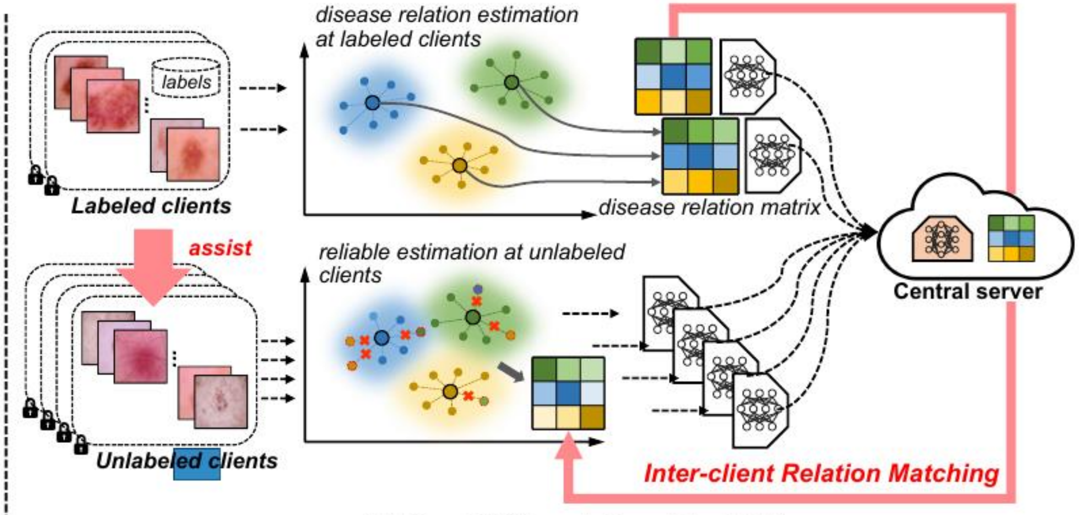
Results: Self Attention with DenseNet-121

```
[16:48:29.380] Semi-Supervised Client: 9
[16:48:29.380] --
[16:49:12.289] Loss Avg: 0.1600619061962269, Common Round: 89, LR: 0.001
[16:49:17.476] TEST AUROC: 0.590694, TEST Accus: 0.703000, F1: 0.652454
[16:49:17.477]
Begin: com_round = 90

[16:49:17.477] Supervised Client: 0
[16:49:17.477] --
[16:49:37.165] Supervised Client: 1
[16:49:37.165] --
[16:49:56.450] Semi-Supervised Client: 2
[16:49:56.450] --
[16:50:36.444] Semi-Supervised Client: 3
[16:50:36.444] --
[16:51:16.744] Semi-Supervised Client: 4
[16:51:16.744] --
[16:51:57.037] Semi-Supervised Client: 5
[16:51:57.037] --
[16:52:37.348] Semi-Supervised Client: 6
[16:52:37.348] --
[16:53:17.049] Semi-Supervised Client: 7
[16:53:17.049] --
[16:53:57.306] Semi-Supervised Client: 8
[16:53:57.307] --
[16:54:38.264] Semi-Supervised Client: 9
[16:54:38.265] --
[16:55:20.522] Loss Avg: 0.16720833418491693, Common Round: 90, LR: 0.001
[16:55:25.708] TEST AUROC: 0.635578, TEST Accus: 0.710034, F1: 0.681145
[16:55:25.709]
Begin: com_round = 91

[16:55:25.709] Supervised Client: 0
```

Evaluation Metrics	Accuracy **= 0.710034**	F1 Score **= 0.681145**	AUROC **= 0.635578**

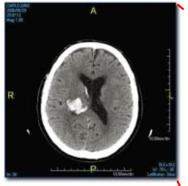
Making the use of unannotated data reduces the cost in individual annotations which can be redirected to more meaningful research.

