Project in “Deep Learning”

2024-25

Aslanidou Aikaterina-Sofia

aaslanidou@uth.gr

Electrical & Computer Engineering

University of Thessaly

Contents

Introduction

In this project, I explore the use of Neural Collaborative Filtering (NeuMF) for top-K item recommendation, following the methodology presented in the paper by Xiangnan He et al. Instead of the original dataset, I use the MovieLens u.data file to train and evaluate the model. I experiment with different configurations, such as varying the number of MLP layers, negative samples, and latent factors, and assess performance using HR@10 and NDCG@10. I also implement Non-negative Matrix Factorization (NMF) for comparison and apply three knowledge distillation techniques to compress the NeuMF model. All results are averaged over 10 runs, and the findings are presented through plots and tables, along with my observations.

1)Dataset splitting & parameter setting

(i) To reproduce the experiments of the Neural Collaborative Filtering paper, the u.data file from the MovieLens dataset was used, containing 100,000 ratings from 943 users on 1,682 items. The dataset was split using the leave-one-out strategy: for each user, the most recent interaction (based on timestamp) was selected as the test instance, while the remaining interactions were used for training. All ratings were converted to binary format (implicit feedback), and negative sampling was applied during evaluation by selecting 99 items the user had not interacted with. User and item IDs were converted from 1-based to 0-based indexing to ensure compatibility with PyTorch. The entire preprocessing pipeline was implemented in a custom script, generate_dataset.py, which produced the following files: data/ml-1m.train.rating (training set) and data/ml-1m.test.negative (test set with negative samples).

(ii) Parameter setting

I trained the model using the following hyperparameters:

- num_layers = 3: number of layers in the MLP part, chosen to match the full structure from the paper.
- factor_num : The optimal value will be selected based on performance results from Questions 2, 3, and 4.
- num_ng = 4: number of negative samples per positive instance during training, following the paper's strategy.
- test_num_ng = 99: 99 negative items per user in the test set for top-K evaluation.

- batch_size = 256: balanced choice for speed and training stability.
- learning_rate = 0.001, dropout = 0.0, epochs = 20, top_k = 10: default values based on the original paper.

To compare with NeuMF-pre, I first pretrained two separate models: GMF (with the same settings) and MLP (with num_layers = 1, 2, 3). Then, I loaded them into NeuMF-pre and trained it for 20 more epochs using the SGD optimizer, as described in the methodology.

2)Experiments with and without pre-training

(i)With pre-training

To study how the performance of NeuMF is affected by the number of MLP layers, I tested the model with 1, 2, and 3 layers, both with and without pretraining. I observed that when pretraining is used, the model achieves not only higher HR@10 scores but also much more consistent results across different runs. In other words, the performance doesn't fluctuate much—there's very little variation from run to run. This shows that pretraining helps the model start from better initial weights, which leads to more stable and effective learning.

On the other hand, without pretraining, the results are more sensitive to the number of MLP layers. In many cases, the HR@10 scores vary more between runs, and the model doesn't always benefit from deeper architectures. For example, with lower factor values like 8 or 16, I noticed that adding more layers sometimes leads to worse performance.

In conclusion, pretraining improves not only the overall HR@10 performance but also the training stability of NeuMF. Without it, the model behaves less predictably and deeper networks don't always lead to better results.

