E2FL: Equal and Equitable Federated Learning

Hamid Mozaffari, Amir Houmansadr

University of Massachusetts Amherst
hamid@cs.umass.edu, amir@cs.umass.edu

Abstract

Federated Learning (FL) enables data owners train a shared global model without sharing their private data. Unfortunately, FL is susceptible to an intrinsic fairness issue: due to the heterogeneity of clients' data distributions, the final FL model can give disproportionate advantages across the participating clients. In this work, we present Equal and Equitable Federated Learning (E2FL) to produce fair federated learning models by preserving two main fairness properties, equity and equality, *concurrently*. We validate the efficiency and fairness of E2FL in different real-world FL applications, and show that E2FL outperforms existing baselines in terms of the resulting efficiency, fairness of different groups, and fairness among all individual clients.

1 Introduction

Federated Learning (FL) is an emerging AI technology where *clients* collaborate to train a shared model, called the *global model*, without explicitly sharing their local training data. FL training involves a *server* which collects model updates from selected FL clients in each round of training, and uses them to update the global model. In FL, the performance of the global model varies across the clients due to heterogeneity in the data that each client owns. This concern is called *representation disparity* (Hashimoto et al. 2018) and results in unfair performance gaps for the participating clients. That is, although the accuracy may be high on average, some tail user whose data distribution differs from the majority of the clients is likely to receive a much lower performance compared to the average.

In this work, we look at FL fairness with two different lenses: **a)** *Equality*: whose goal is providing similar performances for all individual clients; **b)** *Equity*: whose goal is providing similar performances across all groups of clients (i.e., groups of majority and minority), where a group is defined as a set of clients with similar data distributions. The key question we try to answer is: *Can we design an efficient federated learning algorithm that achieves both equality and equity concurrently?*

Due to the heterogeneity in clients' data distributions, one single model cannot represent all the clients equally. There is a *trade-off* between training one global model and multiple global models; if we train one global model all the clients can utilize each other's knowledge, however it will be biased towards whom that have the majority of the population. On the other hand, if we train multiple models (e.g., as in IFCA (Ghosh et al. 2020), HypCluster (Mansour et al. 2020) and MOCHA (Smith et al. 2017)), we improve fairness, but each global model will lose the knowledge from excluded clients. To get the best of both worlds, we present **E**qual and **E**quitable **F**ederated **L**earning (E2FL), a novel FL algorithm to achieve both equality and equity. In E2FL, we train multiple global models, but in each round we combine all of the models into one global model to take advantage of the knowledge of all client groups.

The key insight used in E2FL is converting the problem of model weight optimization (in standard FL) to the problem of ranking model edges (a technique recently proposed in (Mozaffari, Shejwalkar, and Houmansadr 2021)). Therefore, in each round of E2FL training, the clients and the server exchange rankings for the edges of a randomly initialized neural network (called *supernetwork*), as opposed to exchanging parameter gradients. More specifically, each E2FL client computes the importance of the edges within a randomly initialized neural network (called supernetowrk) on their local data, represented by a ranking vector. Next, E2FL server uses a majority voting mechanism to aggregate the collected local rankings into multiple global rankings based on the index of group they belong to. Finally, the E2FL server aggregates all the group rankings into one global ranking for next round of training. Applying the majority vote on the group rankings instead of all the local rankings helps E2FL enforce equity because each group has an equal vote to influence the global model. To provide equality in E2FL, if a client wants to use the model in a downstream task, they use their own group global ranking, instead of the global ranking, which is a better representation model for the client and its groupmates.

Our ranking-based FL training enables attractive fairness properties, as shown through our experiments, which is intuitively due to the following reason: In rank-based federated learning, each client computes a local ranking (i.e., a permutation of integers $\in [1, d]$ where d is the layer size), so each local ranking has a fixed norm (i.e., $\sqrt{1^2 + 2^2 + ... + d^2}$). This fixed norm of local updates makes the rank aggregation more fair as each local ranking has the same impact on the aggregated global ranking. On the other hand, in standard FL, when the server aggregates the local model updates into the global model, each local update has a different impact on the global model (because of their different l_2 norms). For example in FedAvg, the server averages the parameter updates for the d dimensions, therefore a large parameter update has more influence on the final average compared to a small parameter update.

E2FL when the group IDs are unknown. In many applications, clients may be unaware of their protected attributes (i.e., the group they belong to). We propose one approach on server-side and three approaches on client-side for inferring group IDs. To infer the group IDs on the server-side, we propose to use a rank clustering approach to cluster clients into groups. Moreover, a client can also infer its group IDs by picking the right group based on their local training data. Using rankings allows us to exchange only the binary masks produced by each group ranking which lowers the communication cost compared to prior works. Each client can pick the right binary mask based on three approaches. First, each client can pick the binary mask that produces the smallest loss. Binary masks also enable the clients to find their matching group by a new novel idea from (Wortsman et al. 2020), where clients can infer the group ID using gradient based optimization to find a linear superposition of learned masks which minimizes the output entropy. We propose two variants of this approach, one based on a binary search and the other using OneShot optimization.

Empirical results: We experiment with three datasets in real-world heterogeneous FL settings and show that E2FL can help clients from both majority and minority groups, while q-FFL (Li et al. 2020b), state-of-the-art fair FL, improves equality by helping the majorities and ignoring the minorities. Since there are no existing distributed datasets containing different groups of clients with different data distributions, we create a new dataset, called FairMNISTRotate, to evaluate equality and equity in FL applications. FairMNISTRotate represents 10 different handwriting styles, produced by rotating samples from the MNIST dataset. For FairMNISTRotate, q-FFL reduces the variance of accuracies for all clients by 4% while it increases the variance between groups by 81% compared to FedAvg. On the other hand, our algorithm reduces the variance of both clients and groups by 93% and 95% respectively compared to FedAvg. That is, E2FL improves both equity and equality, unlike prior fair FL algorithms.

2 Background

Federated Learning: In FL (McMahan et al. 2017; Kairouz et al. 2019; Konečný et al. 2016), N clients collaborate to train a global model without directly sharing their data. In round t, the service provider (server) selects n out of N total clients and sends them the most recent global model θ^t. Each client trains a local model for E local epochs on their data starting from the θ^t using stochastic gradient descent (SGD). Then the client sends back the calculated gradients to the server. The server then aggregates the collected gradients and updates the global model for the next round.

Rank-based Federated Learning: Federated Rank Learning (FRL) (Mozaffari, Shejwalkar, and Houmansadr 2021) is an approach to perform FL that is built on a novel learning paradigm called *supermasks* (Zhou et al. 2019; Ramanujan et al. 2020). Specifically, in FRL, clients collaborate to find a *subnetwork* within a *randomly initialized* neural network (the supernetwork), where this is in contrast to conventional FL where clients collaborate to *train* neural network parameter weights. The goal of training in FRL is to collaboratively identify a supermask M_g, which is a binary mask of 1's and 0's, that is superimposed on the random neural network to obtain the final subnetwork, i.e., $\theta^w \bigodot M_g$ where θ^w is showing the weight parameters for supernetwork. M_g contains the edges of top k ranks, i.e., edges in top k ranks (layer-wise) get '1' in the binary mask, and others get '0' in the mask. We use $k = 50\%$ in this work to find a subnetwork of 50% of the original size. The subnetwork is then used for downstream tasks, e.g., image classification, hence it is equivalent to the global model in conventional FL. Note that in entire FRL training, weights of the network (θ^w) do not change.

More specifically, each FRL client computes the importance of the edges of the supernetwork based on their local data. The importance of the edges is represented as a ranking vector. Each FRL client will use the edge-popup algorithm (Ramanujan et al. 2020) and their data to compute their local rankings (the edge-popup algorithm aims at learning which edges in the random network are more important over the other edges by minimizing the loss of the subnetwork on their local data). Each client then will send their local edge ranking to the server. Finally, the FRL server uses a *voting mechanism* to aggregate client rankings into a supermask, which represents which edges of the random neural network (the supernetwork) will form the global subnetwork. Specifically, the FRL server uses Borda count rank aggregation method (Emerson 2013) where it gives a reputation to each edge for each ranking, sums the reputations, and sorts them from least to most to find the global ranking. We defer further details of FRL and edge-popup Algorithms to Appendix D.

