

1277 B SUPPLEMENTARY

1278 B.1 Hyperparamters

1280 We list the hyperparameters used for each baseline and our model
1281 as follows:

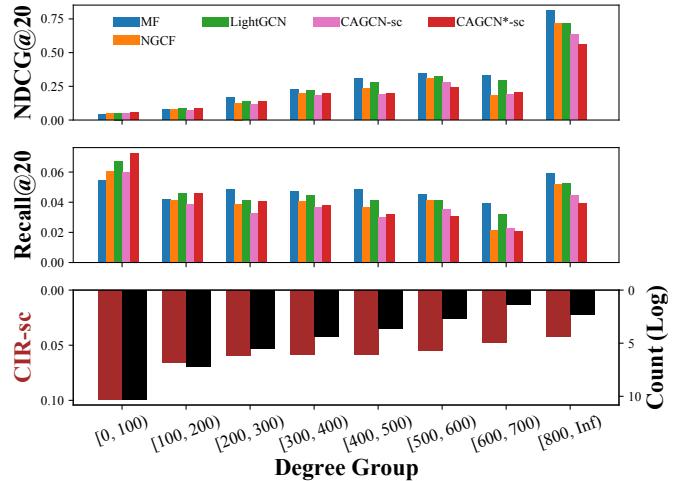
- 1282 • **LightGCN.** Propagation layers: $L = 3$; Pooling layer: Meaning
1283 pooling;
- 1284 • **NGCF.** Propagation layers: $L = 3$; Slope of LeakyRelu: 0.2; Pool-
1285 ing layer: Concatenation
- 1286 • **UltraGCN.** For Gowalla, Yelp, Amazon and Ml-1M, we use ex-
1287 actly the same hyperparameter configurations provided here. For
1288 Loseit and News, the hyperparamters are as follows:
1289 (1) Loseit: Training epochs 2000; Learning rate $1e^{-3}$; batch size
1290 512; Loss weights $w_1 = 1e^{-6}, w_2 = 1, w_3 = 1e^{-6}, w_4 = 1$; the
1291 number of negative samples per epoch per positive pair 20;
1292 negative weight 20; weight of l_2 regularization $\gamma = 1e^{-4}$, 2nd-
1293 constraining loss coefficient $\lambda = 5e^{-4}$.
- 1294 (2) News: Training epochs 2000; Learning rate $1e^{-3}$; batch size
1295 1024; Loss weights $w_1 = 1e^{-8}, w_2 = 1, w_3 = 1, w_4 = 1e^{-8}$; the
1296 number of negative samples per epoch per positive pair 1000;
1297 negative weight 200; weight of l_2 regularization $\gamma = 1e^{-4}$, 2nd-
1298 constraining loss coefficient $\lambda = 5e^{-4}$.
- 1299 • **GTN.** For Gowalla, Yelp, Amazon, we directly report the result
1300 provided here.

1301 In the following, we introduce the hyparamemeters we used for
1302 our CAGCN(*)-variants. With specification, the number of train-
1303 ing epochs is set to be 1000; the learning rate 0.001; l_2 regularization
1304 $1e^{-4}$; number of negative samples 1; embedding dimension
1305 64; batch size 256.

- 1306 • **CAGCN-jc.** (1) Gowalla: $\gamma = 1$; (2) Yelp: $\gamma = 1.2$; (3) Amazon:
1307 $\gamma = 1$; (4) Ml-1M: $\gamma = 2$; (5) Loseit: $\gamma = 1$; (6) News: $\gamma = 1$.
- 1308 • **CAGCN-cn.** (1) Gowalla: $\gamma = 1$; (2) Yelp: $\gamma = 1.2$; (3) Amazon:
1309 $\gamma = 1$; (4) Ml-1M: $\gamma = 1$; (5) Loseit: $\gamma = 1$; (6) News: $\gamma = 1$.
- 1310 • **CAGCN-sc.** (1) Gowalla: $\gamma = 1$; (2) Yelp: $\gamma = 1$; (3) Amazon:
1311 $\gamma = 1$; (4) Ml-1M: $\gamma = 2$; (5) Loseit: $\gamma = 1$; (6) News: $\gamma = 1$.
- 1312 • **CAGCN-lhn.** (1) Gowalla: $\gamma = 1.2$; (2) Yelp: $\gamma = 1$; (3) Amazon:
1313 $\gamma = 1$; (4) Ml-1M: $\gamma = 2$; (5) Loseit: $\gamma = 1, L = 1$; (6) News: $\gamma = 1.5$.
- 1314 • **CAGCN*-jc.** (1) Gowalla: $\gamma = 1.2, l_2$ -regularization $1e^{-3}$; (2)
1315 Yelp: $\gamma = 1.7, l_2$ -regularization $1e^{-3}$; (3) Amazon: $\gamma = 1.7, l_2$ -
1316 regularization $1e^{-3}$; (4) Ml-1M: $\gamma = 1, l_2$ -regularization $1e^{-3}$; (5)
1317 Loseit: $\gamma = 1, L = 2$; (6) News: $\gamma = 1, L = 2$.
- 1318 • **CAGCN*-sc.** (1) Gowalla: $\gamma = 1.2, l_2$ -regularization $1e^{-3}$; (2)
1319 Yelp: $\gamma = 1.7, l_2$ -regularization $1e^{-3}$; (3) Amazon: $\gamma = 1.7, l_2$ -
1320 regularization $1e^{-3}$; (4) Ml-1M: $\gamma = 1, l_2$ -regularization $1e^{-3}$; (5)
1321 Loseit: $\gamma = 1, L = 2$; (6) News: $\gamma = 1, L = 2$.
- 1322 • **CAGCN*-lnh.** (1) Gowalla: $\gamma = 1, l_2$ -regularization $1e^{-3}$; (2)
1323 Yelp: $\gamma = 1, l_2$ -regularization $1e^{-3}$; (3) Amazon: $\gamma = 1.5, l_2$ -
1324 regularization $1e^{-3}$; (4) Ml-1M: $\gamma = 1, l_2$ -regularization $1e^{-3}$; (5)
1325 Loseit: $\gamma = 0.5, L = 2$; (6) News: $\gamma = 1, L = 2$.