Figure 1 : 8 Factors

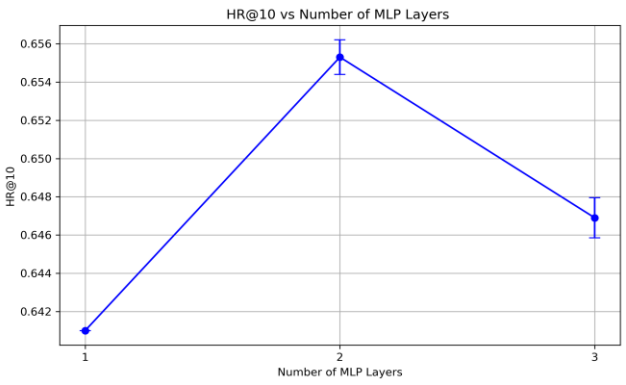


Figure 2 : 16 Factors

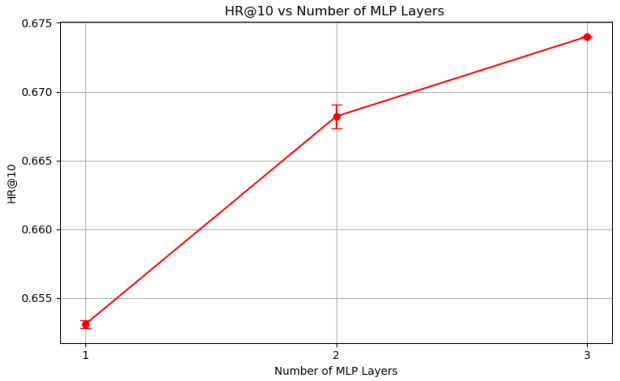


Figure 3 : 32 Factors

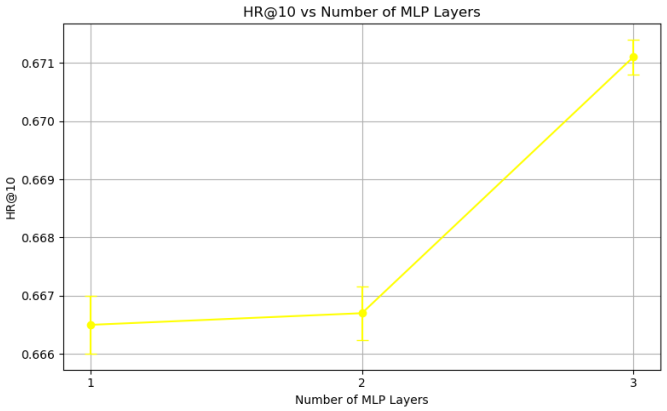


Figure 4 : 64 Factors

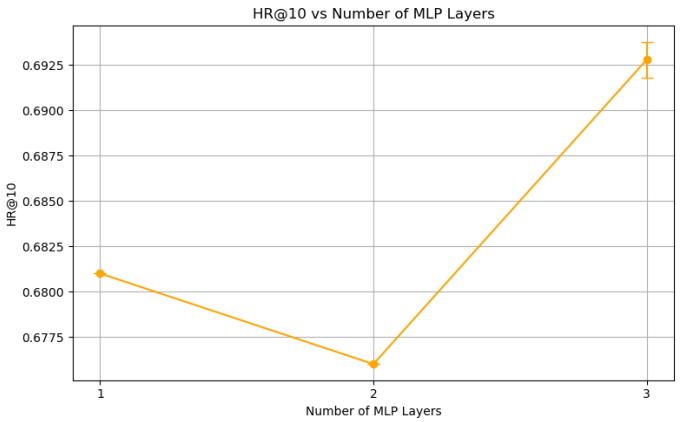
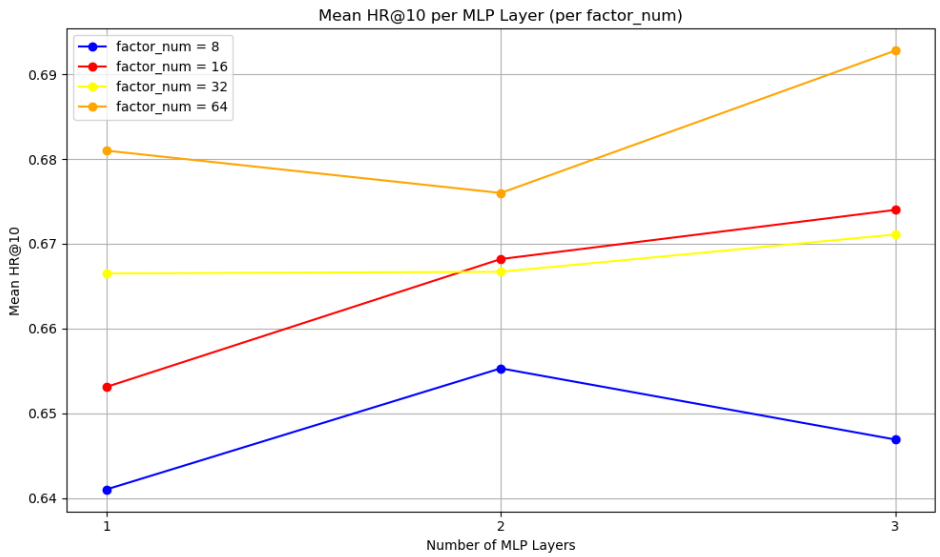


Figure 5 : Mean HR@10 per MLP Layer(per factor_num)



8 Number of factors

Table 1:MLP 1 Layer

Run	*HR@10*
1	0.641
2	0.641
3	0.641
4	0.641
5	0.641
6	0.641
7	0.641
8	0.641
9	0.641
10	0.641
Mean	**0.641**
Std	**0.000**

Table 2: MLP 2 Layer

Run	*HR@10*
1	0.656
2	0.656
3	0.655
4	0.656
5	0.654
6	0.655
7	0.655
8	0.655
9	0.657
10	0.654
Mean	**0.655**
Std	**0.001**

Table 3: MLP 3 Layer

Run	*HR@10*
1	0.647
2	0.647
3	0.647
4	0.647
5	0.648
6	0.647
7	0.647
8	0.648
9	0.644
10	0.647
Mean	**0.647**
Std	**0.001**

16 Number of actors

Table 4 :MLP 1 Layer

Run	*HR@10*
1	0.653
2	0.653
3	0.653
4	0.653
5	0.653
6	0.653
7	0.653
8	0.654
9	0.653
10	0.653
Mean	**0.653**
Std	**0.000**

Table 5 : MLP 2 Layer

Run	*HR@10*
1	0.669
2	0.668
3	0.669
4	0.669
5	0.669
6	0.667
7	0.667
8	0.669
9	0.667
10	0.668
Mean	**0.668**
Std	**0.001**

Table 6: MLP 3 Layer

Run	*HR@10*
1	0.674
2	0.674
3	0.674
4	0.674
5	0.674
6	0.674
7	0.674
8	0.674
9	0.674
10	0.674
Mean	**0.674**
Std	**0.000**

32 Number of Factors

Table 7 :MLP 1 Layer

Run	*HR@10*
1	0.667
2	0.666
3	0.666
4	0.666
5	0.667
6	0.666
7	0.667
8	0.666
9	0.667
10	0.667
Mean	**0.666**
Std	**0.001**

Table 8 : MLP 2 Layer

Run	*HR@10*
1	0.667
2	0.666
3	0.666
4	0.667
5	0.667
6	0.666
7	0.667
8	0.667
9	0.667
10	0.667
Mean	**0.667**
Std	**0.000**

Table 9: MLP 3 Layer

Run	*HR@10*
1	0.671
2	0.672
3	0.671
4	0.671
5	0.671
6	0.671
7	0.671
8	0.671
9	0.671
10	0.671
Mean	**0.671**
Std	**0.000**

64 Number of Factors

Table 10 :MLP 1 Layer

Run	*HR@10*
1	0.681
2	0.681
3	0.681
4	0.681
5	0.681
6	0.681
7	0.681
8	0.681
9	0.681
10	0.681
Mean	**0.681**
Std	**0.000**

Table 11 : MLP 2 Layer

Run	*HR@10*
1	0.676
2	0.676
3	0.676
4	0.676
5	0.676
6	0.676
7	0.676
8	0.676
9	0.676
10	0.676
Mean	**0.676**
Std	**0.000**

