SplitFed
Federated Learning meets Split Learning

PRESENTED BY

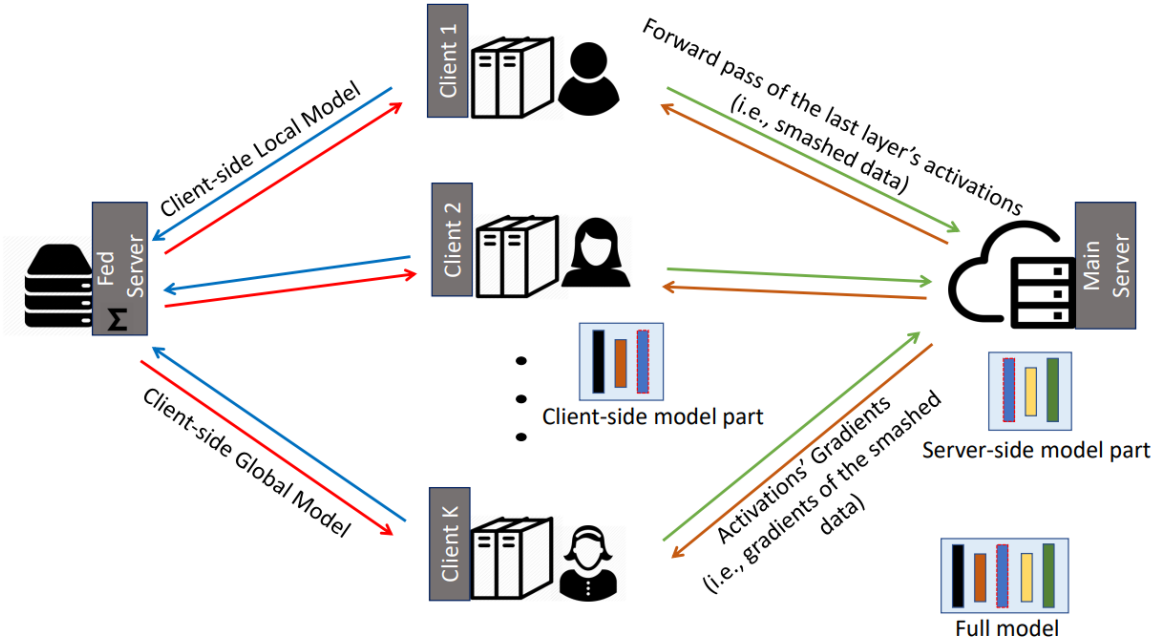
0424052053 - MOBASWIRUL ISLAM

Research Paper

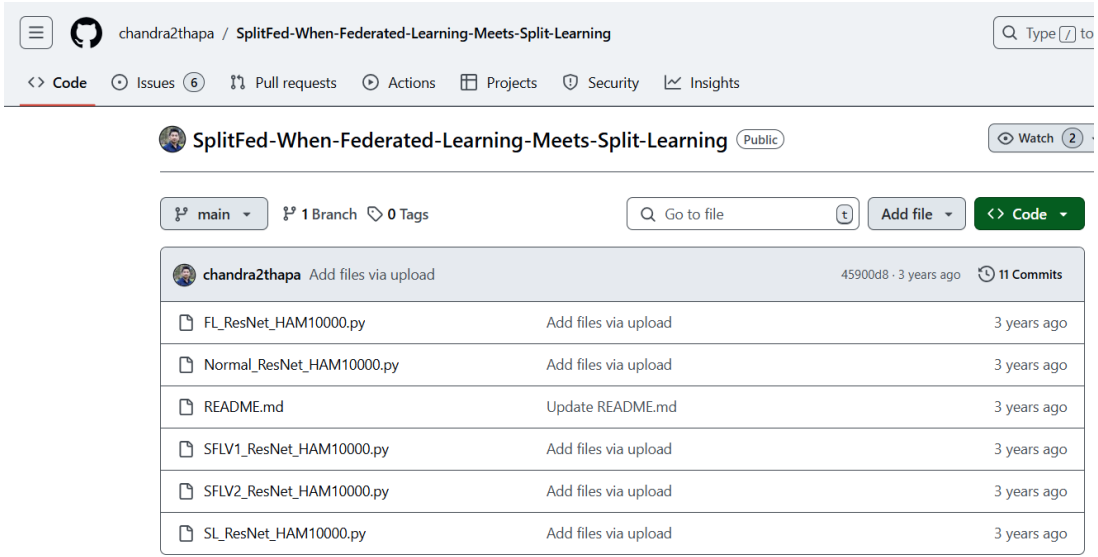
SplitFed: When Federated Learning Meets Split Learning