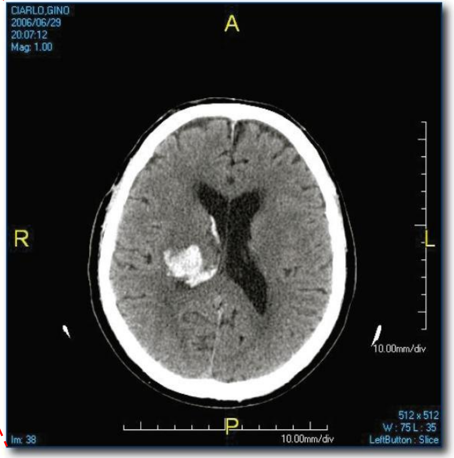
Unlabelled Data

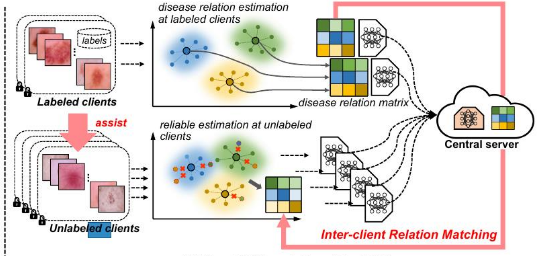


(b) Our FSSL solution (FedIRM)

Intracranial Hemorrhage

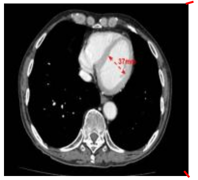


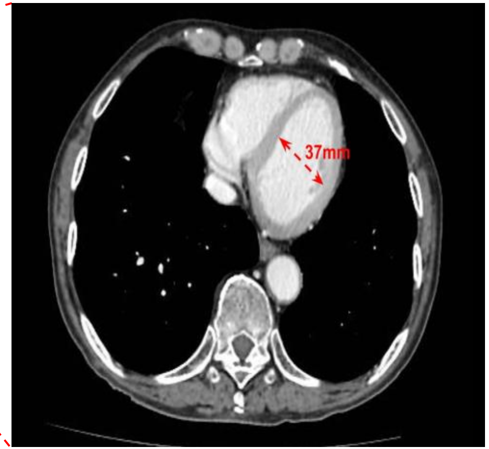


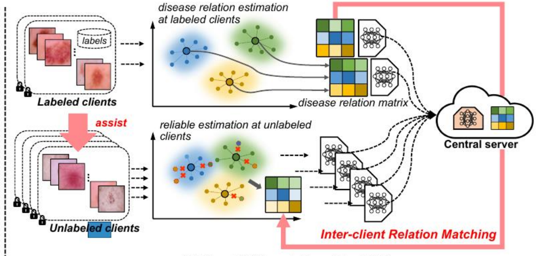


(b) Our FSSL solution (FedIRM)

Myocardial Infarction

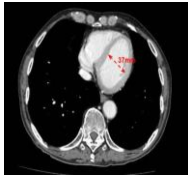


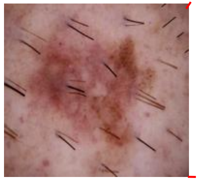


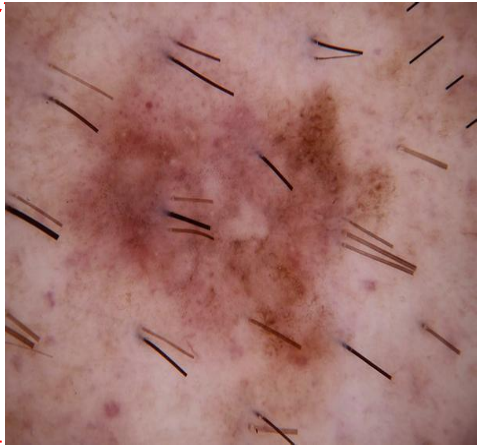


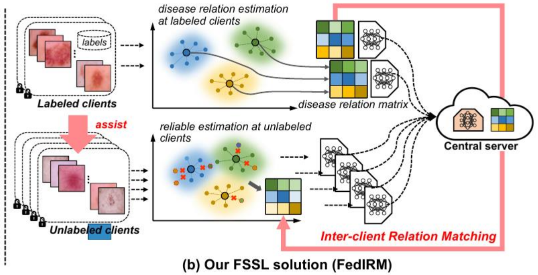
(b) Our FSSL solution (FedIRM)