1326 B.2 Performance Interpretation

1327 To demonstrate the generality of our observation in Figure 7, we fur-
1328 ther perform exactly the same analysis on Yelp (shown in Figure 11)
1329 and derive almost the same insights: 1) Graph-based recom-
1330 mendation models achieve higher performance than non-graph-based
1331 ones for lower degree nodes; 2) the opposite performance trends be-
1332 tween NDCG and Recall indicates that different evaluation metrics
1333 have different levels of sensitivity to node degrees.



1335 Figure 11: Performance of model w.r.t. node degree on Yelp.

1336 B.3 Thorough complexity analysis

1337 Generally compared with the very basic MF, the main computa-
1338 tional issue of LightGCN comes from the message-passing which
1339 takes $O(L|\mathcal{E}|F)$ time and $O(L|\mathcal{V}|F)$ space to save the intermediate
1340 node representations. For NGCF, the extra complexity comes from
1341 the nonlinear transformation, which takes $O(L|\mathcal{V}|F^2)$ time and
1342 $O(LF^2)$ space to save the transformation weights. For UltraGCN,
1343 the main bottleneck comes from computing the user-user connec-
1344 tions, which involves the power of adjacency matrix and hence
1345 $O(|\mathcal{V}|^3)$. Furthermore, as it samples hundreds of negative samples
1346 and the optimization is also performed on the user-user connec-
1347 tions, then its time complexity would be $O(r(|\mathcal{E}| + |\mathcal{V}|K)F)$. For
1348 CAGCN, since we only consider 2-hops away connections to com-
1349 pute CIR in Eq. (5)(essentially for each center node, we count the
1350 number of paths of length 2 from each of its neighbors to its whole
1351 neighborhood), the main computational load would be computing
1352 the power of adjacency matrix, which takes $O(|\mathcal{V}|^3)$. Note that for
1353 both of our CAGCN and UltraGCN, we can apply Strassens's Algo-
1354 rithm to further reduce the $O(|\mathcal{V}|^3)$ to $O(|\mathcal{V}|^{2.8})$ for computing
1355 the power of adjacency matrix.

1356 B.4 Graph Isomorphism

1357 We review the concepts of subtree/subgraph-isomorphism [40].

1358 **Definition B.1. Subtree-isomorphism:** S_u and S_i are subtree-
1359 isomorphic, denoted as $S_u \cong_{\text{subtree}} S_i$, if there exists a bijective
1360 mapping $h: \tilde{\mathcal{N}}_u^1 \rightarrow \tilde{\mathcal{N}}_i^1$ such that $h(u) = i$ and $\forall v \in \tilde{\mathcal{N}}_u^1, h(v) = j, e_v^l = e_j^l$.

1361 **Definition B.2. Subgraph-isomorphism:** S_u and S_i are subgraph-
1362 isomorphic, denoted as $S_u \cong_{\text{subgraph}} S_i$, if there exists a bijective
1363 mapping $h: \tilde{\mathcal{N}}_u^1 \rightarrow \tilde{\mathcal{N}}_i^1$ such that $h(u) = i$ and $\forall v_1, v_2 \in \tilde{\mathcal{N}}_u^1, e_{v_1 v_2} \in$
1364 \mathcal{E}_{S_u} iff $e_{h(v_1)h(v_2)} \in \mathcal{E}_{S_i}$ and $e_{v_1}^l = e_{h(v_1)}^l, e_{v_2}^l = e_{h(v_2)}^l$.