Table 12 : MLP 3 Layer

Run	*HR@10*
1	0.694
2	0.694
3	0.694
4	0.692
5	0.692
6	0.692
7	0.692
8	0.692
9	0.692
10	0.694
Mean	**0.693**
Std	**0.001**

(ii) Without pre-training

Figure 6: 8 factors

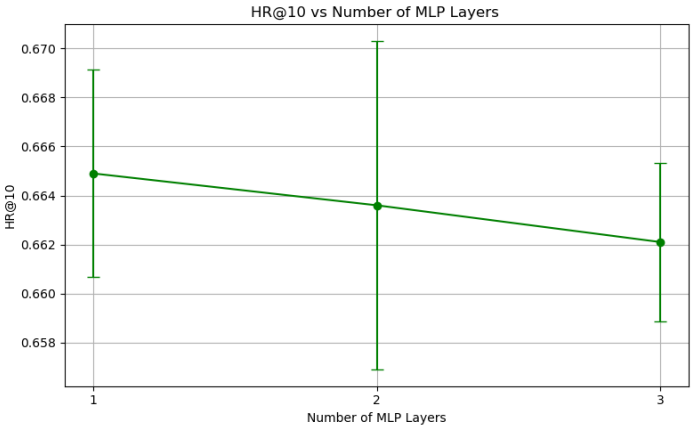


Figure 7 : 16 factors

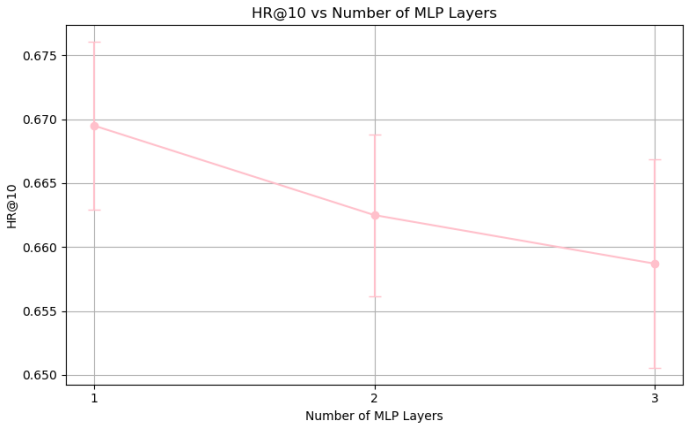


Figure 8 : 32 Factors

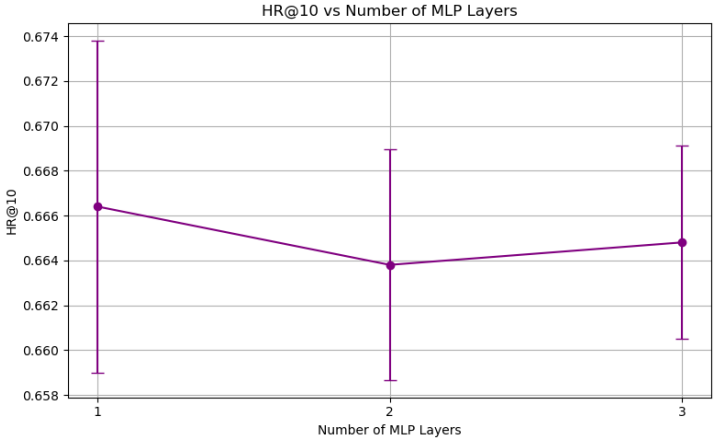


Figure 9 : 64 Factors

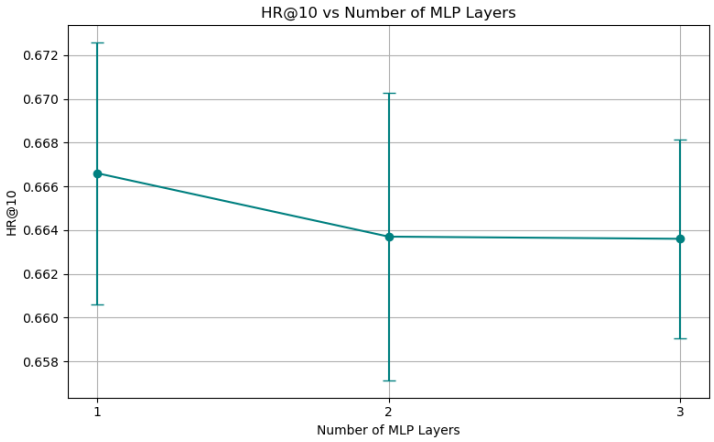
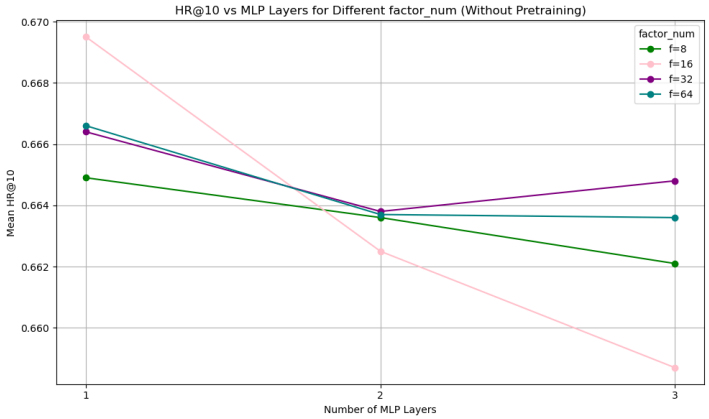


