

# 000 001 002 003 004 005 ACHIEVING EXACT FEDERATED UNLEARNING WITH 006 IMPROVED POST-UNLEARNING PERFORMANCE 007 008 009

010 **Anonymous authors**  
 011 Paper under double-blind review  
 012  
 013  
 014  
 015  
 016  
 017  
 018  
 019  
 020  
 021  
 022  
 023  
 024

## ABSTRACT

025 Federated learning is a machine learning paradigm that allows multiple clients  
 026 to train aggregated model via sharing model updates to a central server without  
 027 sharing their data. Even though the data is not shared, it can indirectly influence  
 028 the aggregated model via the shared model updates. In many real-life scenarios, we  
 029 need to completely remove a client’s influence (unlearning) from the aggregated  
 030 model, such as competitive clients who want to remove their influence from the  
 031 aggregated model after leaving the coalition to ensure other clients do not benefit  
 032 from their contributions. The influence removal is also needed when the adversarial  
 033 client negatively affects the aggregated model. Though the aggregated model can  
 034 be retrained from scratch to ensure exact unlearning (completely removing the  
 035 client’s influence from the aggregated model), it performs poorly just after the  
 036 unlearning, which is undesirable during deployment. To overcome this challenge,  
 037 this paper proposes federated unlearning algorithms that ensure exact unlearning  
 038 while achieving better performance post-unlearning. Our experimental results on  
 039 different real datasets validate the performance of the proposed algorithms.

## 1 INTRODUCTION

040 An individual user may have insufficient data to train a state-of-the-art machine learning model. Yet,  
 041 we can significantly improve the model performance by leveraging the combined data from multiple  
 042 users. *Federated learning* (FL) (Zhang et al., 2021) is one of the most prevalent paradigms to perform  
 043 such collaboration today, especially in sectors with strong privacy demands such as finance and health  
 044 care (Li et al., 2020; Xu et al., 2021). In the FL setting, collaborative clients train local models on  
 045 their own data, and a central server model is obtained by aggregating these local model updates for  
 046 multiple communication rounds. FL is well-suited for many commercial applications as it eliminates  
 047 the need to share users’ private data during training. For example, multiple companies from the same  
 048 industrial sector (e.g., banking, insurance, or healthcare) often possess diverse user data. To leverage  
 049 all available data, these companies can collaborate and train a more accurate model using suitable FL  
 050 algorithms (Aledhari et al., 2020).

051 Although FL algorithms do not directly access users’ data, the aggregated model is still influenced by  
 052 the local models trained on each client’s data. When a client leaves the collaboration, it is necessary  
 053 to update the aggregated model to remove the influence of its data—a process known as *federated  
 054 unlearning* (FU),<sup>1</sup> e.g., a company leaving the collaboration of many companies may demand the  
 055 removal of their contributions to ensure their competitors do not benefit from them. FU techniques  
 056 are also desirable to remove the influence of adversarial clients, i.e., the adversary behaves like a  
 057 client and degrades the model performance by contributing contaminated updates (Fang et al., 2020).  
 058 Additionally, the development of FU techniques facilitates the exercise of the *right to be forgotten*  
 059 formalized in many regional or government data regulations such as GDPR (2016) and CCPA (2018).

060 We can trivially achieve FU by retraining the collaboration from scratch without the target client’s  
 061 data (Liu et al., 2023). Despite its simplicity, the new server model suffers from low performance as  
 062 it is restarted with random initialization. As a result, it slows down the deployment of the unlearned  
 063 model, as training large models on the collaboration of many users can be time-consuming. Due to