3 Fairness Using Two Lenses: Equity and Equality

Fairness in FL can be evaluated from two main perspectives: a) *Equality* which is fairness between individuals and b) *equity* which is fairness between groups. A group is a set of individuals with the same protected attribute. The protected attribute may be known to the clients, e.g., race, gender, or age. Alternatively, clients may be unaware of their particular group, e.g., handwriting style (as this needs clustering clients into groups by someone who has samples from all clients).

Figure 1 shows an example of two FL systems where six clients want to learn a global model for prediction of handwritten digits. These clients have three handwriting styles: **(A)** normal handwriting style, **(B)** a little bit rotated handwriting, and **(C)** 180 degree rotated handwriting (upside-down). We consider each model update (θ_u^q for client u in group q) has the same effect on updating the global model, so each client update is like a vote. In this example, group A

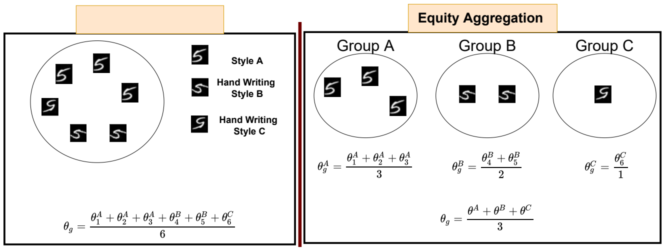

Equity Aggregation
Style A
Hand Writing Style B
Hand Writing Style C
$\theta_g = \frac{\theta_1^A + \theta_2^A + \theta_3^A + \theta_4^B + \theta_5^B + \theta_6^C}{6}$
Group A
Group B
Group C
$\theta_g^A = \frac{\theta_1^A + \theta_2^A + \theta_3^A}{3}$
$\theta_g^B = \frac{\theta_4^B + \theta_5^B}{2}$
$\theta_g^C = \frac{\theta_6^C}{1}$
$\theta_g = \frac{\theta^A + \theta^B + \theta^C}{3}$

Figure 1: An example showing two different FL systems with two goals: equality (on left) and equity (on right).

has the majority of the voters, and group B and C are in minorities. The left part of figure shows an FL in which the goal is providing equality, so we give same chance (one vote) to each client to change the final model by an aggregation such as averaging (e.g., what we have in FedAvg). In this setting, the majority group with higher population (group A) has more influence on the final vote. On the other hand, the right part shows an FL in which the goal is providing equity. In this setting, first we aggregate the votes inside each group to find the group votes ($\theta_g^A, \theta_g^B, \theta_g^C$), and then aggregate the group votes to produce the final model. In this setting, each client has the same chance (one vote) to influence its own group vote, and finally each group of voters have the same chance (one vote) to influence the final vote. We define two aspects of fairness in FL as follows:

Definition 1 (Equality: User-level Fairness): Trained global model θ is more *equalized* when its performance is more uniform across the individual clients participating in FL, i.e., when STD$\{F_u(\theta)\}_{u\in[N]}$ is smaller where STD$\{.\}$ is the standard deviation, and $F_u(.)$ denotes the local objective function of client u from N clients. Existing fair federated learning literature (Li et al. 2020b, 2021b; Smith et al. 2017; Hashimoto et al. 2018; Zhang et al. 2021; Mohri, Sivek, and Suresh 2019; Yu, Bagdasaryan, and Shmatikov 2020) use this definition in their designs.

Definition 2 (Equity: Group-level Fairness): Trained global model θ is more *equitable* when its performance is more uniform across the groups, i.e., when STD$\{\text{Avg}\{F_u(\theta)\}_{u\in[q]}\}_{q\in[Q]}$ is smaller where AVG$\{\}_{u\in[q]}$ denotes the average of performances for all the individual clients in the qth group, and there are Q total groups.

In E2FL, our goal is providing both equity and equality. To provide equity, an individual client has one vote in their group, and each group has one vote among all the groups. To provide equality, we allow the clients in each group use their group model which represents their training data better.

4 E2FL: Design

In this section, we provide the design of our Equal and Equitable Federated Learning (E2FL) algorithm. We first describe how E2FL provides equity and then discuss how it provides equality. The intuition behind E2FL is to train multiple global models, and first perform a majority vote among the clients' model updates in each group, and then another majority vote among the group models to find the global model. Algorithm 1 describes E2FL training.

Algorithm 1: Equal and Equitable Federated Learning (E2FL) Algorithm.

```
Input: number of FL rounds $T$, number of local epochs $E$, number of selected users in each round $n$, number of groups $Q$, seed SEED, learning rate $\eta$, subnetwork size $k\%$
$\theta^s, \theta^w \leftarrow$ Initialize random scores and weights of global model $\theta$ using SEED
$R_g^1 \leftarrow$ ArgSort$(\theta^s)$   ▷ Sort the initial scores and obtain initial global rankings
for $t \in [1, T]$ do
    $U \leftarrow$ set of $n$ randomly selected clients out of $N$ total clients
    for $u$ in $U$ do
        $\theta^s, \theta^w \leftarrow$ Initialize scores and weights using SEED
        If $q$ (group ID) is not known, use Algorithms 2 and 3 for ID inference (Section 5)
        $\theta^s[R_g^t] \leftarrow$ sort$(\theta^s)$   ▷ Reorder the scores based on the global ranking
        $\theta_u^s \leftarrow$ Edge-PopUp$(E, D_u^{tr}, \theta^w, \theta^s, k, \eta)$ ▷ Train local scores on the local training data
        $R_{u,q}^t \leftarrow$ ArgSort$(\theta_u^s)$   ▷ Ranking of the client $u$ with estimated group ID: $q$
        return $R_{u,q}^t$
    end for
    $R_{g,q\in[Q]}^{t+1} \leftarrow$ Vote$(R_{u\in U,q\in[Q]}^t)$   ▷ Majority vote aggregation inside each group
    $R_g^{t+1} \leftarrow$ Vote$(R_{g,q\in[Q]}^{t+1})$   ▷ Majority vote aggregation among all the groups
end for
function Vote $(R_{\{u\in U\}})$:
    $V \leftarrow$ ArgSort$(R_{\{u\in U\}})$   ▷ Reputation of each edge in each local ranking
    $A \leftarrow$ Sum$(V)$   ▷ Sum the reputations
    return ArgSort$(A)$   ▷ Order of the reputations
end function
```

The key insight used in E2FL is converting the problem of model weight optimization (in standard FL) to the problem of ranking model edges (Mozaffari, Shejwalkar, and Houmansadr 2021). In Section 2, we explained how an FL works by training on parameter ranks. In E2FL, each local update (one vote) is a ranking, i.e., a permutation of integers $\in [1, d]$ where d is the size of network layer. We use rankings because of their intrinsic fairness feature: In rank aggregation, each local ranking has the same impact on the aggregated global ranking. In rank-based FL, all the local rankings are bounded to be a permutation of unique integers $\in [1, d]$. For example, for a network layer with $d = 3$ parameters, there are only 3! possible permutations for local ranking $([1, 2, 3], \ldots, [3, 2, 1])$. However, in existing standard FL designs, the local updates ($\in \mathbb{R}^d$) have different impacts on the aggregated global model because the direction and magnitude of each parameter update is not bounded to other parameters.

In E2FL, different FL clients gather together to learn a global model, but each one of them belongs to a different group (which could be considered known or unknown). In this section, we assume the clients know their group IDs, and in Section 5, we explain how the clients can infer their group IDs using the features of rankings when the groups are unknown. In E2FL, the server trains multiple global rankings, each one belonging to a different group. These global group rankings are showing different orders of importance of same supernetwork for different groups from least to most important edges. Each client participates in the training of their group model by sending the local ranking they have. For aggregation, the server performs a majority vote among the local rankings (local votes) in each group, and then performs another majority vote among global group rankings (group votes) to find the global model for the next round (i.e., global ranking that clients will start their training for next E2FL round).