Figure 10 : All factors



8 factors

Table 13 :MLP 1 Layer

Run	*HR@10*
1	0.660
2	0.662
3	0.671
4	0.664
5	0.674
6	0.665
7	0.666
8	0.662
9	0.661
10	0.664
Mean	**0.665**
Std	**0.004**

Table 14 : MLP 2 Layer

Run	*HR@10*
1	0.671
2	0.659
3	0.652
4	0.669
5	0.657
6	0.669
7	0.659
8	0.671
9	0.659
10	0.670
Mean	**0.664**
Std	**0.007**

Table 15 : MLP 3 Layer

Run	*HR@10*
1	0.662
2	0.656
3	0.659
4	0.661
5	0.662
6	0.664
7	0.663
8	0.667
9	0.660
10	0.667
Mean	**0.662**
Std	**0.003**

16 factors

Table 16 :MLP 1 Layer

Run	*HR@10*
1	0.673
2	0.666
3	0.678
4	0.673
5	0.657
6	0.676
7	0.663
8	0.665
9	0.677
10	0.667
Mean	**0.669**
Std	**0.007**

Table 17 : MLP 2 Layer

Run	*HR@10*
1	0.668
2	0.656
3	0.672
4	0.655
5	0.672
6	0.664
7	0.659
8	0.664
9	0.661
10	0.654
Mean	**0.662**
Std	**0.006**

Table 18 : MLP 3 Layer

Run	*HR@10*
1	0.668
2	0.672
3	0.668
4	0.657
5	0.647
6	0.651
7	0.663
8	0.650
9	0.654
10	0.657
Mean	**0.659**
Std	**0.008**

32 factors

Table 19 :MLP 1 Layer

Run	*HR@10*
1	0.676
2	0.663
3	0.671
4	0.655
5	0.663
6	0.661
7	0.676
8	0.656
9	0.672
10	0.671
Mean	**0.666**
Std	**0.007**

Table 20 : MLP 2 Layer

Run	*HR@10*
1	0.661
2	0.663
3	0.670
4	0.659
5	0.664
6	0.660
7	0.659
8	0.676
9	0.661
10	0.665
Mean	**0.664**
Std	**0.005**

Table 21 : MLP 3 Layer

Run	*HR@10*
1	0.672
2	0.661
3	0.672
4	0.662
5	0.665
6	0.662
7	0.662
8	0.668
9	0.665
10	0.659
Mean	**0.665**
Std	**0.004**

64 factors

Table 22 :MLP 1 Layer

Run	*HR@10*
1	0.660
2	0.662
3	0.667
4	0.667
5	0.681
6	0.663
7	0.669
8	0.672
9	0.661
10	0.664
Mean	**0.667**
Std	**0.006**

Table 23 : MLP 2 Layer

Run	*HR@10*
1	0.672
2	0.671
3	0.666
4	0.659
5	0.657
6	0.663
7	0.672
8	0.661
9	0.651
10	0.665
Mean	**0.664**
Std	**0.007**

Table 24 : MLP 3 Layer

Run	*HR@10*
1	0.657
2	0.663
3	0.659
4	0.669
5	0.671
6	0.666
7	0.663
8	0.657
9	0.666
10	0.665
Mean	**0.664**
Std	**0.005**

3)Exploring How MLP Layers and Embedding Size Affect Model Parameters in NeuMF

To answer this question, I examined how the number of weight parameters in the NeuMF model without pretraining changes when adjusting the number of MLP layers from 1 to 3 and also varying the factor_num values. I wrote a script that calculates the total number of trainable parameters for each configuration. Since the number of parameters is completely determined by the model's architecture—such as the number of users, items, the embedding size (factor_num), and the number of layers—there's no randomness involved, so I didn't need to repeat experiments or calculate averages.