064  
 065  
 066  
 067  
 068  
 069  
 070  
 071  
 072  
 073  
 074  
 075  
 076  
 077  
 078  
 079  
 080  
 081  
 082  
 083  
 084  
 085  
 086  
 087  
 088  
 089  
 090  
 091  
 092  
 093  
 094  
 095  
 096  
 097  
 098  
 099  
 100  
 101  
 102  
 103  
 104  
 105  
 106  
 107  
 108  
 109  
 110  
 111  
 112  
 113  
 114  
 115  
 116  
 117  
 118  
 119  
 120  
 121  
 122  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137  
 138  
 139  
 140  
 141  
 142  
 143  
 144  
 145  
 146  
 147  
 148  
 149  
 150  
 151  
 152  
 153  
 154  
 155  
 156  
 157  
 158  
 159  
 160  
 161  
 162  
 163  
 164  
 165  
 166  
 167  
 168  
 169  
 170  
 171  
 172  
 173  
 174  
 175  
 176  
 177  
 178  
 179  
 180  
 181  
 182  
 183  
 184  
 185  
 186  
 187  
 188  
 189  
 190  
 191  
 192  
 193  
 194  
 195  
 196  
 197  
 198  
 199  
 200  
 201  
 202  
 203  
 204  
 205  
 206  
 207  
 208  
 209  
 210  
 211  
 212  
 213  
 214  
 215  
 216  
 217  
 218  
 219  
 220  
 221  
 222  
 223  
 224  
 225  
 226  
 227  
 228  
 229  
 230  
 231  
 232  
 233  
 234  
 235  
 236  
 237  
 238  
 239  
 240  
 241  
 242  
 243  
 244  
 245  
 246  
 247  
 248  
 249  
 250  
 251  
 252  
 253  
 254  
 255  
 256  
 257  
 258  
 259  
 260  
 261  
 262  
 263  
 264  
 265  
 266  
 267  
 268  
 269  
 270  
 271  
 272  
 273  
 274  
 275  
 276  
 277  
 278  
 279  
 280  
 281  
 282  
 283  
 284  
 285  
 286  
 287  
 288  
 289  
 290  
 291  
 292  
 293  
 294  
 295  
 296  
 297  
 298  
 299  
 300  
 301  
 302  
 303  
 304  
 305  
 306  
 307  
 308  
 309  
 310  
 311  
 312  
 313  
 314  
 315  
 316  
 317  
 318  
 319  
 320  
 321  
 322  
 323  
 324  
 325  
 326  
 327  
 328  
 329  
 330  
 331  
 332  
 333  
 334  
 335  
 336  
 337  
 338  
 339  
 340  
 341  
 342  
 343  
 344  
 345  
 346  
 347  
 348  
 349  
 350  
 351  
 352  
 353  
 354  
 355  
 356  
 357  
 358  
 359  
 360  
 361  
 362  
 363  
 364  
 365  
 366  
 367  
 368  
 369  
 370  
 371  
 372  
 373  
 374  
 375  
 376  
 377  
 378  
 379  
 380  
 381  
 382  
 383  
 384  
 385  
 386  
 387  
 388  
 389  
 390  
 391  
 392  
 393  
 394  
 395  
 396  
 397  
 398  
 399  
 400  
 401  
 402  
 403  
 404  
 405  
 406  
 407  
 408  
 409  
 410  
 411  
 412  
 413  
 414  
 415  
 416  
 417  
 418  
 419  
 420  
 421  
 422  
 423  
 424  
 425  
 426  
 427  
 428  
 429  
 430  
 431  
 432  
 433  
 434  
 435  
 436  
 437  
 438  
 439  
 440  
 441  
 442  
 443  
 444  
 445  
 446  
 447  
 448  
 449  
 450  
 451  
 452  
 453  
 454  
 455  
 456  
 457  
 458  
 459  
 460  
 461  
 462  
 463  
 464  
 465  
 466  
 467  
 468  
 469  
 470  
 471  
 472  
 473  
 474  
 475  
 476  
 477  
 478  
 479  
 480  
 481  
 482  
 483  
 484  
 485  
 486  
 487  
 488  
 489  
 490  
 491  
 492  
 493  
 494  
 495  
 496  
 497  
 498  
 499  
 500  
 501  
 502  
 503  
 504  
 505  
 506  
 507  
 508  
 509  
 510  
 511  
 512  
 513  
 514  
 515  
 516  
 517  
 518  
 519  
 520  
 521  
 522  
 523  
 524  
 525  
 526  
 527  
 528  
 529  
 530  
 531  
 532  
 533  
 534  
 535  
 536  
 537  
 538  
 539  
 540  
 541  
 542  
 543  
 544  
 545  
 546  
 547  
 548  
 549  
 550  
 551  
 552  
 553  
 554  
 555  
 556  
 557  
 558  
 559  
 560  
 561  
 562  
 563  
 564  
 565  
 566  
 567  
 568  
 569  
 570  
 571  
 572  
 573  
 574  
 575  
 576  
 577  
 578  
 579  
 580  
 581  
 582  
 583  
 584  
 585  
 586  
 587  
 588  
 589  
 590  
 591  
 592  
 593  
 594  
 595  
 596  
 597  
 598  
 599  
 600  
 601  
 602  
 603  
 604  
 605  
 606  
 607  
 608  
 609  
 610  
 611  
 612  
 613  
 614  
 615  
 616  
 617  
 618  
 619  
 620  
 621  
 622  
 623  
 624  
 625  
 626  
 627  
 628  
 629  
 630  
 631  
 632  
 633  
 634  
 635  
 636  
 637  
 638  
 639  
 640  
 641  
 642  
 643  
 644  
 645  
 646  
 647  
 648  
 649  
 650  
 651  
 652  
 653  
 654  
 655  
 656  
 657  
 658  
 659  
 660  
 661  
 662  
 663  
 664  
 665  
 666  
 667  
 668  
 669  
 670  
 671  
 672  
 673  
 674  
 675  
 676  
 677  
 678  
 679  
 680  
 681  
 682  
 683  
 684  
 685  
 686  
 687  
 688  
 689  
 690  
 691  
 692  
 693  
 694  
 695  
 696  
 697  
 698  
 699  
 700  
 701  
 702  
 703  
 704  
 705  
 706  
 707  
 708  
 709  
 710  
 711  
 712  
 713  
 714  
 715  
 716  
 717  
 718  
 719  
 720  
 721  
 722  
 723  
 724  
 725  
 726  
 727  
 728  
 729  
 730  
 731  
 732  
 733  
 734  
 735  
 736  
 737  
 738  
 739  
 740  
 741  
 742  
 743  
 744  
 745  
 746  
 747  
 748  
 749  
 750  
 751  
 752  
 753  
 754  
 755  
 756  
 757  
 758  
 759  
 760  
 761  
 762  
 763  
 764  
 765  
 766  
 767  
 768  
 769  
 770  
 771  
 772  
 773  
 774  
 775  
 776  
 777  
 778  
 779  
 780  
 781  
 782  
 783  
 784  
 785  
 786  
 787  
 788  
 789  
 790  
 791  
 792  
 793  
 794  
 795  
 796  
 797  
 798  
 799  
 800  
 801  
 802  
 803  
 804  
 805  
 806  
 807  
 808  
 809  
 810  
 811  
 812  
 813  
 814  
 815  
 816  
 817  
 818  
 819  
 820  
 821  
 822  
 823  
 824  
 825  
 826  
 827  
 828  
 829  
 830  
 831  
 832  
 833  
 834  
 835  
 836  
 837  
 838  
 839  
 840  
 841  
 842  
 843  
 844  
 845  
 846  
 847  
 848  
 849  
 850  
 851  
 852  
 853  
 854  
 855  
 856  
 857  
 858  
 859  
 860  
 861  
 862  
 863  
 864  
 865  
 866  
 867  
 868  
 869  
 870  
 871  
 872  
 873  
 874  
 875  
 876  
 877  
 878  
 879  
 880  
 881  
 882  
 883  
 884  
 885  
 886  
 887  
 888  
 889  
 890  
 891  
 892  
 893  
 894  
 895  
 896  
 897  
 898  
 899  
 900  
 901  
 902  
 903  
 904  
 905  
 906  
 907  
 908  
 909  
 910  
 911  
 912  
 913  
 914  
 915  
 916  
 917  
 918  
 919  
 920  
 921  
 922  
 923  
 924  
 925  
 926  
 927  
 928  
 929  
 930  
 931  
 932  
 933  
 934  
 935  
 936  
 937  
 938  
 939  
 940  
 941  
 942  
 943  
 944  
 945  
 946  
 947  
 948  
 949  
 950  
 951  
 952  
 953  
 954  
 955  
 956  
 957  
 958  
 959  
 960  
 961  
 962  
 963  
 964  
 965  
 966  
 967  
 968  
 969  
 970  
 971  
 972  
 973  
 974  
 975  
 976  
 977  
 978  
 979  
 980  
 981  
 982  
 983  
 984  
 985  
 986  
 987  
 988  
 989  
 990  
 991  
 992  
 993  
 994  
 995  
 996  
 997  
 998  
 999  
 1000  
 1001  
 1002  
 1003  
 1004  
 1005  
 1006  
 1007  
 1008  
 1009  
 1010  
 1011  
 1012  
 1013  
 1014  
 1015  
 1016  
 1017  
 1018  
 1019  
 1020  
 1021  
 1022  
 1023  
 1024  
 1025  
 1026  
 1027  
 1028  
 1029  
 1030  
 1031  
 1032  
 1033  
 1034  
 1035  
 1036  
 1037  
 1038  
 1039  
 1040  
 1041  
 1042  
 1043  
 1044  
 1045  
 1046  
 1047  
 1048  
 1049  
 1050  
 1051  
 1052  
 1053  
 1054  
 1055  
 1056  
 1057  
 1058  
 1059  
 1060  
 1061  
 1062  
 1063  
 1064  
 1065  
 1066  
 1067  
 1068  
 1069  
 1070  
 1071  
 1072  
 1073  
 1074  
 1075  
 1076  
 1077  
 1078  
 1079  
 1080  
 1081  
 1082  
 1083  
 1084  
 1085  
 1086  
 1087  
 1088  
 1089  
 1090  
 1091  
 1092  
 1093  
 1094  
 1095  
 1096  
 1097  
 1098  
 1099  
 1100  
 1101  
 1102  
 1103  
 1104  
 1105  
 1106  
 1107  
 1108  
 1109  
 1110  
 1111  
 1112  
 1113  
 1114  
 1115  
 1116  
 1117  
 1118  
 1119  
 1120  
 1121  
 1122  
 1123  
 1124  
 1125  
 1126  
 1127  
 1128  
 1129  
 1130  
 1131  
 1132  
 1133  
 1134  
 1135  
 1136  
 1137  
 1138  
 1139  
 1140  
 1141  
 1142  
 1143  
 1144  
 1145  
 1146  
 1147  
 1148  
 1149  
 1150  
 1151  
 1152  
 1153  
 1154  
 1155  
 1156  
 1157  
 1158  
 1159  
 1160  
 1161  
 1162  
 1163  
 1164  
 1165  
 1166  
 1167  
 1168  
 1169  
 1170  
 1171  
 1172  
 1173  
 1174  
 1175  
 1176  
 1177  
 1178  
 1179  
 1180  
 1181  
 1182  
 1183  
 1184  
 1185  
 1186  
 1187  
 1188  
 1189  
 1190  
 1191  
 1192  
 1193  
 1194  
 1195  
 1196  
 1197  
 1198  
 1199  
 1200  
 1201  
 1202  
 1203  
 1204  
 1205  
 1206  
 1207  
 1208  
 1209  
 1210  
 1211  
 1212  
 1213  
 1214  
 1215  
 1216  
 1217  
 1218  
 1219  
 1220  
 1221  
 1222  
 1223  
 1224  
 1225  
 1226  
 1227  
 1228  
 1229  
 1230  
 1231  
 1232  
 1233  
 1234  
 1235  
 1236  
 1237  
 1238  
 1239  
 1240  
 1241  
 1242  
 1243  
 1244  
 1245  
 1246  
 1247  
 1248  
 1249  
 1250  
 1251  
 1252  
 1253  
 1254  
 1255  
 1256  
 1257  
 1258  
 1259  
 1260  
 1261  
 1262  
 1263  
 1264  
 1265  
 1266  
 1267  
 1268  
 1269  
 1270  
 1271  
 1272  
 1273  
 1274  
 1275  
 1276  
 1277  
 1278  
 1279  
 1280  
 1281  
 1282  
 1283  
 1284  
 1285  
 1286  
 1287  
 1288  
 1289  
 1290  
 1291  
 1292  
 1293  
 1294  
 1295  
 1296  
 1297  
 1298  
 1299  
 1300  
 1301  
 1302  
 1303  
 1304  
 1305  
 1306  
 1307  
 1308  
 1309  
 1310  
 1311  
 1312  
 1313  
 1314  
 1315  
 1316  
 1317  
 1318  
 1319  
 1320  
 1321  
 1322  
 1323  
 1324  
 1325  
 1326  
 1327  
 1328  
 1329  
 1330  
 1331  
 1332  
 1333  
 1334  
 1335  
 1336  
 1337  
 1338  
 1339  
 1340  
 1341  
 1342  
 1343  
 1344  
 1345  
 1346  
 1347  
 1348  
 1349  
 1350  
 1351  
 1352  
 1353  
 1354  
 1355  
 1356  
 1357  
 1358  
 1359  
 1360  
 1361  
 1362  
 1363  
 1364  
 1365  
 1366  
 1367  
 1368  
 1369  
 1370  
 1371  
 1372  
 1373  
 1374  
 1375  
 1376  
 1377  
 1378  
 1379  
 1380  
 1381  
 1382  
 1383  
 1384  
 1385  
 1386  
 1387  
 1388  
 1389  
 1390  
 1391  
 1392  
 1393  
 1394  
 1395  
 1396  
 1397  
 1398  
 1399  
 1400  
 1401  
 1402  
 1403  
 1404  
 1405  
 1406  
 1407  
 1408  
 1409  
 1410  
 1411  
 1412  
 1413  
 1414  
 1415

054 these shortcomings, it is natural to consider the following question: *How can we guarantee the exact*  
 055 *federated unlearning while ensuring better post-unlearning performance?*

057 This paper proposes two novel methods for achieving exact FU with improved post-unlearning  
 058 performance. The first method, Bi-Models Training (BMT) (Section 3.1), preserves isolated copies  
 059 of local models and reuses clients’ existing knowledge residing in these models during unlearning  
 060 for better aggregation. Despite being unlearning-friendly, these local models fail to capture the joint  
 061 influence of multiple clients on the global model. Training the power set of clients can capture the  
 062 influence of all possible influences of the clients but is computationally expensive and may lead  
 063 to *double influence*, where a client affects multiple sub-FL models. As a result, we propose the  
 064 second method, Multi-Models Training (MMT) (Section 3.2), that trains each sub-FL model on disjoint  
 065 subsets of clients to avoid double influence and aggregates the best sub-FL models upon unlearning to  
 066 achieve improved initialization of the aggregated model. We empirically justify the effectiveness of  
 067 BMT and MMT through multiple experiments on real-world vision and language datasets (Section 4).

## 068 1.1 RELATED WORKS

070 In this section, we now review the relevant work, especially in federated learning, machine unlearning,  
 071 and federated unlearning, to our problem setting.

073 **Federated Learning (FL).** FL emerges from the industrial needs to train centralized models on  
 074 large, decentralized data residing on users’ device (McMahan et al., 2017) and is particularly favored  
 075 in sectors requiring strong privacy guarantees, such as finance and health care (Li et al., 2020; Xu  
 076 et al., 2021). Based on the characteristics of the decentralized data, Yang et al. (2019) divided FL into  
 077 three categories: horizontal FL, vertical FL, and federated transfer learning. To optimize the federated  
 078 models, McMahan et al. (2017) proposed the FedAvg algorithm that averages local updates from  
 079 contributing clients and works well on independent and identically distributed (i.i.d.) data. However,  
 080 as real-world data is often heterogeneous (e.g., users with different demographics), subsequent works  
 081 have proposed new methods that target model architecture or algorithm design to alleviate model drift  
 082 that can degrade model performance (Zeng et al., 2023; Mu et al., 2023; Idrissi et al., 2021; Li et al.,  
 083 2021; Karimireddy et al., 2020). We refer the readers to Zhang et al. (2021) for a detailed survey of  
 various works covering different settings of federated learning.