Edge-PopUp Algorithm The edge-popup (EP) algorithm (Ramanujan et al. 2020) is an optimization method to find supermasks within a large, randomly initialized neural network, i.e., a supernetwork, with performances close to the fully trained supernetwork. EP algorithm does not train the weights of the network, instead only decides the set of edges to keep and removes the rest of the edges (i.e., pop). Specifically, EP algorithm assigns a positive score (θ^s) to each of the edges in the supernetwork and updates it. In E2FL, each client learns its local scores θ^s_u by using EP on its local data for E local epochs starting from the global scores θ^s. On forward pass, it selects the top $k\%$ edges with highest scores, where k is the percentage of the total number of edges in the supernetwork that will remain in the final subnetwork. On the backward pass, it updates the scores with the straight-through gradient estimator (Bengio, Léonard, and Courville 2013). Algorithm 5 in Appendix D presents EP algorithm.

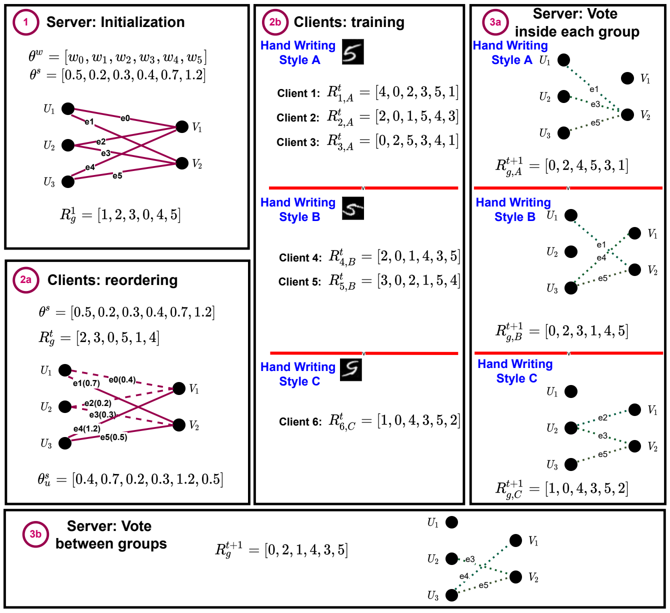

1 Server: Initialization
$\theta^w = [w_0, w_1, w_2, w_3, w_4, w_5]$
$\theta^s = [0.5, 0.2, 0.3, 0.4, 0.7, 1.2]$
U_1
U_2
U_3
V_1
V_2
e0
e1
e2
e3
e4
e5
$R^1_g = [1, 2, 3, 0, 4, 5]$
2a Clients: reordering
$\theta^s = [0.5, 0.2, 0.3, 0.4, 0.7, 1.2]$
$R^t_g = [2, 3, 0, 5, 1, 4]$
e0(0.4)
e1(0.7)
e2(0.2)
e3(0.3)
e4(1.2)
e5(0.5)
$\theta^s_u = [0.4, 0.7, 0.2, 0.3, 1.2, 0.5]$
2b Clients: training
Hand Writing Style A
Client 1: $R^t_{1,A} = [4, 0, 2, 3, 5, 1]$
Client 2: $R^t_{2,A} = [2, 0, 1, 5, 4, 3]$
Client 3: $R^t_{3,A} = [0, 2, 5, 3, 4, 1]$
Hand Writing Style B
Client 4: $R^t_{4,B} = [2, 0, 1, 4, 3, 5]$
Client 5: $R^t_{5,B} = [3, 0, 2, 1, 5, 4]$
Hand Writing Style C
Client 6: $R^t_{6,C} = [1, 0, 4, 3, 5, 2]$
3a Server: Vote inside each group
Hand Writing Style A
$R^{t+1}_{g,A} = [0, 2, 4, 5, 3, 1]$
Hand Writing Style B
$R^{t+1}_{g,B} = [0, 2, 3, 1, 4, 5]$
Hand Writing Style C
$R^{t+1}_{g,C} = [1, 0, 4, 3, 5, 2]$
3b Server: Vote between groups
$R^{t+1}_g = [0, 2, 1, 4, 3, 5]$

Figure 2: A single E2FL round with six clients from three groups and a network of 6 edges. Note that all the operations in E2FL training are performed in a layer-wise manner.

We detail a round of E2FL training and depict it in Figure 2, where we use a supernetwork with six edges $e_{i\in[0,5]}$ to demonstrate a single E2FL round and consider six clients $C_{j\in[1,6]}$ from three groups (handwriting style A, B, C) who aim to find a subnetwork of size k=50% of the original supernetwork.

Server: Initialization Phase (Only for round $t = 1$) : In the first round, the E2FL server chooses a random seed SEED to generate initial random weights θ^w and scores θ^s for the global supernetwork θ; note that, θ^w, θ^s, and SEED remain constant during the entire E2FL training. Next, the E2FL server shares SEED with E2FL clients, who can then locally reconstruct the initial weights θ^w and scores θ^s using SEED. Figure 2-① depicts this step. Recall that, the goal of E2FL training is to find the most important edges in θ^w without changing the weights. At the beginning, the E2FL server finds the global rankings of the initial random scores , i.e., $R^1_g = \text{ARGSORT}(\theta^s)$. We define *rankings of a vector* as the indices of elements of vector when the vector is sorted from low to high, which is computed using ARGSORT function.

Clients: Calculating the ranks (For each round t) : In the t^{th} round, E2FL server shares the global rankings R^t_g with the clients. Each of the clients locally reconstructs the weights θ^w's and scores θ^s's using SEED. Then, each E2FL client reorders the random scores based on the global rankings, R^t_g. We depict this in Figure 2-(2a). For instance, the initial global rankings for this round are $R^t_g = [2, 3, 0, 5, 1, 4]$, meaning that edge e_4 should get the highest score ($s_4 = 1.2$), and edge e_2 should get the lowest score ($s_2 = 0.2$).

Next, each of the clients uses reordered θ^s_u and finds a subnetwork within θ^w using edge-popup algorithm (Ramanujan et al. 2020); to find a subnetwork, they use their local data and E local epochs. Note that, each iteration of edge-popup algorithm updates the scores θ^s_u. Then client u computes their local rankings R^t_u using the final updated scores and ARGSORT(.), and sends $R^t_{u,q}$ to the server where q is the group identifier. We will explain the group inference methods we propose in Section 5. Figure 2-(2b) shows, for each client, the local rankings they obtained after finding their local subnetwork. For example, rankings of client C_1 are $R^t_{1,A} = [4, 0, 2, 3, 5, 1]$, i.e., e_4 is the least important and e_1 is the most important edge for C_1. Considering desired subnetwork size to be 50%, C_1 uses edges {3,5,1} in their final subnetwork in this round.

Server: Majority Vote (For each round t) : The server receives all the local rankings of the clients, i.e., $\{R^t_{1,A}, R^t_{2,A}, R^t_{3,A}, R^t_{4,B}, R^t_{5,B}, R^t_{6,C}\}$. Then, it performs a majority vote over all the local rankings inside each group, i.e., $\{A, B, C\}$. We depict this in Figure 2-(3a). Note that, for group q, the index i in $R^{t+1}_{g,q}$ represents the importance of the edge ith for clients in group q. For instance, in Figure 2-(3a), rankings of A are $R^t_{g,A} = [0, 2, 4, 5, 3, 1]$ and rankings of B are $R^t_{g,B} = [0, 2, 3, 1, 4, 5]$, hence the edge e_1 is the most important edge for group A, while the edge e_5 is the most important edge for group B. Next, the server performs a majority vote over all the group rankings of different groups $\{R^{t+1}_{g,A}, R^{t+1}_{g,B}, R^{t+1}_{g,C}\}$ to find the global ranking R^{t+1}_g. We depict this in Figure 2-(3b).