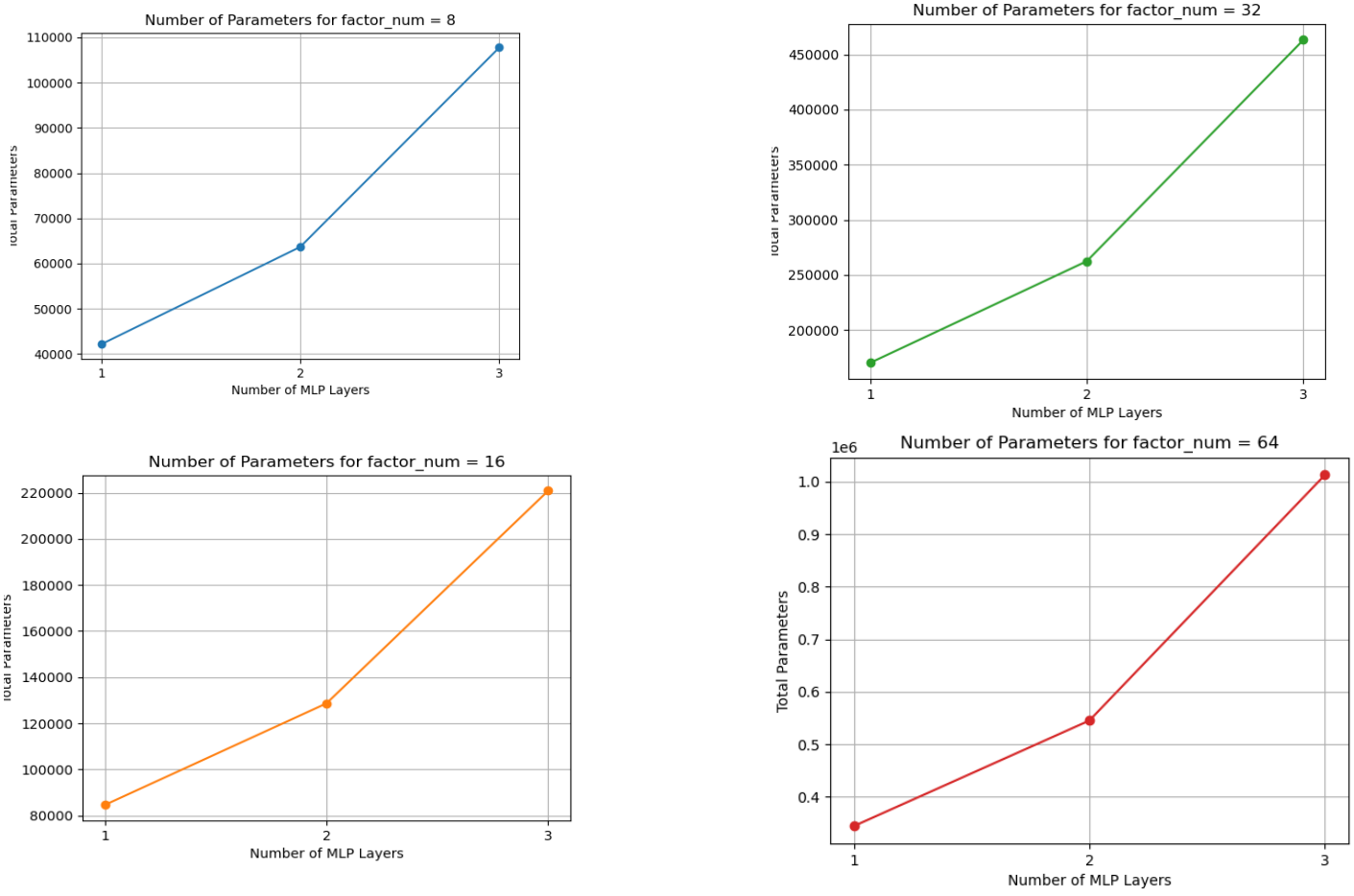
The results showed a clear pattern: increasing factor_num results in a linear increase in parameters because the embeddings for users and items grow in size. However, increasing the number of MLP layers causes a much sharper increase in parameters. That's because each MLP layer reduces the size of the previous layer by half, so adding more layers introduces several new weight matrices, each with many connections. For example, with factor_num = 64, the parameter count went from around 344K with 1 layer to over 1 million with 3 layers. This demonstrates how model complexity and memory requirements grow not only with larger embeddings but especially with deeper architectures. I visualized these trends in separate plots for each factor_num and one combined plot to make the comparison easier.

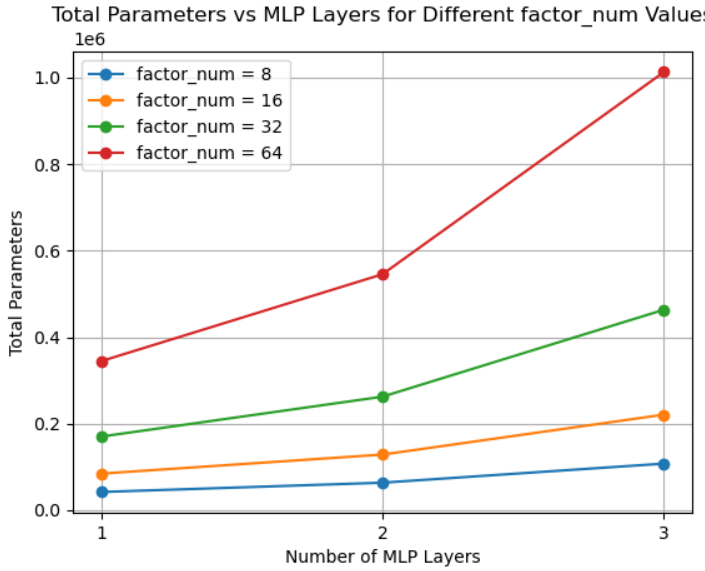
Table 25 : *Total Number of Trainable Parameters in NeuMF Without Pretraining*

factor_num	num_layers	total_params
8	1	42153
8	2	63681
8	3	107761
16	1	84561
16	2	128641
16	3	220897
32	1	170145

32	2	262401
32	3	463297
64	1	344385
64	2	545281
64	3	1012609

Figure 10 : Total Number of Trainable Parameters in NeuMF Without Pretraining for Different MLP Layers and Embedding Sizes