084 **Machine Unlearning (MU).** MU aims to remove the influence of a selected subset of data from  
 085 the trained ML model. Based on the guarantee of removal, MU methods are broadly categorized  
 086 into exact unlearning and approximate unlearning (Nguyen et al., 2022; Wang et al., 2024). In  
 087 exact unlearning, we aim for an identical model to one that would have been obtained by retraining  
 088 without that data to be erased. Retraining is a method that trivially achieves exact unlearning but is  
 089 computationally expensive with large models and datasets. Existing works can exactly unlearn for  
 090 support vector machines (Cauwenberghs & Poggio, 2000), k-means (Ginart et al., 2019), random  
 091 forests (Brophy & Lowd, 2021). Bourtoule et al. (2021) partitions the entire training data set into a  
 092 few disjoint subsets and trains one base model with each of these subsets. Since each base model  
 093 is only trained with a subset of the original training data, the performance may be sub-optimal.  
 094 Approximate unlearning aims for a model whose distribution closely resembles that of the retrained  
 095 model. Guo et al. (2020) proposed a certified removal method to approximately unlearn linear model  
 096 by Newton-like update. Nguyen et al. (2020) minimizes the KL divergence between the approximate  
 posterior of the unlearned model and the retrained model under the variational inference framework.

098 **Federated Unlearning (FU).** Many recent works adapt machine unlearning to the federated learning  
 099 settings (Liu et al., 2020; Wang et al., 2021; Gong et al., 2021). Liu et al. proposed FedEraser, which  
 100 involves using historical updates from the server and local calibration training on the unlearned client.  
 101 The federated unlearning protocol proposed in this work can be used to unlearn an arbitrary subset of  
 102 clients without any constraint on the type of data each client possesses. At the same time, it requires  
 103 no participation of the unlearned client. Wang et al. proposed a channel pruning-based method to  
 104 selectively forget a specific class from the trained ML model. Such an approach has limited scope as  
 105 it is impractical to assume that each participant in the FL setting possesses precisely one class of data.  
 106 Gong et al. concerned with the setting where no centralized party/server is present, which does not  
 107 apply to the centralized FL setting. In terms of exact federated unlearning, Xiong et al. (2023) and  
 Tao et al. (2024) use quantization and sampling strategies, respectively, to get a checkpoint during  
 the FL training where the unlearned client’s data have not made a quantifiable impact and use it

as initialization for model retraining and since speed up the retraining process. On the other hand, [Qiu et al. \(2023\)](#) proposed to cluster the clients and train a few intermediate FL models and then subsequently obtain the global FL model through one-shot aggregation. At the unlearning stage, only the intermediate FL model where the unlearning client is present is retrained (and hence reducing the retraining cost). Our proposed method touches on both ideas and uses aggregation of a few sub-FL models to obtain a good initialization for much more efficient retraining. The way we obtained our sub-FL models trades-off between computation budget and post-unlearning performance, played an essential role in ensuring its effectiveness.

## 2 PROBLEM SETTING

**Federated Learning.** This paper considers the centralized federated learning (FL) setting with a trusted central server and multiple clients. In this setting, a central server shared an aggregated model with the clients and then each client trains this model on his dataset and send model updates (weights or gradients) to the central server, which then aggregates these updates to get a better aggregated model. In our setting, we assume that the number of clients participating in FL process varies over time. Let  $\mathcal{C}_t$  denote the set of participating clients at the beginning of the FL communication round  $t$ . An FL communication round (communication round for brevity) represents one cycle of model sharing by the central server with clients and then receiving the updated aggregated model.

Each client  $c \in \mathcal{C}_t$  has training dataset  $\mathcal{D}_{c,t}$  with  $n_{c,t}$  labeled samples, where each sample is drawn from the distribution  $\nu_c$  over  $\mathcal{X} \times \mathcal{Y}$ . Here,  $\mathcal{X}$  represents the input space, and  $\mathcal{Y}$  represents the label space. The learning model is denoted by  $h_\theta : \mathcal{X} \rightarrow \mathcal{Y}$  for model parameters  $\theta \in \mathbb{R}^d$ , where  $d$  is the number of model parameters. The loss incurred by the learning model  $h_\theta$  on a sample  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  is denoted by  $l(h_\theta(x), y)$ , which can be the root mean squared error (for regression problems) or cross-entropy loss (for classification problems).

After the communication round  $t$ , the loss incurred by the client  $c$  for model parameters  $\theta$  is the average loss of the model  $h_\theta$  on the samples in  $\mathcal{D}_{c,t}$  and defined by  $f_{c,t}(\theta) := \frac{1}{n_{c,t}} \sum_{s=1}^{n_{c,t}} l(h_\theta(x_{c,s}), y_{c,s})$ , where  $(x_{c,s}, y_{c,s})$  is the  $s$ -th sample in  $\mathcal{D}_{c,t}$ . The central server aims to find a learning model with the minimum average loss for each client. The server achieves this by finding a model  $\theta$  that minimizes the average clients' loss weighted by their respective number of samples, which is given by solving the following optimization problem in the communication round  $t$ :

$$\operatorname{argmin}_{\theta} \frac{1}{n_t} \sum_{c \in \mathcal{C}_t} n_{c,t} f_{c,t}(\theta) = \frac{1}{n_t} \sum_{c \in \mathcal{C}_t} \sum_{s=1}^{n_{c,t}} l(h_\theta(x_{c,s}), y_{c,s}), \quad (1)$$

where  $n_t = \sum_{c=1}^{C_t} n_{c,t}$ . Since the clients cannot share their local data  $D_{c,t}$  with the server (due to communication or privacy constraints), the optimization problem given in Eq. (1) must be solved in a federated manner by using the suitable FL algorithm (e.g., FedAvg ([McMahan et al., 2017](#))).

**Exact Federated Unlearning.** Let client  $c$  influence be completely removed from the aggregated model. Exact federated unlearning is the process of completely removing the influence of client  $c$  training data from the aggregated model, resulting in a model that is equivalent to the models trained without the training data of client  $c$ . However, the aggregated model resulting from retraining without the data of client  $c$  may have a poor performance in the initial round, which may not be expected when these models are deployed in practice. Therefore, our goal is to design methods that ensure exact federated unlearning while leading to an aggregated model with as high accuracy as possible.

## 3 EXACT FEDERATED UNLEARNING METHODS

Due to multiple communication rounds of the FL training, it becomes impossible to completely remove a client's data influence from the trained aggregated model. Therefore, the most straightforward way to achieve the exact federated unlearning is to restart the federated learning process from scratch with the remaining clients. This method of retraining the aggregated model from scratch is called *retraining from scratch* (RfS) ([Bourtoule et al., 2021](#); [Liu et al., 2023](#)). Although RfS is a simple method, the new model may have very low accuracy [in the initial rounds after unlearning](#) compared to the aggregated model before unlearning due to restarting the FL process with

random initialization of the aggregated model. Such performance reduction of the aggregated model may not be desirable during deployment in practice involving critical applications such as healthcare (Prayitno et al., 2021; Dhade & Shirke, 2024) and finance (Long et al., 2020). This shortcoming of RfS raises a natural question: How can we guarantee the exact federated unlearning while ensuring better post-unlearning performance? To answer this question, we propose two novel methods for achieving exact federated unlearning that completely remove the client’s influence while giving better post-unlearning performance than RfS.

### 3.1 BI-MODELS TRAINING

To get a better performing aggregated model post-unlearning, we must design a new FL training process that allows exact federated unlearning while having a better initialization than random initialization. One way to achieve better initialization is to design methods that can exploit the remaining clients’ existing knowledge. To do this, we propose a method named Bi-Models Training (BMT) that can be incorporated into any existing federated learning framework. The main idea of BMT is to have an additional model for each client that is only trained on its data, making these models unaffected by other clients’ training data. We refer to this model as *local model*. We use the term *global model* for referring to the aggregated model, which is trained using all client’s data and used for deployment. Next, we discuss how BMT can be incorporated into the different stages of any existing federated learning framework (as depicted in Fig. 1), namely: Initialization, FL Training, Unlearning, and New Client joining the FL process, whose details are given as follows.

**Initialization.** The central server starts the standard FL training process by randomly initializing the global aggregated model. This randomly initialized global model is then shared with all clients. Each client updates the global model using its local training data and then shares the model update (updated model or gradients) with the central server. As compared to the standard initialization in any FL training process, each client makes a copy of the locally updated global model<sup>2</sup> (i.e., local model). Since the initial global model is randomly initialized, these local models are, by design, isolated from the influence of other clients’ training data.

**FL Training.** After receiving the first model updates, the central server aggregates them to get the aggregated global model as per the underlying FL algorithm (McMahan et al., 2017; Shleizerger et al., 2020; Zhang et al., 2021). In each subsequent communication round, each client receives the updated global model from the central server and then trains it using its training data. After updating the global model, each client shares the model update with the central server. Besides the standard FL training process, each client also updates their local model using their training data.

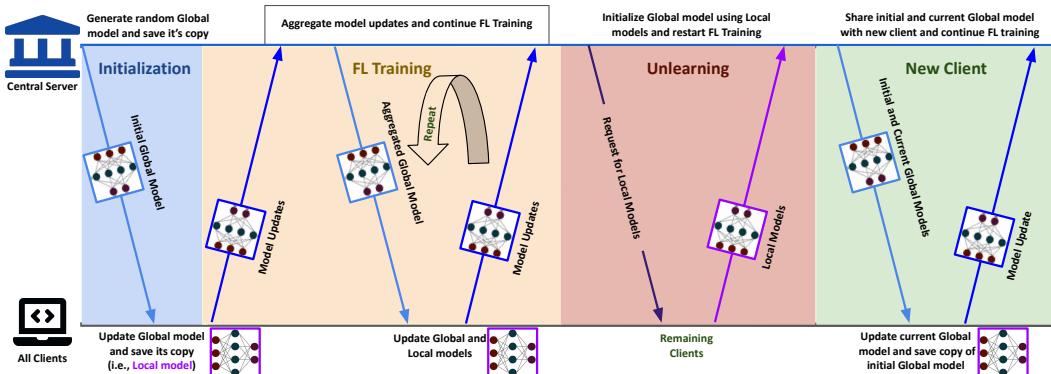


Figure 1: Bi-Models Training (BMT) in the different stages of any federated learning framework.