E2FL provides both equity and equality. Equity is not the main goal of existing distributed learning systems because it can hurt the motivation of the majority of clients to participate in the FL. If we have a learning algorithm that provides equity, it has this constraint to allow the same contribution from all groups (i.e., majorities and minorities). This comes with the price of reducing the performance of the majorities, which can demotivate them to participate to learn a model.

E2FL provides both equity and equality. In this algorithm, at the final round of the learning, instead of using the global ranking, each group uses its own group global rankings. The global rankings can provide better performances to the majority groups as they have access to more training data, so they can train better group global rankings. For example, a client of handwriting style A will use $f(x, \theta^w \bigodot M^t_{g,A})$ in their downstream classification task, where $M^t_{g,A}$ is the learned binary mask for group A at FL round t, and θ^w is the random weights (initialized randomly and kept fixed), and x is the test input. Note that in E2FL and its variants, $M^t_{g,q}$ is the supermask trained for group q where for top $k\%$ of the top rankings of group ranking $R^t_{g,q}$, we put 1's and we set other masks to 0's.

5 E2FL when Group IDs are Unknown

In the previous section, we assumed that the clients know their group IDs. In this section, we explain the approaches the server or a client can utilize to estimate the group IDs when the groups are unknown. In this setting, there are federated clients that have small amount of data with no known protected attribute. For instance, people with their own style of handwriting want to learn a global model by learning a local model on the images of their handwriting. The clients have no knowledge about their style as it is not something to be identified. It is not even possible to announce that clients with similar style collaborate with each other. This type of scenario usually happens in cross-device setting, where each user has a small dataset and there are so many clients in the system. For the server-side approach, the server clusters the local rankings into different groups and assigns a group ID to each client. For client-side approaches, each client should estimate its own group ID using the binary masks learned so far (Algorithm 1 line 8).

Communication Cost of E2FL when group IDs are unknown: Please note that for finding the best group ID at the client-side, there is no need to send all the group rankings (e.g., $\{R^{t+1}_{g,A}, R^{t+1}_{g,B}, R^{t+1}_{g,C}\}$ in our example) to the clients. As we mentioned before, in E2FL each ranking (local or global) can be converted to a binary mask of '0's and '1's that is superimposed on the random weights (i.e., supermask). Thus, the server only broadcasts the binary masks of groups (e.g., $\{M^{t+1}_{g,A}, M^{t+1}_{g,B}, M^{t+1}_{g,C}\}$ in our example) to the clients so they can estimate their group ID where they belong to.

5.1 Server-side: Rank Clustering

Working with rankings enables us to design an efficient algorithm to cluster the local rankings. Clustering rankings is more efficient than clustering the model weight updates in standard FL. The main reason is that rankings are from a *discrete* space ($\in perm([1, d])$, all the possible permutation of integers $\in [1, d]$ where d is the layer size) while model updates in standard FL are from a *continuous* space ($\in \mathbb{R}^d$). In this approach, all the clients should learn a local rankings on their local data at the beginning of the learning, and send it to the server. Then the server clusters the rankings into Q clusters to find the group ID of each client. This is just one time clustering, and throughout the E2FL learning, the server selects the local rankings for different group rank aggregation (majority voting) based on their group IDs (estimated with this approach).

Algorithm 2 in Appendix C shows how the server can cluster the local rankings of N clients into Q groups. We adapt K-means clustering to cluster the rankings. In this algorithm, at first step (Algorithm 2 line 3), we are choosing Q random rankings as our initial Q clusters, called centroids. Then we assign the cluster ID of the closest centroid for all the N local rankings (Algorithm 2 line 7-10). To determine the distance of two rankings, we use Spearman rank distance in which the distance between two rankings of R_1 and R_2 is $D(R_1, R_2) = \sum_{\ell \in [L]} \sum_{i \in [n_\ell]} |R_1[i] - R_2[i]|$ where $R_1[i]$ shows the rank of parameter ith in ranking R_1, L is the number of layers in the network, and n_ℓ shows the number of parameters in layer ℓ. In next step (Algorithm 2 line 11), we are updating the centroid of each cluster by applying majority vote on the ranking inside each cluster. We repeat this process for T iterations to find final Q groups of rankings.

5.2 Client-side: Lowest Loss

In this approach, each client estimates its group ID by choosing the group that its binary mask produces the lowest loss. Thus, in each E2FL round, the server broadcasts all the binary masks related to existing groups ($M^t_{g,q \in [Q]}$), and each selected client calculates the loss for each binary mask on its training data. Algorithm 3 line 2-4 in Appendix C shows this approach. This approach was used by IFCA (Ghosh et al. 2020), where in their algorithm in each training round, the server broadcasts all the model parameters to clients, and then they can find the lowest loss group. Note that our E2FL, compared to IFCA, needs $\times 32 (\times 64)$ less download bandwidth in each round because it is working on the binary masks.

5.3 Client-side: Entropy of the output

Binary masks enable us to utilize other approaches for group inference. In these approaches (Wortsman et al. 2020), the client can infer the group ID using gradient-based optimization to find a linear superposition of learned binary masks which minimizes the output entropy. Wortsman et al. (2020) proposed these solutions to learn multiple tasks without catastrophic forgetting in continual learning. In these solutions, each client infers the group ID by choosing the most confident binary mask that produces more stable results. There are two variants of this approach as follows:

OneShot inference: Algorithm 3 line 5-9 in Appendix C shows this approach. At E2FL training round t, the server broadcasts Q leaned binary masks $M^t_{g,q \in [Q]}$ to the selected clients. Next, each client assigns a confidence coefficient (α_q) to each binary mask. Each α_q represents how much the client is confident that qth binary mask is its match. Then it calculates the output of the model as the weighted superposition of these masks (i.e., $p(\alpha) = f\left(x, \theta^w \bigodot(\sum_{q=1}^{Q} \alpha_q M_{g,q}^t)\right)$). The α is initialized in a way that all the masks have equal chance ($\alpha_0[q] = \frac{1}{Q}$ for $q \in [Q]$). Now, Each client tries to find the perfect α_q that minimizes the entropy of the outputs $H(p(\alpha))$ by applying gradient descent with respect to α just for one round, i.e., $\alpha \leftarrow \alpha - \beta \nabla_\alpha H(p(\alpha))$. In this approach, the client chooses the group ID q that changing its confidence level (α_q) has the most impact on the entropy of the output of mixed models, i.e., $\arg\max_q \left(-\frac{\partial H(p(\alpha)))}{\partial \alpha_q}\right)$.

Binary Search: Algorithm 3 line 10-23 in Appendix C shows this approach. In this approach, the client utilizes binary search to find the best group ID by removing half of the candidates at each step until one α_q remains nonzero which indicates the best candidate. The client calculates the $p(\alpha)$ for α, then it applies the gradient descent with respect to α on entropy of $p(\alpha)$. Next, the client eliminates half of the coefficients where they produce gradients less than the median of the gradients.

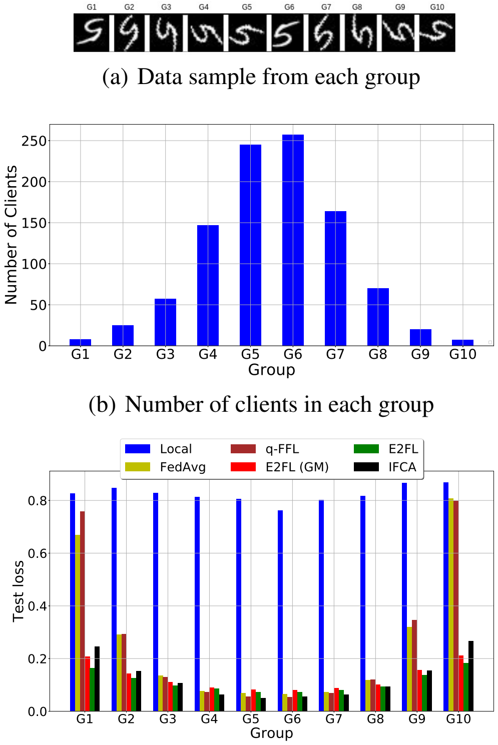
6 Experiments

In this section, we investigate the utility, fairness, and communication cost of E2FL in two different settings of known and unknown group IDs. We use three benchmark datasets widely used in prior works on federated learning application. We run all the experiments for 5 runs with different seeds, and report the average of them. Due to space limitations, we defer a detailed discussion of datasets, model architectures, hyperparameters, and the baselines to Appendix E.