4) Training Dynamics: Loss, HR@10, and NDCG@10 Across Epochs

Training Loss

Table 26 : Training loss – 8 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.3586	0.3562	0.3564	0.3549	0.3539	0.3562	0.3544	0.3564	0.3565	0.4367	0.36402	0.0242588
2	0.2935	0.2931	0.2948	0.296	0.2952	0.2965	0.2949	0.2938	0.2964	0.3365	0.29907	0.0125268
3	0.2752	0.2761	0.2787	0.2804	0.2785	0.2788	0.2773	0.2749	0.2779	0.3045	0.28023	0.0082536
4	0.2623	0.2622	0.2646	0.2666	0.2659	0.2624	0.2622	0.263	0.2628	0.2859	0.26579	0.0068751
5	0.254	0.252	0.2535	0.2562	0.2568	0.2518	0.253	0.2525	0.2522	0.2763	0.25583	0.0070147
6	0.2457	0.2439	0.2437	0.2472	0.2467	0.2432	0.245	0.2451	0.2439	0.2686	0.2473	0.0072086
7	0.2396	0.2371	0.2371	0.2397	0.2387	0.2358	0.2385	0.2391	0.2365	0.2607	0.24028	0.006924
8	0.2328	0.2317	0.2304	0.2338	0.2326	0.2288	0.2327	0.2335	0.2309	0.2527	0.23399	0.0064029
9	0.2274	0.2271	0.2253	0.2293	0.2265	0.2239	0.2273	0.2287	0.2257	0.2447	0.22859	0.0055757
10	0.224	0.2228	0.2208	0.2244	0.2211	0.2183	0.2232	0.2249	0.2207	0.2386	0.22388	0.0052735
11	0.2196	0.2189	0.2163	0.2206	0.2163	0.2145	0.2177	0.2218	0.2159	0.2327	0.21943	0.0049269
12	0.2153	0.2153	0.2125	0.2166	0.2133	0.2111	0.2136	0.2172	0.2118	0.2263	0.2153	0.0041342
13	0.2106	0.2125	0.2098	0.2125	0.2089	0.2068	0.2089	0.2135	0.2092	0.2218	0.21145	0.0039581
14	0.2084	0.2077	0.2061	0.2098	0.2057	0.2049	0.2063	0.2103	0.205	0.2168	0.2081	0.0034135
15	0.204	0.2041	0.2029	0.207	0.2027	0.2011	0.2019	0.2072	0.2021	0.2119	0.20449	0.0031399
16	0.2017	0.2015	0.1997	0.204	0.2002	0.1967	0.1988	0.2043	0.1979	0.207	0.20118	0.0030262
17	0.1992	0.1983	0.1965	0.2003	0.1967	0.1945	0.1959	0.201	0.1949	0.2026	0.19799	0.0025921
18	0.1953	0.195	0.1942	0.1977	0.1942	0.1911	0.1933	0.1984	0.1921	0.1978	0.19491	0.0023343
19	0.1938	0.1923	0.1909	0.1953	0.1916	0.1887	0.1899	0.1949	0.1892	0.1951	0.19217	0.0023762
20	0.1911	0.1896	0.1889	0.1928	0.1892	0.187	0.1869	0.1916	0.1871	0.1925	0.18967	0.0021401

Table 27 : Training loss – 16 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.3472	0.3487	0.3473	0.3481	0.3492	0.3482	0.35	0.3489	0.3489	0.3494	0.34859	0.0008491
2	0.2883	0.2831	0.2855	0.2889	0.2889	0.2843	0.2849	0.2896	0.2827	0.2849	0.28611	0.0024456
3	0.2657	0.2615	0.2657	0.2659	0.2661	0.2639	0.2623	0.2675	0.2619	0.2622	0.26427	0.0020523
4	0.2501	0.2465	0.2511	0.2493	0.2494	0.249	0.2494	0.2507	0.2489	0.2465	0.24909	0.0014611
5	0.2384	0.2359	0.2396	0.2377	0.2375	0.238	0.2369	0.2389	0.2375	0.2353	0.23757	0.0012321
6	0.2279	0.2262	0.2297	0.2277	0.228	0.2282	0.2275	0.228	0.2289	0.2258	0.22779	0.001083
7	0.2191	0.2174	0.2202	0.2185	0.2183	0.22	0.2179	0.2188	0.2208	0.2181	0.21891	0.0010473
8	0.2095	0.2098	0.2117	0.2104	0.2095	0.2106	0.2103	0.2108	0.212	0.2095	0.21041	0.0008514
9	0.2026	0.2021	0.204	0.2037	0.2024	0.2023	0.2036	0.204	0.2054	0.2027	0.20328	0.0009867
10	0.1946	0.1962	0.1966	0.1973	0.1951	0.1951	0.1975	0.1962	0.198	0.1962	0.19628	0.0010591
11	0.1873	0.1898	0.1901	0.1915	0.1887	0.1887	0.1915	0.1899	0.1918	0.1903	0.18996	0.0013618
12	0.1821	0.1827	0.1836	0.1844	0.1822	0.1826	0.1862	0.1837	0.1864	0.1841	0.1838	0.0014533
13	0.1757	0.1772	0.1783	0.1782	0.1764	0.1766	0.1806	0.1782	0.1808	0.1778	0.17798	0.0015892
14	0.169	0.1719	0.1728	0.1729	0.1709	0.17	0.1757	0.1732	0.1756	0.1724	0.17244	0.002049
15	0.1635	0.1664	0.1666	0.1679	0.1652	0.1644	0.17	0.1675	0.17	0.168	0.16695	0.0020689
16	0.1574	0.1608	0.1622	0.1622	0.16	0.1602	0.1657	0.1627	0.1644	0.1622	0.16178	0.002223
17	0.154	0.1566	0.1566	0.1576	0.1559	0.1546	0.1596	0.1586	0.1602	0.1576	0.15713	0.0019079
18	0.1476	0.1527	0.1529	0.1534	0.1511	0.1496	0.1561	0.1536	0.1571	0.1527	0.15268	0.002654
19	0.1445	0.1475	0.1475	0.1486	0.1462	0.1465	0.1515	0.1493	0.152	0.1487	0.14823	0.0022023
20	0.1393	0.1431	0.1439	0.1441	0.1426	0.141	0.147	0.1446	0.1481	0.1436	0.14373	0.0024389