Chandra Thapa, Pathum Chamikara, Mahawaga Arachchige, Seyit Camtepe, Lichao Sun

Conference: AAAI - `22



SplitFed Recap



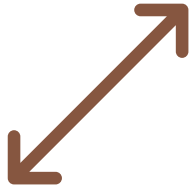
https://github.com/chandra2thapa/SplitFed-When-Federated-Learning-Meets-Split-Learning

Implementation

SIMULATION ONLY

Cloud VM and Storage

1 Server



3 Clients

Google Bucket

```
class MessageType(Enum):...

class Message:...


def send_message_as_json(client_socket, message_type, sender="", receiver = "", content = ""):...

def receive_message_as_json(client_socket):...

def send_file(client_socket, directory_path, file_path, message_type = MessageType.REQUEST_TO_SEND_FILE):...

def receive_file(client_socket, directory_path, file_path, message_type = MessageType.SEND_FILE):...
```

Methodology

CUSTOM SERVER SOCKET SCHEME

```python
class ClientModel(nn.Module):
    def __init__(self):
        super(ClientModel, self).__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(784, 128)
        self.activation1 = nn.ReLU()

    def forward(self, x):...
```

```python
class ServerModel(nn.Module):
    def __init__(self):
        super(ServerModel, self).__init__()
        self.fc1 = nn.Linear(128, 64)
        self.activation1 = nn.ReLU()
        self.fc2 = nn.Linear(64, 10)
        self.activation2 = nn.Softmax(dim=1)

    def forward(self, x):...
```

Split Fed (Separate Client and Server Network)

```
for epoch in range(TOTAL_GLOBAL_EPOCH):
    for i, (images, labels) in enumerate(train_loader):
        images, labels = images.to(device), labels.to(device)
        client_optimizer.zero_grad()
        split_layer_tensor = client_model(images)
        split_layer_tensor = split_layer_tensor.detach().requires_grad_(True)

        grads = comm_with_server(labels, split_layer_tensor, epoch, i)

        split_layer_tensor.backward(grads)
        client_optimizer.step()
```

Split Fed Mechanism – Client Side

```python
def communicate_with_server_during_training(labels, split_layer_tensor, epoch, i):
    global client_socket

    torch.save(labels, f"{client_file_directory_path}/labels_{epoch}_{i}.pt")
    torch.save(split_layer_tensor, f"{client_file_directory_path}/split_layer_tensor_{epoch}_{i}.pt")

    send_file(client_socket, client_file_directory_path, file_path: f"labels_{epoch}_{i}.pt")
    send_file(client_socket, client_file_directory_path, file_path: f"split_layer_tensor_{epoch}_{i}.pt")

    grads_message = receive_message_as_json(client_socket)
    if grads_message.message_type == MessageType.REQUEST_TO_SEND_FILE:
        receive_file(client_socket, server_file_directory_path, file_path: f"grads_{epoch}_{i}.pt")
        grads = torch.load(f"{server_file_directory_path}/grads_{epoch}_{i}.pt")

        return grads

    return None
```

Split Fed Mechanism – Client Side

Split Fed Mechanism Server Side

```
if msg.message_type == MessageType.REQUEST_TO_SEND_FILE:
    labels_path = msg.content
    receive_file(client_socket, specific_client_file_directory_path, labels_pat
    slt_message = receive_message_as_json(client_socket)
    if slt_message.message_type == MessageType.REQUEST_TO_SEND_FILE:
        slt_path = slt_message.content
        receive_file(client_socket, specific_client_file_directory_path, slt_pa
        labels = torch.load(f"{specific_client_file_directory_path}/{labels_pat
        split_layer_tensor = torch.load(f"{specific_client_file_directory_path}

        server_optimizer.zero_grad()
        server_output = server_model(split_layer_tensor)
        loss = loss_criteria(server_output, labels)

        # print to log file
        with open(f"shared_files/server/{server_log_file}", "a") as log_file:
            log_file.write(f"{clients_nos[client_name]},{labels_path},loss,{los

        loss.backward()
        server_optimizer.step()
        split_layer_tensor.retain_grad()
        torch.save(split_layer_tensor.grad, f"{specific_client_file_directory_p

        send_file(client_socket, specific_client_file_directory_path, file_path: f
    else:
        print("Invalid message 1.")
```

Thank You

HTTPS://GITHUB.COM/EZMATA-101/SFL