**Unlearning.** Let  $c$  be the client whose influence needs to be completely removed from the global model after a communication round  $t$  and  $\mathcal{C}_{t,r}$  be the set of remaining clients, i.e.,  $\mathcal{C}_{t,r} = \mathcal{C}_t \setminus \{c\}$ . The central server first discards the current global model and requests each client to share their current copy of local models. Once the central server receives the local models from all remaining clients, the central server aggregates them to get the new initialization for the global model as per the underlying

<sup>2</sup>The locally updated global model in the first communication rounds is the same as the model that is a copy of the initial global model and trained on client’s training data.

216 FL algorithm, e.g., for FedAvg, the central server performs weighted aggregation on the remaining  
 217 client’s local models, where each client’s weight is proportional to their respective training data. Our  
 218 extensive experimental results (in Section 4) show that the resulting initialized global model performs  
 219 better than random model initialization as done in RfS. Lastly, the central server restarts the FL  
 220 training process with the newly initialized global model, which is completely free from the influence  
 221 of the unlearned client’s data.

222 **New Client.** When a new client wants to join the ongoing FL collaboration, the central server waits  
 223 until the end of the ongoing communication round. Once it is over, the central server starts the FL  
 224 training process with the new client by sharing the current global model with the new client, who then  
 225 updates the current global model using its training data and shares the model update with the central  
 226 server. Apart from this, the central client also shares the randomly initialized global model with the  
 227 new client, who updates it, which then acts as the local model of the new client for subsequent rounds.  
 228 Other clients do not influence this local model, as the initial global model is randomly initialized.

229 In summary, BMT has two models for each client: global and local. All clients train their local  
 230 model on their data in isolation, whereas the global model is trained using the underlying FL training  
 231 protocol. To completely remove a client influence from the global model, the central server first  
 232 discards the global model and then uses the local models of the remaining clients to re-initialize the  
 233 global model, which is further updated via FL training. This process ensures that BMT, by design,  
 234 guarantees the exact federated unlearning. Further, using the remaining clients’ local models leads to  
 235 an initialization of the global model that is already influenced by the remaining clients to some extent,  
 236 leading to a better performance than RfS, as corroborated by our experiments in Section 4. The price  
 237 for this improved post-unlearning performance is the cost of pre-training the local models in advance.  
 238 Such a trade-off is worthwhile for applications that require exact unlearning and an unlearned model  
 239 with good performance as quickly as possible for deployment.

### 240 3.2 MULTI-MODELS TRAINING

242 The key insight of the previous section is that BMT achieves a better initial global model because it  
 243 is influenced by the clients’ local models. However, the local model only contains influence from  
 244 an individual client and has no joint influence of multiple clients. Since all clients influence the  
 245 global model, we should capture the joint influence of different clients and then use it to get a better  
 246 initialization of the global model. To capture the joint influence, we can train FL models using only a  
 247 subset of clients. We refer to these FL models as *sub-FL models*. Formally, a sub-FL model is an FL  
 248 model that is trained via FL protocol using a subset of clients, where the size of the subset varies  
 249 from 2 to  $N - 1$ . One can train all possible sub-FL models (power set of clients excluding global  
 250 model) to capture the influence of all possible interactions of different subsets of clients. However,  
 251 this approach is not computationally feasible as these sub-FL models increase exponentially with  
 252 the number of clients (i.e.,  $2^n - n - 2$  for  $n$  clients). Another problem of training arbitrary sub-FL  
 253 models leads to a situation of *double influence*, which is defined as follows:

254 **Definition 1.** Let  $S_i$  be the set of clients whose data are used in training the  $i$ -th sub-FL model. The  
 255 sub-models  $i$  and  $j$  leads to double influence if  $S_i \cap S_j \neq \emptyset$ ,  $S_i \setminus S_j \neq \emptyset$ , and  $S_j \setminus S_i \neq \emptyset$ .

256 When one client data is used to train two sub-FL models, it can lead to double influence if both are  
 257 also trained using data from different clients, e.g., one is trained on clients {1, 2} and another on  
 258 clients {1, 3}; the client 1 data is used in both sub-FL models and hence having the double influence.

259 To avoid the double influence, each sub-FL model should be trained on disjoint subsets of clients,  
 260 or the set of clients used for training sub-FL models is a proper superset of the set of clients used  
 261 for another sub-FL model. One possible way to achieve this is to organize sub-FL models in a  
 262 hierarchical tree structure. In this tree, the root node represents the global model while the leaf nodes  
 263 correspond to the local models, and intermediate nodes represent sub-FL models, with each child  
 264 node having disjoint sets of clients compared to its siblings. As we move from the root node to the  
 265 leaf nodes, each sub-FL model branches into further subsets, maintaining either disjoint relationships  
 266 or superset relations, thus ensuring a clear and systematic flow of influence throughout the hierarchy.  
 267 We refer to this hierarchical tree structure as an *influence tree*. After unlearning a client, we should  
 268 aggregate the sub-FL models with higher influence (those influenced by a larger number of clients)  
 269 and local models to get the initialization for the global model. If the number of models to aggregate  
 is less, it implies that the initialization of the global model contains the most joint influence of clients.

This relationship inspires our proposed metric *influence degradation score*, which measures how good is an influence tree. Next, we formally define the influence degradation score.

**Definition 2** (Influence Degradation Score (IDS)). *Let  $\mathcal{T}$  be any influence tree structure. The influence degradation score for  $\mathcal{T}$ , denoted by  $s(\mathcal{T})$ , is defined as the average number of sub-FL and local models that are aggregated to get the initial global model after unlearning any client.*

Though the tree structure, by design, eliminates double influence, we do not know which tree structure gives the lowest IDS for given clients' different likelihood of requesting unlearning (as the probability of requesting unlearning may vary across the clients). As our goal is to construct an influence tree with minimum IDS, our following result shows that the binary influence tree **constructed using Huffman coding** has the lowest IDS among all  $n$ -ary influence tree structures, where  $n > 2$ .

**Theorem 1.** *Given an  $n$ -ary influence tree  $\mathcal{T}$ , there exists a binary influence tree  $\mathcal{T}_2$  that has smaller IDS, i.e.,  $s(\mathcal{T}_2) < s(\mathcal{T})$ . Let  $p_c$  be the unlearning probability of the client  $c$ . Then, Huffman coding with  $n$  symbols representing clients and weights  $\{p_c\}_{c=1}^n$  gives the optimal binary influence tree such that  $s(\mathcal{T}_{Huffman}) \leq s(\mathcal{T}_2)$  for any influence tree  $\mathcal{T}_2$  for the same group of clients.*

With Theorem 1, we can use Huffman coding (Huffman, 1952) to construct an influence tree that has the lowest IDS among all types of influence trees. In some real-life applications, the client's unlearning probability can be unknown. In such cases, we can assume that each client is equally likely to be unlearned, hence having the same unlearning probability. We show the influence tree for 8 clients having the same unlearning probability in Fig. 3a. A client (on the leaf node) influences the sub-FL model if there is a path from a sub-FL model to the leaf node representing that client. We next propose a method named Multi-Models Training (MMT) that uses the sub-FL models to get better initialization for the global model. MMT can be easily incorporated into the different stages of any existing federated learning framework (as depicted in Fig. 1), whose details are given as follows.

**Initialization.** Similar to BMT, the central server starts the standard FL training process by randomly initializing the global aggregated model. This randomly initialized global model is then shared with all clients. Each client updates the global model using its local training data and then shares the model update (updated model or gradients) with the central server. Each client makes a copy of the locally updated global model. Compared to BMT, MMT also initializes the sub-FL models using the model updates of clients corresponding to the sub-FL models.

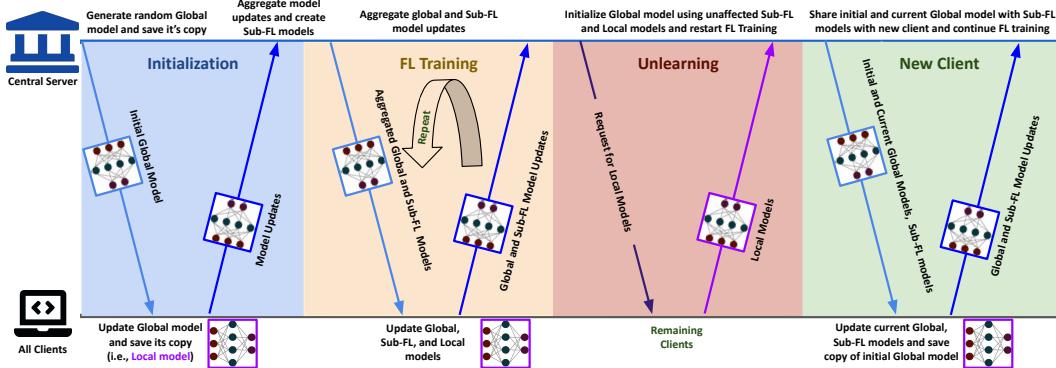


Figure 2: Multi-Models Training (MMT) in the different stages of any federated learning framework.

**FL Training.** After receiving the first model updates, the central server aggregates them to get the aggregated global and sub-FL models. In each subsequent communication round, each client receives the updated global and its sub-FL models from the central server and then trains them using its training data. After updating these models, each client shares the global and its sub-FL model updates with the central server. Apart from this, each client also updates their local model.