Skin Lesions

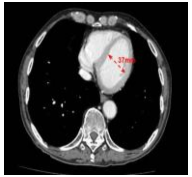


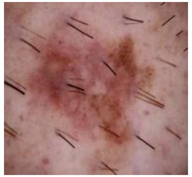


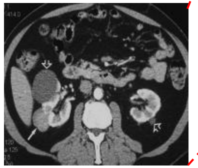


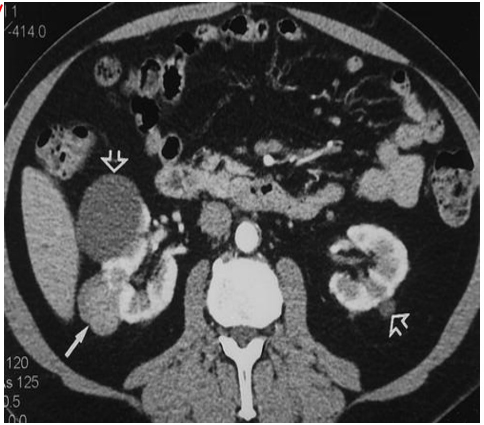


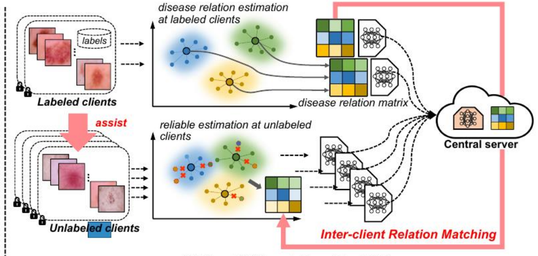
Renal Cell Carcinoma





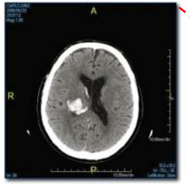
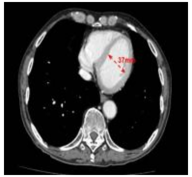
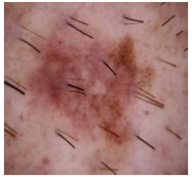
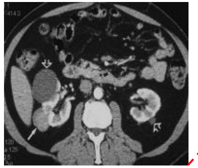


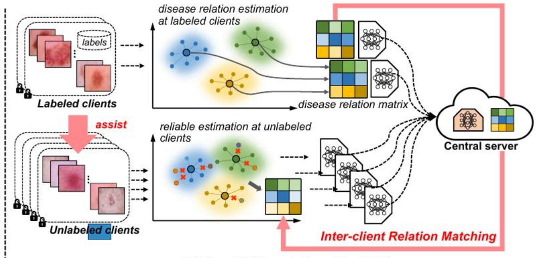




(b) Our FSSL solution (FedIRM)

Huge number of application possibilities



(b) Our FSSL solution (FedIRM)

References

1. Aviles-Rivero, A.I., Papadakis, N., Li, R., Sellars, P., Fan, Q., Tan, R.T., Schönlieb, C.: Graphx net chest x-ray classification under extreme minimal supervision.

2. Bai, W., Oktay, O., Sinclair, M., Suzuki, H., Rajchl, M., Tarroni, G., Glocker, B., King, A., Matthews, P.M., Rueckert, D.: Semi-supervised learning for network- based cardiac mr image segmentation

3. Chang, Q., Qu, H., Zhang, Y., Sabuncu, M., Chen, C., Zhang, T., Metaxas, D.N.: Synthetic learning: Learn from distributed asynchronized discriminator gan without sharing medical image data.

4. Cheplygina, V., de Bruijne, M., Pluim, J.P.: Not-so-supervised: a survey of semi- supervised, multi-instance, and transfer learning in medical image analysis. Medical image analysis.

5. Cui, W., Liu, Y., Li, Y., Guo, M., Li, Y., Li, X., Wang, T., Zeng, X., Ye, C.: Semi-supervised brain lesion segmentation with an adapted mean teacher model.

6. Tschandl, P., Rosendahl, C., Kittler, H.: The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 1–9 (2018)

7. Wang, D., Zhang, Y., Zhang, K., Wang, L.: Focalmix: semi-supervised learning for 3D medical image detection. In: CVPR, pp. 3951–3960 (2020)

8. Zhang, Z., Yao, Z., Yang, Y., Yan, Y., Gonzalez, J.E., Mahoney, M.W.: Benchmarking semi-supervised federated learning. arXiv preprint arXiv:2008.11364 (2020)

Thanks!

Any questions ?

You can find me at

- cs21mtech16001@iith.ac.in