Table 28 : Training loss – 32 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.3428	0.3418	0.3431	0.3417	0.3422	0.3422	0.3436	0.3437	0.342	0.3422	0.34253	0.0006885
2	0.2783	0.2754	0.2778	0.2783	0.2779	0.2785	0.2799	0.2741	0.2735	0.2753	0.2769	0.0020372
3	0.2506	0.2503	0.2505	0.2518	0.2522	0.2515	0.2535	0.2494	0.2493	0.2509	0.251	0.0012223
4	0.2317	0.2339	0.2325	0.2343	0.2353	0.2333	0.2362	0.2321	0.2339	0.2342	0.23374	0.0013283
5	0.2175	0.2205	0.2174	0.2184	0.2195	0.2189	0.2207	0.217	0.2204	0.2198	0.21901	0.0013103
6	0.2034	0.2067	0.2025	0.2052	0.2049	0.2045	0.2052	0.2038	0.2055	0.2058	0.20475	0.0011724
7	0.1909	0.1933	0.1901	0.1918	0.1914	0.1916	0.1916	0.1911	0.192	0.1912	0.1915	0.0007861
8	0.1799	0.1815	0.1772	0.1807	0.179	0.179	0.1801	0.1794	0.1812	0.1793	0.17973	0.00119
9	0.1689	0.1711	0.1666	0.1695	0.1681	0.1677	0.1695	0.1695	0.1709	0.1685	0.16903	0.0013146
10	0.1598	0.1622	0.1569	0.1607	0.1579	0.158	0.1598	0.1604	0.1611	0.1584	0.15952	0.0015829
11	0.1509	0.1526	0.1471	0.1505	0.1493	0.1486	0.1516	0.1523	0.1524	0.1497	0.1505	0.0017286
12	0.1431	0.1453	0.1377	0.1439	0.1422	0.1409	0.1439	0.1444	0.1453	0.142	0.14287	0.0021941
13	0.1358	0.1372	0.1309	0.1362	0.1342	0.1326	0.1359	0.1377	0.1374	0.1346	0.13525	0.0020863
14	0.1271	0.13	0.1236	0.1287	0.1269	0.126	0.128	0.13	0.1304	0.1272	0.12779	0.0019957
15	0.1215	0.1235	0.1161	0.1224	0.1201	0.1192	0.1214	0.1238	0.1242	0.1208	0.1213	0.0023216
16	0.1149	0.1178	0.1108	0.117	0.1142	0.1124	0.1143	0.1169	0.1169	0.1148	0.115	0.0021128
17	0.1097	0.1112	0.1041	0.1095	0.1079	0.1084	0.11	0.111	0.1113	0.1088	0.10919	0.002028
18	0.1035	0.1054	0.0991	0.1047	0.1035	0.1012	0.1036	0.1063	0.1063	0.1031	0.10367	0.0021275
19	0.0977	0.0994	0.0937	0.0996	0.0978	0.0977	0.0984	0.1011	0.1011	0.0985	0.0985	0.002009
20	0.0939	0.0949	0.0894	0.0948	0.0933	0.0925	0.0926	0.097	0.0956	0.0936	0.09376	0.0019612

Table 29 : Training loss – 64 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.3361	0.3363	0.3377	0.3376	0.3377	0.3384	0.3381	0.3374	0.3371	0.3381	0.33745	0.0007187
2	0.2662	0.268	0.2676	0.2666	0.2665	0.2666	0.2676	0.2678	0.2698	0.2703	0.2677	0.0013191
3	0.2388	0.2389	0.2379	0.2372	0.2379	0.2354	0.2386	0.2397	0.2402	0.2412	0.23858	0.0015439
4	0.2162	0.2167	0.2155	0.2142	0.2148	0.2131	0.2145	0.2166	0.2172	0.2175	0.21563	0.0013712
5	0.1957	0.1938	0.1937	0.1924	0.1924	0.1918	0.1941	0.1947	0.1948	0.1944	0.19378	0.0011746
6	0.1736	0.1729	0.1731	0.1717	0.1705	0.171	0.1732	0.1732	0.1726	0.1729	0.17247	0.000988
7	0.1544	0.1532	0.153	0.1523	0.1526	0.1523	0.1534	0.1541	0.154	0.1534	0.15327	0.0007029
8	0.1385	0.1361	0.1369	0.1356	0.1358	0.1362	0.1364	0.1377	0.1366	0.1363	0.13661	0.000842
9	0.1234	0.1212	0.1227	0.1212	0.1211	0.1223	0.1217	0.1221	0.1224	0.1222	0.12203	0.0007015
10	0.1117	0.1089	0.1112	0.1093	0.1093	0.1107	0.1092	0.1103	0.1109	0.1102	0.11017	0.0009111
11	0.1008	0.0993	0.1018	0.0989	0.0993	0.1002	0.0985	0.1	0.1008	0.1003	0.09999	0.0009534
12	0.0926	0.0899	0.0916	0.0892	0.0908	0.0914	0.0897	0.0911	0.0919	0.0911	0.09093	0.001002
13	0.0848	0.0815	0.0848	0.082	0.082	0.0843	0.0814	0.0836	0.0855	0.0835	0.08334	0.0014382
14	0.0779	0.0754	0.0779	0.0755	0.075	0.0765	0.0746	0.0769	0.0771	0.077	0.07638	0.0011232
15	0.0718	0.0681	0.0721	0.0696	0.0697	0.0712	0.0682	0.0709	0.0715	0.0705	0.07036	0.0013492
16	0.0658	0.0625	0.0664	0.0637	0.0644	0.0652	0.0626	0.0655	0.0659	0.0656	0.06476	0.0013245
17	0.0616	0.0575	0.0619	0.0592	0.0582	0.0605	0.0579	0.0605	0.0617	0.0605	0.05995	0.0015596
18	0.0567	0.0545	0.0561	0.0549	0.0548	0.0565	0.0538	0.0561	0.0564	0.0567	0.05565	0.0009962
19	0.0535	0.0499	0.0529	0.0512	0.05	0.052	0.0496	0.052	0.0537	0.0517	0.05165	0.0014009
20	0.0494	0.0461	0.0494	0.0472	0.0458	0.049	0.0462	0.048	0.0496	0.0492	0.04799	0.0014577