**Unlearning.** For unlearning a client, the central server first discards the current global model and related sub-FL models (as shown in Fig. 3b after unlearning client 2) and then requests all clients not in any of the remaining sub-FL models to share their current copy of local models. Once the central server receives all requested local models, it aggregates them with sub-FL models (choosing only the most influential unaffected sub-FL model over its descendants) to get the new initialization for the global model as per the underlying FL algorithm. After removing the sub-FL models related to the

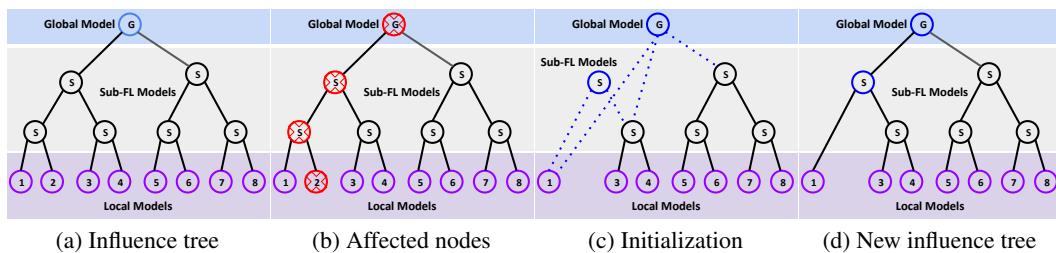


Figure 3: **Fig. 3a:** Influence Tree for 8 clients having the same unlearning probability. **Fig. 3b:** Showing the global and all sub-FL models affected after unlearning client 2 by the node’s red cross and red outline. **Fig. 3c:** Initialization of global and new sub-FL models, where a dotted blue line shows the the models used to initialized them. **Fig. 3d:** Final influence tree after unlearning the client.

unlearned client, the remaining influence tree may no longer have the lowest IDS for the remaining clients. It leads to two options: create a new influence tree while using earlier sub-FL models as much as possible (as shown in Fig. 3c) or keep using the existing influence tree, which may not be the best but retains the sub-FL models trained over time. Lastly, the central server restarts the FL training process with the newly initialized global and sub-FL (if any) models (as shown in Fig. 3d), which are completely free from the influence of the unlearned client’s data.

**New client.** Adding a new client to the ongoing FL collaboration can worsen the existing influence tree compared to the influence tree created using a new client. Like BMT, when a new client wants to join, the central server waits until the end of the ongoing communication round. Once it is over, the central server can create a new influence tree while using earlier sub-FL models as much as possible or keep using the existing influence tree to retain the existing sub-FL models, which are trained over time by adding new sub-FL models. After that, the central server starts the FL training process with the new client by sharing the current global and corresponding sub-FL models with the new client, who then updates the current models using its training data and shares the model updates with the central server. The central client also shares the randomly initialized global model with the new client, who updates it, which then acts as the local model of the new client for subsequent rounds.

Overall, the initialization of the global model in MMT has the joint influence of multiple clients, which makes it better than BMT and hence leads to better post-unlearning performance, as corroborated by our experiments in Section 4. However, note that there is an additional computational cost for this improved performance over BMT as we need to train multiple sub-FL models in parallel.

## 4 EXPERIMENTS

In this section, we empirically verify the effectiveness of the proposed methods in two important settings: (1) *sequential unlearning* setting, where multiple clients sequentially leave the federation, and (2) *continual learning and unlearning* setting, where clients can join and/or leave the federation at will. Subsequently, we analyze the impact of data heterogeneity and the branching factor in the MMT structure on the model performance. Then, we consider special scenarios when the clients follow a fixed unlearning order (e.g., according to their subscription plans) and clients with non-uniform unlearning probabilities (e.g., clients from different demographics).

### 4.1 EXPERIMENTAL SETTING

**Datasets.** We conduct our experiments on four popular vision datasets: MNIST (LeCun et al., 1998), FashionMNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009). We also consider language tasks with large language models in Section 4.5. To simulate clients with realistic non-IID data, we let the client  $i$  receives the most data from the  $i$ -th class and the same amount of data from the remaining classes. We use  $\rho$  to denote the ratio between the majority class and minority class for all clients. Each client contains 200 training/test samples and  $\rho = 0.02$  for MNIST and FashionMNIST; 1000 training samples, 300 test samples,  $\rho = 0.2$  for CIFAR-10; 400 training samples, 100 test samples,  $\rho = 0.1$  for CIFAR-100.

**Models.** For MNIST and FashionMNIST, we use simple MLP networks with 30 and 80 hidden units, respectively. For CIFAR-10, we use a CNN network with  $5 \times 5$  convolutional layers followed by  $2 \times 2$  max pool layer for feature extraction and two fully connected layers with 32 hidden units and ReLU for classification. For CIFAR-100, we use a VGG-16 model (Simonyan, 2014).

**Training.** We use FedAvg (McMahan et al., 2017) to train FL models for 100 rounds with 1 local epoch on MNIST and FashionMNIST, 300 rounds with 1 local epoch on CIFAR-10 and 100 rounds with 10 local epoch on CIFAR-100. We use the SGD optimizer with a learning rate 0.01, weight decay 0.1, batch size 20, and gradient clipping 10 for MNIST and FashionMNIST. We use the AdamW optimizer with batch size 64 and the same hyperparameters for CIFAR-10. We use the SGD optimizer with a learning rate 0.005, momentum 0.9, weight decay  $10^{-5}$ , and batch size 64 for CIFAR-100. Our experiments are conducted on NVIDIA L40 46GB and NVIDIA H100 80GB GPUs.

**Metrics.** We report test accuracy measured on a fixed test set that combines local test sets of all possible clients, including those that join/leave the federation in later stages. The combined test set allows us to observe the visible trend of the performance after one client is removed.

**Comparison Methods.** We compare BMT and MMT against the following baselines: Standalone, where the centralized model trains on aggregated data from all remaining clients; Retraining from Scratch (RFS), where the federated model is retrained excluding data from the leaving client; FedCIO (Qiu et al., 2023); Exact-Fun (Xiong et al., 2023); and FATS (Tao et al., 2024).

## 4.2 SEQUENTIAL UNLEARNING

This experiment simulates a practical scenario when clients gradually leave the federation. After each client leaves, we continue training the federated model on the remaining clients. Particularly, we simulate the leaving of 3 clients  $\{1, 3, 5\}$ . It is noteworthy that this unlearning order is to MMT’s disadvantage as none of the sub-FL models can completely replace the server model. Hence, the server parameters must be aggregated from the parameters of other sub-FL models.

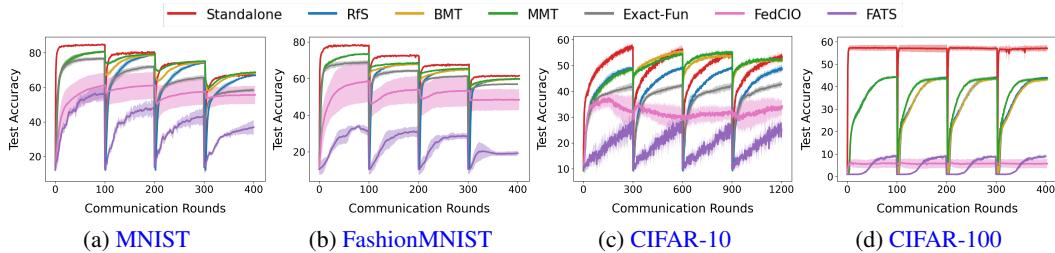


Figure 4: Test accuracy in the sequential unlearning setting for different datasets.

Fig. 4 shows performance in the sequential unlearning setting on different datasets. As can be seen, BMT and MMT consistently outperform other methods by a large margin with better initialization and faster convergence after unlearning<sup>3</sup>. These results highlight the effectiveness of BMT and MMT, especially the advantage of sub-FL models in MMT for faster recovery after unlearning. Therefore, it is infeasible for large-scale experiments like CIFAR-100.

## 4.3 CONTINUAL LEARNING AND UNLEARNING

This experiment aims to simulate a continual setting in which new clients can join, and existing clients can leave the federation at any time during training. We will consider three settings corresponding to varying learning and unlearning order: 1) **2U1N**: Unlearn - New client - Unlearn; 2) **2U2N**: Unlearn - New client - Unlearn - New client; 3) **3U2N**: Unlearn - New client - Unlearn - New client - Unlearn. For a fixed number of communication rounds  $k$ , a new client will be introduced at round  $k/2$ , and an existing client will leave after every  $k$  round. We use the same  $k$  as the previous experiment. Fig. 5 shows performance in the continual learning and unlearning setting on the MNIST dataset. As can be seen, both BMT and MMT can seamlessly accommodate new clients and demonstrate general frameworks that can learn and unlearn rapidly.

<sup>3</sup>We did not include Exact-Fun baseline for CIFAR-100, as it has enormous GPU memory requirement to save all the intermediate model checkpoints, which is not feasible even with a H100 80GB GPU.

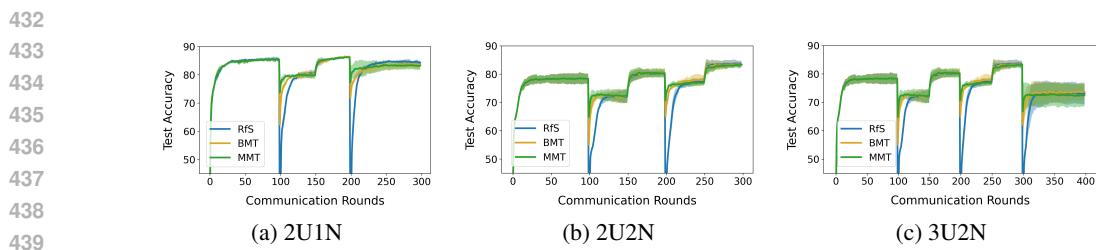


Figure 5: Test accuracy in the continual learning and unlearning setting on MNIST.

## 4.4 ABLATION STUDIES

444 **Data heterogeneity.** As mentioned earlier, the data heterogeneity ratio  $\rho$  defines the ratio between the  
445 number of samples in the majority and minority classes within a client’s dataset.  $\rho = 1$  indicates an  
446 IID dataset while  $\rho \approx 0$  indicates an extremely non-IID dataset. As shown in Fig. 6, MMT consistently  
447 obtains the best performance across different heterogeneity ratios. The gap to other methods is more  
448 pronounced with lower  $\rho$ , suggesting MMT is more favorable when we have extremely non-IID data.