6.1 Equality vs Equity via E2FL

FairMNISTRotate: To measure the equity and equality, we release a new dataset for fairness experiments in FL application. We note that creating different data distributions by manipulating standard datasets such as MNIST has been widely adopted in the continual research community (Goodfellow et al. 2013; Kirkpatrick et al. 2017; Lopez-Paz and Ranzato 2017), therefore, we create this dataset by rotating the images in MNIST for each group. Figure 3a shows image samples in each group, and Figure 3b shows the number of clients in each group. In this dataset, we assign same number of data samples to 1000 clients with 10 different data distributions (with different number of clients in each group). There are majority groups with large number of clients, e.g., G6 with 257 clients, and there are minority groups with small number of clients, e.g., G1 with 8 clients. In this dataset G1 and G10 are minorities, and G5 and G6 are in majorities.

In Figure 3c, we compare the performance of different FL algorithms by plotting the test loss of the global trained model for all the ten groups. In addition to the results of E2FL, which is the performance of the global ranking trained for each group, we report the results of the global model as E2FL (GM). We additionally compare the utility, equity, equality, and communication cost of these baselines in Table 1.



(a) Data sample from each group

(b) Number of clients in each group

(c) Final test loss for all the groups

Figure 3: FairMNISTRotate: a new dataset to investigate equality and equity in FL application.

Our experimental results on FairMNISTRotate show that: **(1) clients have motivation to participate in FL**. All the groups including minorities and majorities get benefit by participating in an FL framework. If each client wants to learn a local model on its local data, they cannot use other clients' knowledge, so their local models perform poorly. **(2) FedAvg gives more attention to majority groups.** FedAvg focuses on the clients from majority groups. Client from majority groups can get more benefit by participating in FedAvg as they have more chance to be selected in each round, so they have more impact on the global model. FedAvg can achieve 97.61% mean test accuracy for all the individual clients (i.e., user-level fairness), but the mean of accuracies for groups is low as 93.89% which shows that this learning paradigm is focusing on user-level fairness more than on group-based fairness (equity). **(3) q-FFL improves equality while worsens equity**. q-FFL is helping the majority groups by ignoring the minorities. q-FFL is a user-level fairness framework, so it makes the results more fair compared to FedAvg in equality; however it produces more unfair results compared in equity. **(4) Training 10 different FLs (i.e., IFCA) is not the best situation for the minorities**. It is important that all the groups in FL share their knowledge. Figure 3c shows that groups G1, G2, and G10 cannot get similar benefits by participating in IFCA since there is no shared knowledge, and these clients have access to limited data. There is another drawback for IFCA compared to E2FL which is communication cost. E2FL clients need to get the

Table 1: Comparing equality, equity, and communication cost of different variant of FLs on FairMNISTRotate with 1000 clients.

Approach	Metric									
	Group-level Fairness (Equity)				User-level Fairness (Equality)				Comm Cost	
	Avg	Worst (10%)	Best (10%)	Variance	Avg	Worst (10%)	Best (10%)	Variance	Up (MB)	Down (MB)
Local training	84.78	84.28	85.35	0.11	85.03	81.36	87.78	3.44	0	0
FedAvg	93.89	81.88	98.32	31.81	97.61	92.87	98.32	4.49	6.20	6.20
IFCA	**97.78**	**95.01**	**99.08**	1.98	**98.79**	**96.86**	**99.08**	0.35	6.20	62.0
q-FFL	92.23	77.31	98.42	57.46	97.56	93.33	98.42	4.32	6.20	6.20
Our E2FL	96.52	93.90	97.81	**1.63**	97.48	96.07	97.81	**0.33**	**4.05**	**5.99**

Table 2: Comparing utility, equality, and communication cost on FEMNIST with 3400 default clients.

Approach	Metric					
	User-level Fairness (Equality)				Communication Cost	
	Average	Worst (10%)	Best (10%)	Variance	Up (MB)	Down (MB)
Local training	68.74	44.45	87.41	154.50	0	0
FedAvg	85.50	63.51	99.39	108.24	6.23	6.23
q-FFL	84.40	64.09	99.14	100.80	6.23	6.23
Our E2FL (OneShot)	83.52	60.37	98.31	121.44	4.06	6.01
Our E2FL (Rank Clustering)	87.40	67.30	**100**	88.54	**4.06**	**4.06**
Our E2FL (Lowest Loss)	**87.93**	**68.59**	99.96	**81.88**	4.06	6.01

binary masks compared to IFCA that they need to get actual weight parameters so E2FL consumes 5.99 MB compared to IFCA which needs 62.0 MB for download bandwidth. Additionally, IFCA cannot get benefit by our entropy approaches since they are designed for binary masks. **(5) E2FL is providing equality and equity**. While q-FFL reduces the variance of accuracies for all the clients by 4% while it increases the variance between groups by 81% compared to FedAvg. On the other hand our algorithm can reduce both variance of clients and groups by 93% and 95% respectively compared to FedAvg

6.2 E2FL when Group IDs are Unknown

In this section, we provide experimental results on the FEMNIST (Caldas et al. 2018) which is a character recognition classification task distributed over 3,400 clients. At first, it seems that data distribution among all the clients are similar as all of them are classifying handwritten letters or digits, but there might be hidden groups of clients among these 3400 clients with even more similar handwriting styles. In this section, we report the performance of E2FL with different group inference approaches.

In Table 2, we compare the performance and fairness of different FL algorithms on FEMNIST. We show the user-level fairness (equality) metrics, as there are not specific groups defined for this dataset. Our experimental results show that: **(1)** Training on local data compared to FedAvg does not provide utility, thus all the clients have incentive to participate in the FL to get more accurate models. **(2)** q-FFL (Li et al. 2020b) can provide more equal results by reducing the variance by 7% with cost of reducing the accuracy by 1.10%. The reason behind this accuracy reduction is that q-FFL gives more attention to making the clients performances more uniform for the clients in majority groups, so it is using smaller portion of the knowledge from minorities . **(3)** E2FL shows clear advantage over other algorithms, it can provide both utility and fairness. As the groups are unknown in FEMNIST, we use three different approaches of group inference. For Lowest Loss and OneShot approaches, we use 2 static groups, and for rank clustering we use 5 clusters. We also report more details of 2, 3, 4, and 5 clusters in Appendix B Table 6. E2FL has good features from both worlds: it can improves the utility and fairness together: EFFL increases the average of accuracies by 2.5% and reduces the variance of clients by 24%. These benefits are coming from a) *training multiple models each for one group*, and b) *using knowledge of all the participating clients for all the clients by combining all the group models into one global model.* **(4)** Lowest Loss and rank clustering perform better than OneShot inference approach because in OneShot we have just one forward and backward passes compared to other methods that needs more operations.

Miscellaneous Discussions Due to space limitations, we defer detailed discussion of ablation studies of E2FL to Appendix. In Appendix A, we consider adult census income dataset which is commonly used in fair machine learning literature. In this experiment, we give data samples of different groups to different clients. In Appendix B, we evaluate the performance of different group inference approaches we introduced on FairMNISTRotate and FEMNIST.

7 Conclusions

In this paper, we look at the fairness issue in FL with two different lenses. First we define equality and equity as fairness in user-level and group-level, respectively. We designed a novel collaborative learning algorithm, called **E**qual and **E**quitable **F**ederated **L**earning (E2FL) to achieve both equality and equity. We validate the efficiency and fairness of E2FL in different real-world FL applications, and show that E2FL outperforms existing baselines in terms of the resulting efficiency, fairness of different groups, and fairness among all individual clients.

Acknowledgements

This work was supported by NSF grant 2131910.