1365 Corresponding to the backward(\Leftarrow) proof of Theorem 2, here
1366 we show two of such graphs S_u, S'_u , which are subgraph isomor-
1367 phic but non-bipartite-subgraph-isomorphic. Assuming u and u'
1368 have exactly the same neighborhood feature vectors e , then directly
1369

1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392

propagating according to 1-WL or even considering node degree as the edge weight as GCN [12] can still end up with the same propagated feature for u and u' . However, if we leverage JC to calculate CIR as introduced in Appendix A.1, then we would end up with $\{(d_{udj_1})^{-0.5} \mathbf{e}, (d_{udj_2})^{-0.5} \mathbf{e}, (d_{udj_3})^{-0.5} \mathbf{e}\} \neq \{(d_{u'}^{-0.5} d_{j'_1}^{-0.5} + \tilde{\Phi}_{u'j'_1}) \mathbf{e}, (d_{u'}^{-0.5} d_{j'_2}^{-0.5} + \tilde{\Phi}_{u'j'_2}) \mathbf{e}, (d_{u'}^{-0.5} d_{j'_3}^{-0.5} + \tilde{\Phi}_{u'j'_3}) \mathbf{e}\}$. Since g is injective by Lemma 1, CAGCN would yield two different embeddings for u and u' .

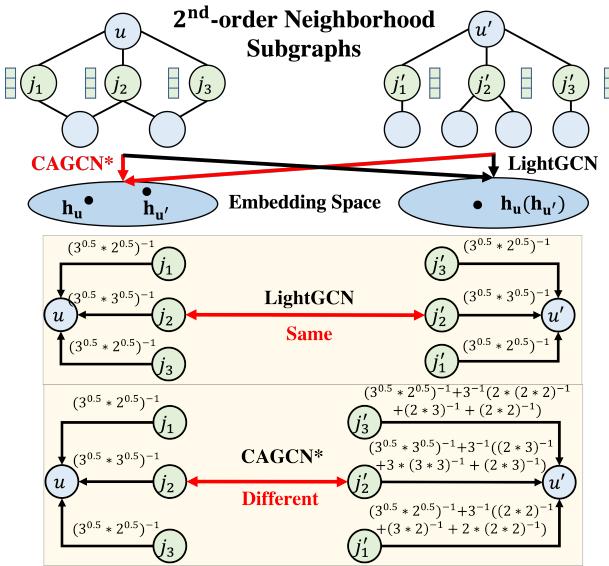


Figure 12: An example showing two neighborhood subgraph $S_u, S_{u'}$ that are subgraph-isomorphic but not bipartite-subgraph-isomorphic.

B.5 Efficiency Comparison

Here we use exactly the same setting introduced in Section 4.3 and keep track the performance/training time per 5 epochs for Gowalla, Yelp2018, ML-1M, and Loseit in Figure 13. Clearly, CAGCN* achieves extremely higher performance in significantly less time because the collaboration-aware graph convolution leverages more beneficial collaborations from neighborhoods. Specifically, in Figure 13(c), we observe the slower performance increase of CAGCN* and LightGCN on ML-1M. We ascribe this to the higher density of ML-1M as in Table 1 that leads to so much noisy neighboring information. One future direction could be to leverage the CIR to prune the graph of these noisy connections in an iterative fashion as either a preprocessing step or even used throughout training when paired with an attention mechanism (although the latter would come at a significantly longer training time).

1393	propagating according to 1-WL or even considering node degree	1451
1394	as the edge weight as GCN [12] can still end up with the same	1452
1395	propagated feature for u and u' . However, if we leverage JC to	1453
1396	calculate CIR as introduced in Appendix A.1, then we would end up	1454
1397	with $\{(d_{udj_1})^{-0.5} \mathbf{e}, (d_{udj_2})^{-0.5} \mathbf{e}, (d_{udj_3})^{-0.5} \mathbf{e}\} \neq \{(d_{u'}^{-0.5} d_{j'_1}^{-0.5} + \tilde{\Phi}_{u'j'_1}) \mathbf{e}, (d_{u'}^{-0.5} d_{j'_2}^{-0.5} + \tilde{\Phi}_{u'j'_2}) \mathbf{e}, (d_{u'}^{-0.5} d_{j'_3}^{-0.5} + \tilde{\Phi}_{u'j'_3}) \mathbf{e}\}$.	1455
1398	Since g is injective by Lemma 1, CAGCN would yield two different embeddings	1456
1399	for u and u' .	1457
1400		1458
1401		1459
1402		1460
1403	2nd-order Neighborhood Subgraphs	1461
1404		1462
1405		1463
1406		1464
1407		1465
1408	CAGCN* ↓	1466
1409		1467
1410		1468
1411	LightGCN	1469
1412		1470
1413	Same	1471
1414		1472
1415	CAGCN*	1473
1416		1474
1417	Different	1475
1418		1476
1419		1477
1420		1478
1421		1479
1422		1480
1423		1481
1424		1482
1425		1483
1426		1484
1427		1485
1428		1486
1429		1487
1430		1488
1431		1489
1432		1490
1433		1491
1434		1492
1435		1493
1436		1494
1437		1495
1438		1496
1439		1497
1440		1498
1441		1499
1442		1500
1443		1501
1444		1502
1445		1503
1446		1504
1447		1505
1448		1506
1449		1507
1450		1508