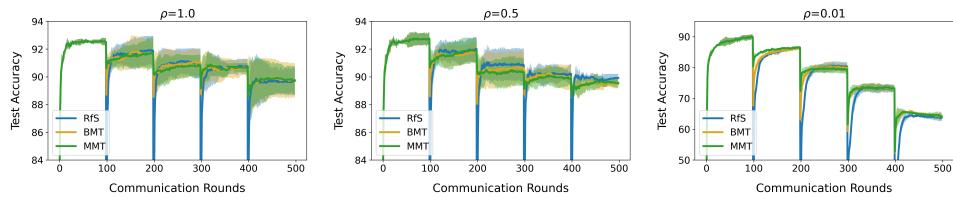
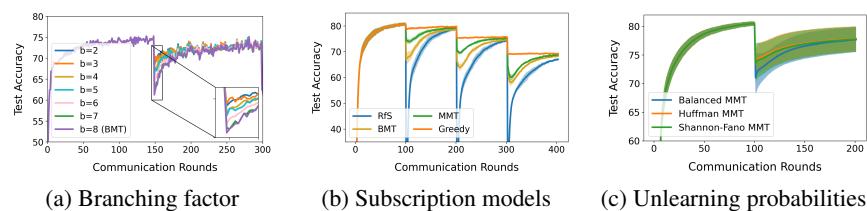


Figure 6: The effect of data heterogeneity on sequential unlearning performance on MNIST.

456 **Branching factor in MMT structure.** Recall that the default MMT uses a binary structure, i.e.,  
457 a branching factor of  $b = 2$  at each node in the tree of sub-FL modes. Therefore, we conduct  
458 experiments to analyze the impact of varying branching factors in MMT structure on the model  
459 performance. Particularly, for a federation consisting of  $n$  clients, the branching factor can range  
460 from 1 to  $n$  where  $b = 1$  indicates a traditional setting with no sub-FL models while  $b = n$  coincides  
461 with BMT, in which an auxiliary local model is maintained for each client. It is worth noting that we  
462 do not compare with  $b = 1$  because it is equivalent to RfS.  
463

Figure 7: **Left:** The effect of branching factor  $b$  in MMT structure. **Middle:** Performance of greedily constructed MMT given fixed unlearning order. **Right:** Performance of various tree construction methods given non-uniform unlearning probabilities.

472 As can be seen in Fig. 7a, a smaller branching factor generally results in higher test accuracy. This  
473 improvement occurs because MMT with a smaller branching factor aggregates fewer sub-FL models  
474 during unlearning. Furthermore, each sub-FL model is more likely to converge to the global optimum  
475 due to training on more local datasets. Thus, MMT with the default binary structure is the most suitable  
476 configuration for unlearning.

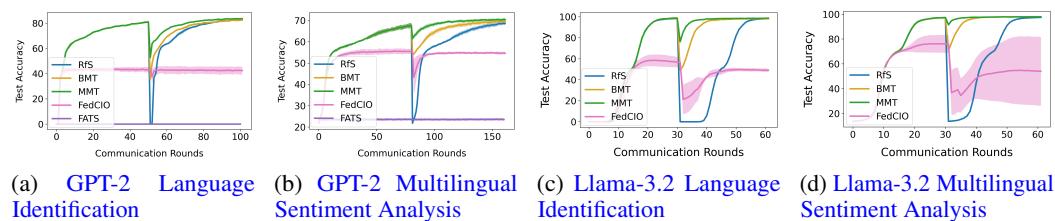
477 **Subscription models.** The unlearning order can be fixed in certain circumstances, e.g. clients may  
478 subscribe to the service that allows them to participate in the federated process for a fixed duration  
479 and will leave once their subscription expires. In such scenarios, it is possible to construct an optimal  
480 MMT structure that maximizes learning performance when clients leave the federation in a fixed order.  
481 Particularly, the optimal structure is the one that arranges the clients by their expiration date, with the  
482 soon-to-expire client at the top and greedily building the tree until reaching those with the farthest  
483 expiration. As observed in Fig. 7b, the greedy implementation of MMT achieves the best unlearning

486 performance and outperforms the default MMT that assumes uniform probabilities of unlearning for  
 487 all clients. Therefore, the greedy structure is preferable if the unlearning order of the clients is known  
 488 in advance, e.g., in subscription models.

489 **Non-uniform unlearning probabilities.** The default MMT implementation assumes uniform  
 490 unlearning probabilities for all clients. However, it is practical to consider the case where these  
 491 probabilities are non-uniform. In fact, we will demonstrate that given the unlearning probabilities  
 492 of each client, it is possible to construct improved MMT structures that achieve better unlearning  
 493 performance through two strategies based on Shannon-Fano coding (Shannon, 1948) and Huffman  
 494 coding (Huffman, 1952). Fig. 7c shows the performance for different tree construction methods. This  
 495 result is obtained by sampling the client to be removed for 100 times according to pre-defined  
 496 non-uniform unlearning probabilities of all clients. MMT structures that follow Shannon-Fano  
 497 coding and Huffman coding obtain visibly improved results over the default MMT. Furthermore,  
 498 Huffman-MMT obtains slightly better results than the Shannon-Fano counterpart, which aligns with  
 499 the classical information theory results that Huffman coding is more optimal than Shannon-Fano  
 500 coding for prefix-free code (Thomas & Joy, 2006).

#### 501 4.5 LANGUAGE TASKS

502 We also compare the performances of the proposed methods on two language tasks: 1) language  
 503 identification, where the goal is to detect the language of the given text (Conneau, 2019), and 2)  
 504 multilingual sentiment analysis, where the goal is to identify the sentiment of the given text using.  
 505 We use the Huggingface papluca/language-identification dataset for the former task and Huggingface  
 506 tyqiangz/multilingual-sentiments for the latter. We then randomly sample 200 and 500 data points  
 507 separately for each class from top-8 classes with the most data. For both datasets, we finetune a  
 508 pretrained GPT-2 Radford et al. (2019) model with the 200 data points set and Llama-3.2-3B model  
 509 with the 500 data points set<sup>4</sup> to predict which language the input sequences belong to, with next-token  
 510 prediction as the objective of getting the correct label. Fig 8 shows performance in unlearning settings  
 511 for different NLP tasks. In all cases, MMT improves the fastest after unlearning, followed by BMT.  
 512 In particular, for the larger model, both methods significantly outperformed other baselines, which  
 513 corroborates our methods’ scalability.<sup>5</sup> This experiment validates our method is effective across  
 514 different modalities and model architectures.



523 Figure 8: Sequential unlearning setting on two language tasks for GPT-2 and Llama-3.2-3B.  
 524

## 525 5 CONCLUSION

526 In this work, we propose two methods, BMT and MMT, for exact federated unlearning in the ongoing  
 527 federated learning collaboration. Our methods ensure the complete removal of an unlearned client’s  
 528 data while having better performance post-unlearning with the remaining clients than retraining  
 529 from scratch. Our methods are particularly useful in practical scenarios where model updation  
 530 in collaborative environments cannot afford long delays, with minimal tolerance for interruptions.  
 531 Our extensive experimental results demonstrate the effectiveness of the proposed methods. A few  
 532 interesting future research directions include proposing a principal approach to design an influence  
 533 tree under a resource constraint (i.e., the number of sub-FL models that can be trained is limited) and  
 534 how to change the influence tree post-unlearning or after a client joins the collaboration while having  
 535 the lowest IDS value and maximizing the use of the existing trained sub-FL models.

536 <sup>4</sup>Due to the high computation and memory requirements to train multiple LLMs, we opted for GPT-2 and  
 537 Llama-3.2-3B instead of the more popular and larger LLMs to show the performance of BMT and MMT.

538 <sup>5</sup>We did not include Exact-Fun for both LLM models and FATS for LLama-3.2-3B, as both methods have poor  
 539 scalability w.r.t. model size due to their GPU memory requirement to save all intermediate model checkpoints.

540 REFERENCES  
541

- 542 Mohammed Aledhari, Rehma Razzak, Reza M. Parizi, and Fahad Saeed. Federated Learning: A  
543 Survey on Enabling Technologies, Protocols, and Applications. *IEEE Access*, pp. 140699–140725,  
544 2020.
- 545 Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers,  
546 Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proc. IEEE SSP*, pp.  
547 141–159, 2021.
- 548 Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *Proc. ICML*, pp.  
549 1092–1104, 2021.
- 550 Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine  
551 learning. In *Proc. NeurIPS*, pp. 409–415, 2000.
- 552 CCPA. California consumer privacy act of 2018, 2018. California Civil Code Title 1.81.5.
- 553 A Conneau. Unsupervised cross-lingual representation learning at scale. *arXiv:1911.02116*, 2019.
- 554 Pallavi Dhade and Prajakta Shirke. Federated learning for healthcare: A comprehensive review.  
555 *Engineering Proceedings*, pp. 230, 2024.
- 556 Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to  
557 byzantine-robust federated learning. In *Proc. USENIX Security*, pp. 1605–1622, 2020.
- 558 GDPR. General data protection regulation, article 17: Right to erasure ('right to be forgotten').  
559 *Official Journal of the European Union*, 2016. Regulation (EU) 2016/679.
- 560 Antonio Ginart, Melody Guan, Gregory Valiant, and James Y. Zou. Making AI forget you: Data  
561 deletion in machine learning. In *Proc. NeurIPS*, pp. 3518–3531, 2019.
- 562 Jinu Gong, Osvaldo Simeone, and Joonhyuk Kang. Bayesian Variational Federated Learning and  
563 Unlearning in Decentralized Networks. *arXiv:2104.03834*, 2021.
- 564 Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal  
565 from machine learning models. In *Proc. ICML*, pp. 3832–3842, 2020.
- 566 David A Huffman. A method for the construction of minimum-redundancy codes. *IRE*, pp. 1098–1101,  
567 1952.
- 568 Meryem Janati Idrissi, Ismail Berrada, and Guevara Noubir. Fedbs: Learning on non-iid data in  
569 federated learning using batch normalization. In *Proc. IEEE ICTAI*, pp. 861–867, 2021.
- 570 Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and  
571 Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *Proc.  
572 ICML*, pp. 5132–5143, 2020.
- 573 Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.
- 574 Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to  
575 document recognition. *Proc. IEEE*, pp. 2278–2324, 1998.
- 576 Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning  
577 on non-iid features via local batch normalization. *arXiv:2102.07623*, 2021.
- 578 Yuzheng Li, Chuan Chen, Nan Liu, Huawei Huang, Zibin Zheng, and Qiang Yan. A blockchain-based  
579 decentralized federated learning framework with committee consensus. *IEEE Network*, pp.  
580 234–241, 2020.
- 581 Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federated Unlearning.  
582 *arXiv:2012.13891*, 2020.
- 583 Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, Xingliang Yuan, and Xiaoning Liu.  
584 A survey on federated unlearning: Challenges, methods, and future directions. *ACM Computing  
585 Surveys*, 2023.