References

Abay, A.; Zhou, Y.; Baracaldo, N.; Rajamoni, S.; Chuba, E.; and Ludwig, H. 2020. Mitigating bias in federated learning. *arXiv preprint arXiv:2012.02447*.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Caldas, S.; Wu, P.; Li, T.; Konečný, J.; McMahan, H. B.; Smith, V.; and Talwalkar, A. 2018. LEAF: A Benchmark for Federated Settings. *CoRR*, abs/1812.01097.

Cohen, G.; Afshar, S.; Tapson, J.; and van Schaik, A. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks, IJCNN*.

Dauphin, Y. N.; and Bengio, Y. 2013. Big neural networks waste capacity. *arXiv preprint arXiv:1301.3583*.

Denil, M.; Shakibi, B.; Dinh, L.; Ranzato, M.; and de Freitas, N. 2013. Predicting parameters in deep learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, 2148–2156.

Emerson, P. 2013. The original Borda count and partial voting. *Social Choice and Welfare*, 40(2): 353–358.

Ezzeldin, Y. H.; Yan, S.; He, C.; Ferrara, E.; and Avestimehr, S. 2021. Fairfed: Enabling group fairness in federated learning. *arXiv preprint arXiv:2110.00857*.

Frankle, J.; and Carbin, M. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR*.

Ghosh, A.; Chung, J.; Yin, D.; and Ramchandran, K. 2020. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*.

Goodfellow, I. J.; Mirza, M.; Xiao, D.; Courville, A.; and Bengio, Y. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Hashimoto, T.; Srivastava, M.; Namkoong, H.; and Liang, P. 2018. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

Kohavi, R.; et al. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, 202–207.

Konečnỳ, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Li, A.; Sun, J.; Wang, B.; Duan, L.; Li, S.; Chen, Y.; and Li, H. 2020a. LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets. *CoRR*.

Li, A.; Sun, J.; Zeng, X.; Zhang, M.; Li, H.; and Chen, Y. 2021a. FedMask: Joint Computation and Communication-Efficient Personalized Federated Learning via Heterogeneous Masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 42–55.

Li, T.; Hu, S.; Beirami, A.; and Smith, V. 2021b. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*.

Li, T.; Sanjabi, M.; Beirami, A.; and Smith, V. 2020b. Fair resource allocation in Federated learning. In *International Conference on Learning Representations*.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30: 6467–6476.

Ludwig, H.; Baracaldo, N.; Thomas, G.; and Zhou, Y. 2020. IBM Federated Learning: an Enterprise Framework White Paper V0.1. *CoRR*, abs/2007.10987.

Mansour, Y.; Mohri, M.; Ro, J.; and Suresh, A. T. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. A. y. 2017. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics*.

Mohri, M.; Sivek, G.; and Suresh, A. T. 2019. Agnostic federated learning. In *International Conference on Machine Learning*, 4615–4625. PMLR.

Mozaffari, H.; Shejwalkar, V.; and Houmansadr, A. 2021. FSL: Federated Supermask Learning. *arXiv preprint arXiv:2110.04350*.

Paulik, M.; Seigel, M.; and and, H. M. 2021. Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications. *CoRR*, abs/2102.08503.

Ramanujan, V.; Wortsman, M.; Kembhavi, A.; Farhadi, A.; and Rastegari, M. 2020. What's Hidden in a Randomly Weighted Neural Network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11893–11902.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. 2017. Federated Multi-Task Learning. In *Neural Information Processing Systems (NIPS)*.

Wortsman, M.; Ramanujan, V.; Liu, R.; Kembhavi, A.; Rastegari, M.; Yosinski, J.; and Farhadi, A. 2020. Supermasks in Superposition. In *Advances in Neural Information Processing Systems 33: NeurIPS*.

Yu, T.; Bagdasaryan, E.; and Shmatikov, V. 2020. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*.

Zhang, D. Y.; Kou, Z.; and Wang, D. 2020. Fairfl: A fair federated learning approach to reducing demographic bias in privacy-sensitive classification models. In *2020 IEEE International Conference on Big Data (Big Data)*, 1051–1060. IEEE.

Zhang, M.; Sapra, K.; Fidler, S.; Yeung, S.; and Alvarez, J. M. 2021. Personalized Federated Learning with First Order Model Optimization. In *International Conference on Learning Representations*.

Zhou, H.; Lan, J.; Liu, R.; and Yosinski, J. 2019. Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. In *Advances in Neural Information Processing Systems 32: NeurIPS*.

A Fairness with Multiple Groups in Each Client

We also provide additional experimental results on a real-world dataset Adult Census Income Dataset (Kohavi et al. 1996). This dataset contains 48,842 samples extracted from the United States Census Database and classifies whether individuals earn more or less than 50K per year. Prediction values are mapped to 0 ($\leq$ 50k) and 1 ($>$ 50k) where output of $\hat{Y} = 1$ is regarded as a positive output (i.e., making more money). We consider gender as the protected attribute where we consider male samples ($A = 1$) as the privileged group and female samples ($A = 0$) as unprivileged users. We split the data into train and test, and Table 3 shows the bias in this dataset for difference in their opportunities for making higher income ($Pr[\hat{Y} = 1|A = a]$ where $a \in \{0, 1\}$). This dataset is biased towards male group where the male group has a higher chance of 31.4% of getting $\hat{Y} = 1$ while female group has a chance of 11.3% of getting a positive prediction. The models that are trained on these data become more biased towards the male samples in test time by predicting $\hat{Y} = 1$ with a higher chance towards the male data input.

Table 3: Training and test data for male and female samples on Adult dataset.

Protected Attr	Stats	train data	test data
Gender	Pr$[A = 1]$	67.50%	67.50%
	Pr$[A = 0]$	32.5%	32.5%
	Pr$[\hat{Y} = 1\|A = 1]$	**31.4%**	**30.8%**
	Pr$[\hat{Y} = 1\|A = 0]$	**11.3%**	**11.3%**

We distribute the data samples among 5 clients using Dirichlet distribution with two settings: a) independent and identically distributed (IID) with Dirichlet parameter of $\alpha =$ 5000, and b) non-independent and identically distributed (non-IID) with Dirichlet parameter of $\alpha = 1$. We use two metrics to measure the fairness in this dataset that are used in previous works (Ezzeldin et al. 2021; Abay et al. 2020; Zhang, Kou, and Wang 2020). First, we use equal opportunity difference (EOD) (i.e., $EOD = Pr(\hat{Y} = 1|A = 0, Y = 1) - Pr(\hat{Y} = 1|A = 1, Y = 1)$) where it measures the true positive rate difference of majority and minority group. Second, we use discrimination index (DI) (i.e., $DI = F1(\theta|A = 0) - F1(\theta|A = 1)$) where it measures the F1 score difference between two groups.

Table 4: Fairness of E2FL compared to other baselines on Adult dataset.

Algorithm	Metric	Heterogeneity Level α	
		5000 (IID)	1 (Non-IID)
FedAvg	Test Accuracy	85.56	85.47
	EOD$_{te}$	-0.0689	-0.0834
	DI$_{te}$	-0.0432	-0.0517
FairFed	Test Accuracy	85.1	84.47
	EOD$_{te}$	-0.0701	-0.069
	DI$_{te}$	-0.0441	-0.041
E2FL	Test Accuracy	85.22	85.20
	EOD$_{te}$	-0.0174	-0.0222
	DI$_{te}$	-0.019	-0.0252

Table 4 shows the fairness comparison between FedAvg (Konečný et al. 2016; McMahan et al. 2017), FairFed (Ezzeldin et al. 2021), and E2FL for different data distributions. We choose FairFed as a baseline as it provides group fairness when we have different data groups (protected attributes) at each client, similar to what we have in the Adult dataset situation. This algorithm adaptively modifies the aggregation weights at the server in each round. The weights are based on the mismatch between the global fairness measure (at the server) and local fairness measure at each client. This algorithm is favouring clients whose local measures match more with the global fairness measure. This table shows E2FL reduces the equal opportunity difference (EOD) and discrimination index (DI) of male and female groups with a small cost on the final test accuracy. In particular, for non-IID data distribution, E2FL improves the EOD by 73% and DI by 51%, with the negligible cost of losing test accuracy (0.27%) compared to FedAvg, while FairFed improves EOD by 17% and DI by 20% with reducing the test accuracy by 1% compared to FedAvg. From this table, we can see that FairFed cannot provide the same improvement when data is IID distributed as its design is based on very heterogeneous data distribution, while E2FL achieves similar improvement in both cases. These results show that E2FL enforces the model to act more fair even when each client has a combination of all the groups.