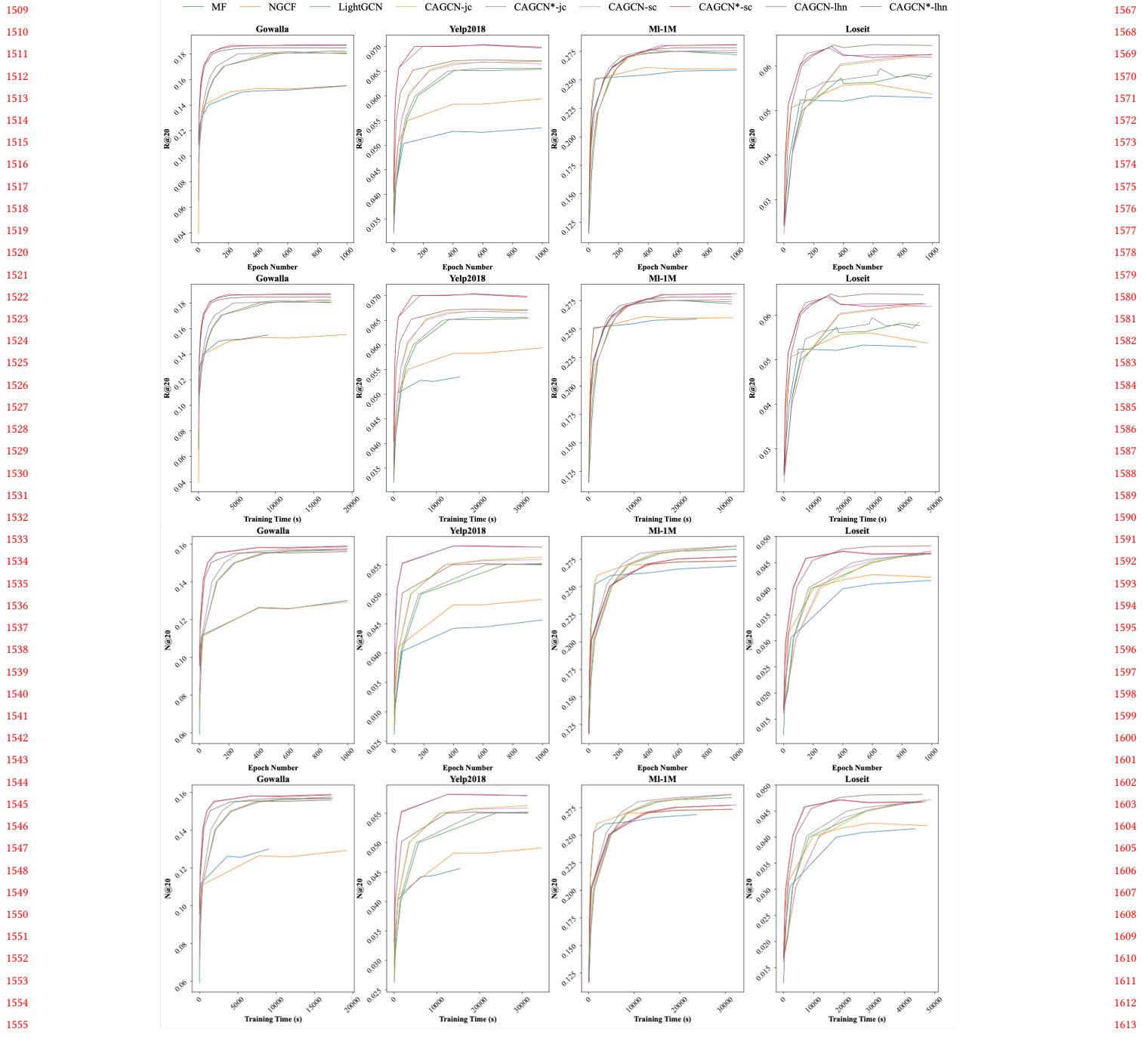


Figure 13: Comparing the training efficiency of each model under R@20 and N@20.