- 594 Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In  
 595 *Federated learning: privacy and incentive*, pp. 240–254. Springer, 2020.  
 596
- 597 Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas.  
 598 Communication-efficient learning of deep networks from decentralized data. In *Proc. AISTATS*,  
 599 pp. 1273–1282, 2017.
- 600 Xutong Mu, Yulong Shen, Ke Cheng, Xueli Geng, Jiaxuan Fu, Tao Zhang, and Zhiwei Zhang.  
 601 Fedproc: Prototypical contrastive federated learning on non-iid data. *Future Generation Computer  
 602 Systems*, pp. 93–104, 2023.
- 603 Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning.  
 604 *Proc. NeurIPS*, pp. 16025–16036, 2020.
- 605 Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and  
 606 Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv:2209.02299*, 2022.
- 607
- 608 Prayitno, Chi-Ren Shyu, Karisma Trinanda Putra, Hsing-Chung Chen, Yuan-Yu Tsai, KSM Tozammel  
 609 Hossain, Wei Jiang, and Zon-Yin Shae. A systematic review of federated learning in the healthcare  
 610 area: From the perspective of data properties and applications. *Applied Sciences*, pp. 11191, 2021.
- 611 Hongyu Qiu, Yongwei Wang, Yonghui Xu, Lizhen Cui, and Zhiqi Shen. Fedcio: Efficient exact  
 612 federated unlearning with clustering, isolation, and one-shot aggregation. In *Proc. IEEE BigData*,  
 613 pp. 5559–5568, 2023.
- 614 Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language  
 615 models are unsupervised multitask learners. *OpenAI Blog*, 2019.
- 616
- 617 Claude Shannon. A mathematical theory of communication. *The Bell system technical journal*, pp.  
 618 379–423, 1948.
- 619 Nir Shlezinger, Mingzhe Chen, Yonina C Eldar, H Vincent Poor, and Shuguang Cui. Uveqfed:  
 620 Universal vector quantization for federated learning. *IEEE Trans. Signal Process*, pp. 500–514,  
 621 2020.
- 622 Karen Simonyan. Very deep convolutional networks for large-scale image recognition.  
 623 *arXiv:1409.1556*, 2014.
- 624
- 625 Youming Tao, Cheng-Long Wang, Miao Pan, Dongxiao Yu, Xiuzhen Cheng, and Di Wang.  
 626 Communication efficient and provable federated unlearning. *arXiv:2401.11018*, 2024.
- 627 MTCAJ Thomas and A Thomas Joy. *Elements of information theory*. Wiley-Interscience, 2006.
- 628
- 629 Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Federated Unlearning via Class-Discriminative  
 630 Pruning. *arXiv:2110.11794*, 2021.
- 631 Weiqi Wang, Zhiyi Tian, and Shui Yu. Machine unlearning: A comprehensive survey.  
 632 *arXiv:2405.07406*, 2024.
- 633 Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking  
 634 machine learning algorithms. *arXiv:1708.07747*, 2017.
- 635
- 636 Zuobin Xiong, Wei Li, Yingshu Li, and Zhipeng Cai. Exact-fun: An exact and efficient federated  
 637 unlearning approach. In *Proc. IEEE ICDM*, pp. 1439–1444, 2023.
- 638 Xiaohang Xu, Hao Peng, Md Zakirul Alam Bhuiyan, Zhifeng Hao, Lianzhong Liu, Lichao Sun, and  
 639 Lifang He. Privacy-preserving federated depression detection from multisource mobile health data.  
 640 *IEEE TII*, pp. 4788–4797, 2021.
- 641 Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and  
 642 applications. *ACM TIST*, pp. 1–19, 2019.
- 643
- 644 Yan Zeng, Yuankai Mu, Junfeng Yuan, Siyuan Teng, Jilin Zhang, Jian Wan, Yongjian Ren, and  
 645 Yunquan Zhang. Adaptive federated learning with non-iid data. *The Computer Journal*, pp.  
 646 2758–2772, 2023.
- 647 Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning.  
 648 *Knowledge-Based Systems*, pp. 106775, 2021.

648 A PROOF OF THEOREM 1  
649

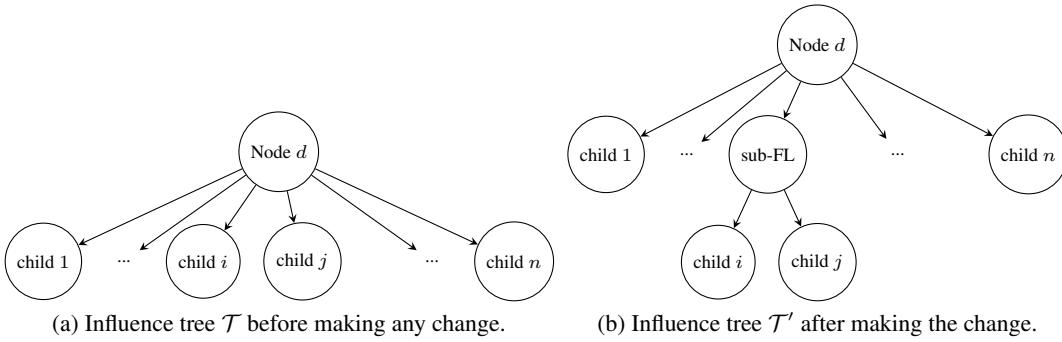
650 *Proof.* We first define the  $k$ -split influence node, which is a node in an influence tree with  $k > 2$  leaf  
651 nodes. We first consider the influence tree  $\mathcal{T}$ , where  $k$ -split influence node only has leaf nodes as  
652 children. We denote this node as  $d$ , and the set of its leaf nodes is denoted by  $\mathcal{C}$ . We now follow the  
653 following procedure. First, we remove the edge between the node  $d$  and any two of its leaf nodes  
654 (siblings), denoted by  $i$  and  $j$ . We create a sub-FL model with these two removed nodes and then add  
655 the node for this sub-FL model as a child to the node  $d$ , as shown in Fig. 9. We denote the resulting  
656 tree as  $\mathcal{T}'$ . Let  $f(\mathcal{T}, c)$  represent the number of sub-FL and local models that are aggregated to get  
657 the initial global modal after unlearning client  $c$  in the influence tree  $\mathcal{T}$ .


Figure 9: Changes in influence tree structure.

668 Note that  $f(\mathcal{T}', c) = f(\mathcal{T}, c) - 1$  for  $c \in \mathcal{C} \setminus \{i, j\}$  as one less leaf node to aggregate due to sub-FL  
669 model for  $\{i, j\}$  leaf nodes, and  $f(\mathcal{T}', c) = f(\mathcal{T}, c)$  for  $c \in \{i, j\}$  as sub-FL model is no longer  
670 useful due to influence of leaf node  $i$  or  $j$ . With this, we have following IDS due to the node  $d$ :  
671

$$\begin{aligned}
672 \quad s_d(\mathcal{T}) &= \sum_{c \in \mathcal{C}} p_c f(\mathcal{T}, c) = \sum_{c \in \mathcal{C} \setminus \{i, j\}} p_c (f(\mathcal{T}', c) + 1) + \sum_{c \in \{i, j\}} p_c (f(\mathcal{T}', c)) \\
673 &= k - 2 + \sum_{c \in \mathcal{C} \setminus \{i, j\}} p_c f(\mathcal{T}', c) + \sum_{c \in \{i, j\}} p_c f(\mathcal{T}', c) \quad (\text{node } d \text{ had } k \text{ leaf nodes}) \\
674 &= k - 2 + \sum_{c \in \mathcal{C}} p_c f(\mathcal{T}', c) \\
675 &= s_d(\mathcal{T}') \\
676 \implies s_d(\mathcal{T}) &> s_d(\mathcal{T}'). \quad (\text{as } k > 2)
\end{aligned} \tag{2}$$

677 Iteratively apply the same procedure on the rest of the child nodes until every node only has two  
678 children. After this, we obtain a binary tree. Since each operation strictly reduces IDS,  $s_d(\mathcal{T}_2) <$   
679  $s_d(\mathcal{T})$ . When the original tree already has some child nodes  $L$  that already belong to a binary subtree  
680  $\mathcal{T}_L$ , we treat this subtree as a single child node  $c'_k$  and apply the aforementioned operations on the  
681 child nodes that do not yet belong to a binary subtree. If all the child nodes belong to some binary  
682 subtree, we check from bottom-up to find the largest binary subtrees and treat them as a single  
683 child node to apply the aforementioned operations. Following this procedure, we can transform any  
684 arbitrary tree into a binary tree. In general,

$$\begin{aligned}
685 \quad f(\mathcal{T}', l) &= f(\mathcal{T}_L, l) + f(\mathcal{T}', c') = f(\mathcal{T}_L, l) + f(\mathcal{T}, c') - 1 = f(\mathcal{T}, l) - 1 \\
686 \quad \text{for } c' \in \mathcal{C} \setminus \{i, j\}, l \in L \text{ and} \\
687 \quad f(\mathcal{T}', l) &= f(\mathcal{T}_L, l) + f(\mathcal{T}', c') = f(\mathcal{T}_L, l) + f(\mathcal{T}, c') = f(\mathcal{T}, l)
\end{aligned}$$

688 for  $c' \in \{i, j\}$ ,  $l \in L$ . Notice that  $f(\mathcal{T}', l) = f(\mathcal{T}_L, l) + f(\mathcal{T}', c')$  always holds. Therefore, the  
689 inequality in Eq. (2) generalizes for any tree structure with the generalized operation. After applying  
690 this procedure on any arbitrary tree  $\mathcal{T}$  with at least one  $k$ -split influence node, the resulting binary  
691 tree  $\mathcal{T}_2$  always has a strictly smaller value of IDS, i.e.,  $s(\mathcal{T}_2) < s(\mathcal{T})$ .