B Missing Experiments when Group IDs are Unknown

Missing experiments for rank clustering E2FL on FairMNISTRotate: Table 5 shows the accuracy of our rank clustering approach for the local rankings in E2FL. In this table we show the average of 10 different runs. In this approach, each E2FL client learns a local ranking for $E = 2$ local epochs on it local data at the beginning of E2FL and send the local ranking to the server, so the server can assign different group IDs to the clients. From this table we can see that with larger rankings, we can predict group ID with higher accuracy.

Missing experiments for rank clustering E2FL on FEMNIST: Table 6 shows the equity and equality measurements of using E2FL on FEMNIST for different numbers of clusters. We can see by increasing the number of clusters, we

Table 5: Accuracy of group inference for rank clustering on FairMNISTRotate with 1000 clients by collecting local rankings that are trained for 2 local epochs. We show the accuracy of right prediction by using rankings of different layers with different number of parameters.

Layer	No Params	Accuracy of predicting the right group (%)
Conv1	288	91.78
Conv2	18432	94.94
FC1	1605632	93.76
FC2	1280	91.99
ALL	1625632	93.21

can cover more diverse groups so the results are more fair.

Missing experiments for client-based group inference approaches: Figure 4 shows the accuracy of group inference approaches by using three methods we proposed (on the client-side) for the 300 initial global epochs of E2FL on the FairMNISTRotate. We can see that the Lowest Loss approach produces more accurate results compared to Binary and OneShot approaches with cost of calculating Q forward passes. Also, the Binary method can produce better results compared to OneShot as it requires more forward and backward passes to determine the group ID.

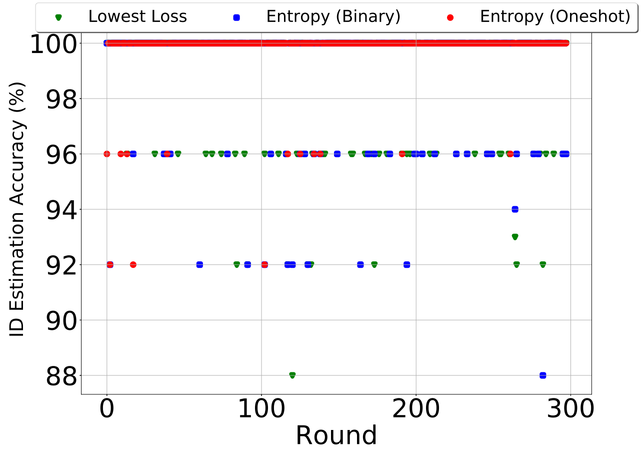


Figure 4: Accuracy of group inference approaches proposed for E2FL for 300 initial global epochs on FairMNISTRotate

C Missing Details of Group Inference Algorithms

In this section, we discuss the approaches proposed in Section 5 for E2FL group inference. Algorithm 2 shows how rank clustering at server works. Algorithm 3 shows three approaches we suggest to be used at client-side for group inference.

Time complexity comparison: Lowest loss approach needs to have $\mathcal{O}(Q)$ forward passes to find the binary mask that produces the lowest loss for Q groups. Binary search over the entropy needs to have $\mathcal{O}(\log(Q))$ forward and backward passes. Finally, the OneShot inference only requires $\mathcal{O}(1)$ forward and backward passes, so it makes the estimation process very fast.

Algorithm 2: Identity Inference with Rank Clustering

```
Input: number of clients $N$, Local rankings $R_{\{i\in[N]\}}$, number of clusters $Q$, number of iterations $T$
function RANKCLUSTERING ($R_{\{i\in[N]\}}$)
    CENTROIDS $\leftarrow$ pick $Q$ random rankings from $R_{\{i\in[N]\}}$
    $r \leftarrow 0$                                          ▷ iteration counter
    while $r < T$ do
        CLUSTERSRANKING $\leftarrow$ [[] for $q \in [Q]$)]
        for $u \in [N]$ do
            $q \leftarrow$ GETCLOSESTCLUSTER(CENTROIDS, $R_i$)
            CLUSTERSRANKING$[q]$.append$(u)$
        end for
        CNETROIDS $\leftarrow$ FRLVOTE(CLUSTERSRANKING)
        $r+=1$
    end while
    return CLUSTERSRANKING
end function
```

Algorithm 3: Identity Inference at Client Side

```
Input: training data $D_u^{tr}$, random weights $\theta^w$, number of groups $Q$, group binary masks $M_{g,q\in[Q]}^t$, loss function $loss(.)$
function LOWESTLOSS ($\theta^w, Q, M_{g,q\in[Q]}^t, D_u^{tr}$)
    return $\text{argmin}_{q\in[Q]}\, loss\left(D_u^{tr}, \theta^w \bigodot M_{g,q\in[Q]}^t\right)$
end function
function ONESHOT ($\theta^w, Q, M_{g,q\in[Q]}^t, D_u^{tr}$)
    $\alpha \leftarrow [\frac{1}{Q}, \frac{1}{Q}, ..., \frac{1}{Q}]$
    $p(\alpha) \leftarrow f\left(D_u^{tr}, \theta^w \bigodot (\sum_{q=1}^{Q} \alpha_q M_{g,q}^t)\right)$
    return $\text{argmax}_{q\in[Q]}\left(-\frac{\partial H(p(\alpha)))}{\partial \alpha_q}\right)$
end function
function BINARY ($\theta^w, Q, M_{g,q\in[Q]}^t, D_u^{tr}$)
    $\alpha \leftarrow [\frac{1}{Q}, \frac{1}{Q}, ..., \frac{1}{Q}]$
    while $||\alpha||_0 > 1$ do
        $p \leftarrow f\left(D_u^{tr}, \theta^w \bigodot (\sum_{q=1}^{Q} \alpha_q M_{g,q}^t)\right)$
        $g \leftarrow \nabla_\alpha H(p)$
        for $q \in [Q]$ do
            if $g_q \leq$ median$(g)$ then
                $\alpha_q \leftarrow 0$
            end if
        end for
        $\alpha \leftarrow \alpha/||\alpha||_1$
    end while
    return $\text{argmax}_{q\in[Q]}\,(\alpha_q)$
end function
```

D Missing Details of Federated Rank Learning (FRL)

Supermask Learning: Modern neural networks have a very large number of parameters. These networks are generally overparameterized (Dauphin and Bengio 2013; Denil et al. 2013; Li et al. 2020a, 2021a), i.e., they have more parameters than they need to perform a particular task,

Table 6: Comparing utility, equality and equity of E2FL using rank clustering for different numbers of clusters on FEMNIST with 3400 default clients.

Approach	Number of Clusters	Metric							
		Group-level Fairness (Equity)				User-level Fairness (Equality)			
		Average	Worst group	Best group	Variance	Average	Worst (10%)	Best (10%)	Variance
E2FL	2	88.12	85.65	90.60	6.10	87.21	66.36	99.87	94.09
	3	88.03	84.74	91.33	10.82	86.36	64.82	99.66	101.80
	4	88.01	85.01	91.39	5.42	87.12	65.81	100	98.80
	5	87.61	85.31	91.06	3.80	87.40	67.30	100	88.54

e.g., classification. The *lottery ticket hypothesis* (Frankle and Carbin 2019) states that a fully-trained neural network, i.e., *supernetwork*, contains sparse *subnetworks*, i.e., subsets of all neurons in the supernetwork, which can be trained from scratch (i.e., by training same initialized weights of the subnetwork) and achieve performances close to the fully trained supernetwork. The lottery ticket hypothesis allows for massive reductions in the sizes of neural networks. Ramanujan et al. (Ramanujan et al. 2020) offer a complementary conjecture that an overparameterized neural network with randomly initialized weights contains subnetworks which perform as good as the fully trained network. To do so, we should identify a supermask M_g, which is a binary mask of 1's and 0's, that is superimposed on the random neural network to obtain the final subnetwork, i.e., $\theta^w \bigodot M_g$ where θ^w is showing the weight parameters for supernetwork.

Federated Rank Learning (FRL): In this section, we provide the design of federated rank learning (FRL) algorithm (Mozaffari, Shejwalkar, and Houmansadr 2021). FRL clients collaborate (without sharing their local data) to *find a subnetwork* within a randomly initialized, untrained neural network called the *supernetwork*. Algorithm 4 describes FRL's training. Training a global model in FRL means first finding a unanimous ranking of supernetwork edges and then using the subnetwork of the top-ranked edges as the final output.

Edge-popup Algorithm Algorithm 5 presents Edge-popup algorithm (Ramanujan et al. 2020).

Algorithm 4: Federated Rank Learning (FRL)

```
Input: number of FL rounds $T$, number of local epochs $E$, number of selected users in each round $n$, seed SEED, learning rate $\eta$, subnetwork size $k\%$
Server: Initialization
$\theta^s, \theta^w \leftarrow$ Initialize random scores and weights of global model $\theta$ using SEED
$R_g^1 \leftarrow \textsc{ArgSort}(\theta^s)$    ▷ Sort the initial scores and obtain initial rankings
for $t \in [1, T]$ do
    $U \leftarrow$ set of $n$ randomly selected clients out of $N$ total clients
    for $u$ in $U$ do
        Clients: Calculating the ranks
        $\theta^s, \theta^w \leftarrow$ Initialize scores and weights using SEED
        $\theta^s[R_g^t] \leftarrow \textsc{sort}(\theta^s)$    ▷ sort the scores based on the global ranking
        $S \leftarrow$ Edge-PopUp$(E, D_u^{tr}, \theta^w, \theta^s, k, \eta)$    ▷ Client u uses Algorithm5 to train a supermask on its local training data
        $R_u^t \leftarrow \textsc{ArgSort}(S)$    ▷ Ranking of the client
    end for
    Server: Majority Vote
    $R_g^{t+1} \leftarrow \textsc{Vote}(R_{u \in U}^t)$ ▷ Majority vote aggregation
end for
function $\textsc{Vote}(R_{\{u \in U\}})$:
    $V \leftarrow \textsc{ArgSort}(R_{\{u \in U\}})$
    $A \leftarrow \textsc{Sum}(V)$
    return $\textsc{ArgSort}(A)$
end function
```

E Missing Details of Experiment Setup

E.1 Datasets and Model Architectures

In this section, we discuss our hyperparameters for each baseline and each dataset. In all the experiments we use SGD as the optimizer, and we use a momentum of 0.9 with weight decay of 1e-4. We tune the learning rate and local epochs for each dataset and each baseline to find the best results. We run all the experiments for 3000 global epochs and report the final results. q-FFL converges slower, so we allow 6000 global epochs for q-FFL.

FairMNISTRotate: We experiment with LeNet architecture given in Table 7. For local training in each E2FL round, each client uses 2 epochs with learning rate of 0.1. For FedAvg, we use 2 local epochs with learning rate of 0.01. For q-FFL, we set the q-FFL fairness hyperparameter to 0.1 along with the same settings of FedAvg. We use batch size of 8, and we select 25 client (out of 1000) in each FL round for all the experiments.

FEMNIST (Caldas et al. 2018; Cohen et al. 2017) is a character recognition classification task with 3,400 clients, 62 classes (52 for upper and lower case letters and 10 for digits), and 671,585 grey-scale images. Each client has data of their own handwritten digits or letters. We use LeNet architecture given in Table 7. For local training in each E2FL round, each client uses 2 epochs with learning rate of 0.05. For FedAvg, we use 2 local epochs with learning rate of 0.05. For q-FFL, we set the q-FFL fairness hyperparameter to 0.1 with the same setting of FedAvg. We use batch size of

Table 7: Model architectures.

Architecture	Layer Name	Number of params
LeNet (FairMNISTRotate)	Convolution(32) + Relu	288
	Convolution(64) + Relu	18432
	MaxPool(2x2)	-
	FC(128) + Relu	1605632
	FC(10)	1280
LeNet (FEMNIST)	Convolution(32) + Relu	288
	Convolution(64) + Relu	18432
	MaxPool(2x2)	-
	FC(128) + Relu	1605632
	FC(62)	7936
FC (Adult)	FC(1024) + Relu	103424
	FC(1024) + Relu	1048576
	FC(2)	2048

Algorithm 5: Edge-popup (EP) algorithm: it finds a subnetwork of size $k\%$ of the entire network θ

```
Input: number of local epochs $E$, training data $D$, initial weights $\theta^w$ and scores $\theta^s$, subnetwork size $k\%$, learning rate $\eta$
for $e \in [E]$ do
    $\mathcal{B} \leftarrow$ Split $D$ in $B$ batches
    for batch $b \in [B]$ do
        EP FORWARD $(\theta^w, \theta^s, k, b)$
        $\theta^s = \theta^s - \eta \nabla \ell(\theta^s; b)$
    end for
end for
return $\theta^s$
function EP FORWARD$(\theta^w, \theta^s, k, b)$
    $m \leftarrow$ sort$(\theta^s)$
    $t \leftarrow int((1-k) * len(m))$
    $m[: t] = 0$
    $m[t :] = 1$
    $\theta^p = \theta^w \odot \mathbf{m}$
    return $\theta^p(b)$
end function
```

10, and we select 25 clients (out of 3400) in each FL round for all the experiments.

Adult Adult Census Income Dataset (Kohavi et al. 1996) is a dataset commonly used for fair machine learning literature. The dataset consists of anonymous information such as gender, race, education, occupation, etc. We experiment with a simple 3-layer fully connected network given in Table 7. We consider gender as the protected attribute where we consider male samples $(A = 1)$ as the privileged group and female samples $(A = 0)$ as unprivileged users. For experiments on the Adult dataset, we use all of the updates $(n = N = 5)$ in each round. We train for 20 global epochs, and in each global epoch, we ask the clients to run for 5 local epochs using SGD as the optimizer with learning rate of 2.0, momentum of 0.9 and weight decay of 1e-4 for all the experiments.

E.2 Missing Details of Baselines

Federated averaging (FedAvg) (Konečnỳ et al. 2016; McMahan et al. 2017): FedAvg is an effective aggregation rule (AGR) where due to its efficiency, Average is the only AGR implemented by FL applications in practice (Ludwig et al. 2020; Paulik, Seigel, and and 2021).

IFCA (Ghosh et al. 2020) This algorithm assumes that users are partitioned into different clusters, and their goal is to train a separate model for each cluster. Their main challenge was how to identify the cluster membership of each user. For their experiments, they assume that they know the number of clusters. Also in case of ambiguous cluster structure, they treat the number of clusters as a hyperparameter to tune. They propose Iterative Federated Clustering Algorithm (IFCA) that solves simultaneously two problems: 1) identifies the cluster membership of each user and 2) optimizes each cluster model. In IFCA, the server will send all the models to selected clients in each round, and each client will pick the best model that produces the least amount of loss and participates in the training of that cluster. On the server side, it is just FedAvg for several clusters.

Local training: A framework in which the clients train standalone models on local datasets without collaboration.

FedFair (Ezzeldin et al. 2021): This algorithm adaptively modifies the aggregation weights at the server in each round. The weights are based on the mismatch between the global fairness measure (at the server) and the local fairness measure at each client. This algorithm is favouring clients whose local measures match more with the global fairness measure.

q-FFL (Li et al. 2020b): Inspired by fair resource allocation algorithms for wireless networks, Li et al. (Li et al. 2020b) design a federated optimization technique called q-FFL. This technique aims at improving FL fairness by minimizing the aggregate reweighed loss, in a way that the devices with higher loss are given higher relative weights.