Now we will proof the second part of theorem. Assume  $N$  is the number of non-root nodes,  $q_d$  is the probability of reaching a non-root node  $d$  starting from the root node, and  $N_d^s$  is the number of siblings of a non-root node  $i$ . Note that for a node  $d$ ,  $q_d = \sum_{c \in \mathcal{C}_d} p_c$  where  $\mathcal{C}_d$  is the set of all the client nodes (i.e., leaf nodes) that are descendants of node  $d$  and  $p_c$  is the probability of unlearning of the  $c$ -th descendant. Given an influence binary tree  $\mathcal{T}_2$  having  $n$  clients with known unlearning probability of each client, the IDS is given as follows:

$$s(\mathcal{T}) = \sum_{c=1}^n p_c f(S, c) = \sum_{d=1}^N q_d * N_d^s. \quad (3)$$

For  $\sum_{c=1}^n p_c f(S, c)$ , we can group all leaf nodes that share some common ancestor node  $d$  into a collection, with  $\mathcal{C}_d$  denoting this set. Since the same node has the same  $N_d^s$ , we can sum  $p_c$  of all such leaf nodes and rewrite  $\sum_{c=1}^n p_c f(S, c)$  as  $\sum_{d=1}^N \sum_{c \in \mathcal{C}_d} p_c * N_d^s = \sum_{d=1}^N q_d * N_d^s$ . Since each node of the binary tree has only one sibling, we have

$$\sum_{d=1}^N q_d * N_d^s = \sum_{d=1}^N q_d * 1 = \sum_{d=1}^N \sum_{c \in \mathcal{C}_d} p_c = \sum_{c=1}^n p_c * l_c \quad (4)$$

where  $n$  is the number of leaf nodes,  $l_c$  is the depth of the  $c$ -th leaf node, and  $p_c$  is the probability of reaching a non-root node  $c$  starting from the root node. As splitting the  $q_d$  of each non-leaf node into  $\mathcal{C}_d = \{p_1, p_2, \dots, p_r\}$ , where  $q_d = \sum_{c \in \mathcal{C}_d} p_c$  and each element in  $\mathcal{C}_d$  corresponds to a  $p_c$  of a leaf node, which is a descendant of that non-leaf node.

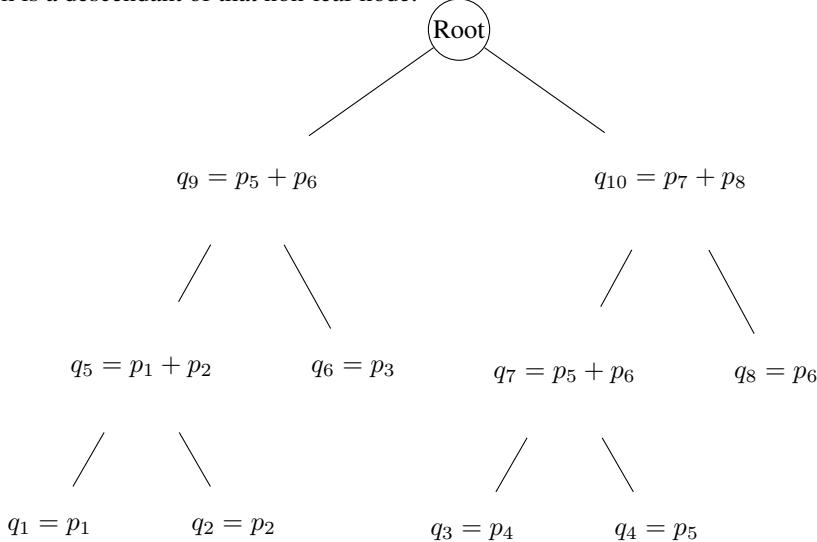


Figure 10: Visualization for  $\sum_{d=1}^N q_d * 1 = \sum_{c=1}^n p_c * l_c$ . One can easily see the equality holds.

Therefore, we can write  $\sum_{d=1}^N q_d * 1$  as the sum of  $p_c * l_c$  of all possible branches that reach each leaf node, as all the ancestor nodes of all leaf nodes have exactly one sibling (refer to Fig 10 for intuition). Finally, notice that  $\sum_{c=1}^n p_c * l_c$  is the expected code word length, if the same binary tree is used to represent a binary prefix-free code encoding scheme. Since Huffman coding is optimal for minimizing  $\sum_{c=1}^n p_c * l_c$ ,  $s(\mathcal{T}_{\text{Huffman}}) \leq s(\mathcal{T}_2)$  where  $\mathcal{T}_{\text{Huffman}}$  is an influence tree constructed following Huffman coding and  $\mathcal{T}_2$  is any binary influence trees for the same set of  $p_c$ .  $\square$

## B ADDITIONAL EXPERIMENTAL RESULTS

**Benchmarking against SISA.** SISA is a model-agnostic exact unlearning method proposed in Bourtoule et al. (2021). It involves partitioning the training dataset into disjoint subsets and training isolated models on each subset, whose predictions are aggregated during inference time. In our

context, we can train isolated models on each client’s data and remove only the influenced model when a client leaves the collaboration to achieve exact unlearning. Fig. 11a shows SISA performance in the sequential unlearning setting on MNIST. Even though SISA obtains a better initialization than RfS, it incurs the worst post-unlearning performance compared to FL methods. This result suggests that the SISA training paradigm may not be well-suited for collaborative training among heterogeneous clients with limited data. Therefore, it serves as a less competitive exact FU benchmark.

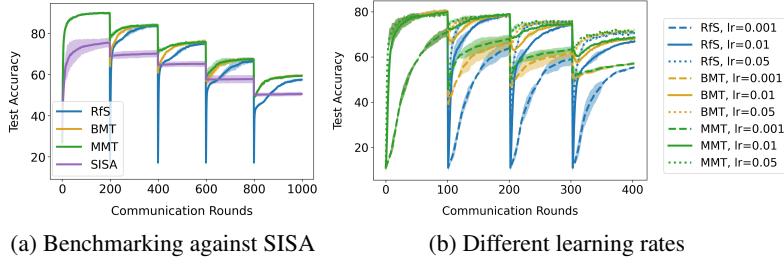


Figure 11: **Left:** Sequential unlearning benchmark against SISA. **Right:** Performance for different learning rates in the sequential unlearning setting.

**Varying learning rates.** Fig. 11b shows performance for three learning rates {0.001, 0.01, 0.05} in the sequential unlearning setting on the MNIST dataset. In all cases, MMT converges the fastest, followed by BMT and RfS. This result validates the effectiveness of BMT and MMT compared to RfS across varying learning rates.

**Converged performance of CNN on CIFAR-100.** We increased the communication rounds from 500 to 1000 to analyze the converged performance of CNN trained on CIFAR-100. As seen in Fig. 12, there is a significant train-test accuracy gap for BMT and MMT starting from the 200th round after each unlearning request, indicating overfitting has occurred in both methods. This difference is due to using simple models like a 2-layer CNN when training for a long period. Therefore, we have included CIFAR-100 results on VGG-16 in Fig. 4(d).

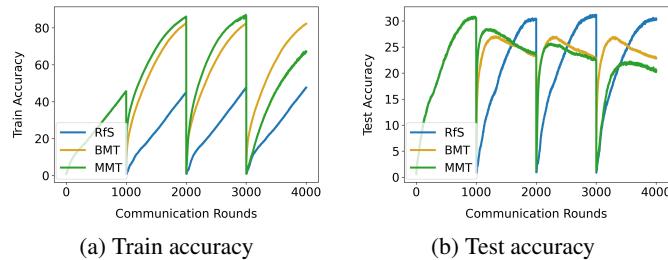


Figure 12: Performance at convergence on CIFAR-100 using a 2-layer CNN.

**Performance on unequal data.** In this experiment, we split the original dataset consisting of  $n$  classes into  $n$  clients with unequal data sizes. Specifically, each client transfers a portion  $p \sim \mathcal{U}(0, 0.9)$  of their training data to another random client. We set the upper bound to 0.9 to prevent clients with empty data. As shown in Fig. 13, MMT and BMT consistently outperform RfS across all experiments, regardless of clients’ data sizes.

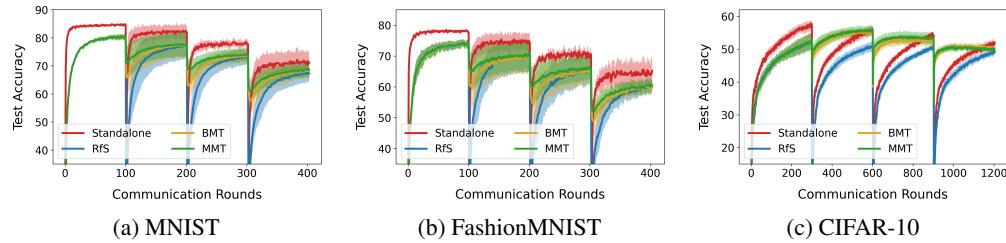


Figure 13: Performance on MNIST, FashionMNIST and CIFAR-10 with unequal clients’ data.