HR@10

Table 30 : HR@10– 8 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.526	0.523	0.521	0.514	0.521	0.521	0.517	0.519	0.51	0.402	0.5074	0.035398
2	0.576	0.56	0.581	0.554	0.574	0.566	0.567	0.583	0.561	0.494	0.5616	0.024237
3	0.596	0.593	0.591	0.585	0.595	0.597	0.575	0.593	0.601	0.551	0.5877	0.014029
4	0.615	0.61	0.62	0.615	0.616	0.618	0.609	0.607	0.614	0.567	0.6091	0.01455
5	0.616	0.628	0.644	0.63	0.627	0.646	0.617	0.621	0.637	0.574	0.624	0.019329
6	0.635	0.645	0.662	0.641	0.643	0.636	0.641	0.642	0.642	0.608	0.6395	0.012612
7	0.634	0.651	0.652	0.647	0.644	0.646	0.63	0.643	0.656	0.612	0.6415	0.012363
8	0.638	0.659	0.634	0.651	0.636	0.662	0.655	0.643	0.637	0.625	0.644	0.011533
9	0.655	0.656	0.642	0.661	0.638	0.656	0.652	0.646	0.659	0.63	0.6495	0.009615
10	0.652	0.656	0.659	0.652	0.637	0.646	0.647	0.645	0.662	0.637	0.6493	0.008075
11	0.644	0.649	0.652	0.656	0.651	0.644	0.641	0.649	0.651	0.635	0.6472	0.005862
12	0.636	0.661	0.636	0.642	0.662	0.65	0.652	0.648	0.659	0.635	0.6481	0.009994
13	0.634	0.652	0.661	0.648	0.645	0.653	0.644	0.65	0.645	0.629	0.6461	0.008769
14	0.648	0.661	0.637	0.636	0.637	0.665	0.653	0.648	0.651	0.621	0.6457	0.012466
15	0.629	0.649	0.656	0.634	0.627	0.661	0.651	0.652	0.651	0.635	0.6445	0.011456
16	0.648	0.645	0.645	0.639	0.629	0.638	0.647	0.651	0.653	0.635	0.643	0.007169
17	0.633	0.642	0.639	0.62	0.632	0.651	0.642	0.649	0.641	0.634	0.6383	0.008556
18	0.63	0.633	0.629	0.629	0.635	0.645	0.644	0.637	0.637	0.634	0.6353	0.005386
19	0.616	0.644	0.633	0.633	0.612	0.641	0.637	0.643	0.647	0.648	0.6354	0.011808
20	0.629	0.642	0.634	0.632	0.624	0.637	0.646	0.646	0.625	0.628	0.6343	0.007785

Table 31 : HR@10– 16 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.52	0.542	0.537	0.533	0.539	0.539	0.524	0.539	0.546	0.548	0.5367	0.008439
2	0.592	0.584	0.594	0.6	0.584	0.607	0.582	0.575	0.595	0.587	0.59	0.008967
3	0.601	0.617	0.612	0.63	0.615	0.62	0.624	0.601	0.621	0.618	0.6159	0.008791
4	0.621	0.641	0.635	0.636	0.63	0.635	0.627	0.622	0.636	0.641	0.6324	0.006785
5	0.653	0.657	0.641	0.653	0.66	0.647	0.643	0.649	0.646	0.648	0.6497	0.005711
6	0.649	0.657	0.662	0.648	0.659	0.644	0.655	0.644	0.66	0.641	0.6519	0.007217
7	0.652	0.653	0.648	0.66	0.657	0.664	0.659	0.66	0.651	0.652	0.6556	0.004841
8	0.667	0.663	0.657	0.662	0.645	0.659	0.659	0.661	0.644	0.65	0.6567	0.00739
9	0.659	0.655	0.662	0.666	0.655	0.642	0.663	0.647	0.65	0.636	0.6535	0.009201
10	0.662	0.655	0.646	0.652	0.638	0.638	0.667	0.653	0.659	0.645	0.6515	0.009266
11	0.652	0.654	0.654	0.649	0.653	0.633	0.657	0.647	0.653	0.638	0.649	0.007321
12	0.635	0.649	0.652	0.649	0.65	0.638	0.638	0.648	0.648	0.652	0.6459	0.006024
13	0.645	0.66	0.644	0.666	0.648	0.627	0.65	0.636	0.638	0.643	0.6457	0.010743
14	0.642	0.649	0.635	0.657	0.63	0.627	0.656	0.62	0.647	0.637	0.64	0.011756
15	0.636	0.65	0.636	0.65	0.628	0.633	0.637	0.617	0.642	0.641	0.637	0.00937
16	0.63	0.638	0.643	0.644	0.632	0.629	0.639	0.625	0.636	0.63	0.6346	0.006037
17	0.632	0.625	0.631	0.636	0.622	0.616	0.653	0.625	0.646	0.63	0.6316	0.010538
18	0.629	0.632	0.626	0.628	0.621	0.612	0.653	0.612	0.639	0.624	0.6276	0.011586
19	0.617	0.621	0.609	0.617	0.629	0.598	0.642	0.619	0.634	0.633	0.6219	0.012324
20	0.617	0.631	0.612	0.621	0.626	0.608	0.622	0.618	0.629	0.632	0.6216	0.007632

Table 32 : HR@10– 32 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.549	0.552	0.54	0.546	0.556	0.536	0.547	0.556	0.555	0.56	0.5497	0.007226
2	0.598	0.621	0.616	0.602	0.603	0.613	0.595	0.618	0.616	0.615	0.6097	0.008787
3	0.631	0.651	0.648	0.639	0.642	0.637	0.63	0.643	0.629	0.633	0.6383	0.007281
4	0.642	0.66	0.666	0.654	0.649	0.645	0.641	0.645	0.65	0.647	0.6499	0.007595
5	0.635	0.659	0.663	0.655	0.661	0.651	0.642	0.654	0.662	0.653	0.6535	0.008559
6	0.654	0.665	0.654	0.655	0.661	0.652	0.648	0.651	0.67	0.653	0.6563	0.006512
7	0.654	0.66	0.665	0.652	0.665	0.65	0.644	0.654	0.651	0.649	0.6544	0.00656
8	0.646	0.644	0.668	0.643	0.647	0.648	0.66	0.642	0.657	0.651	0.6506	0.008052
9	0.645	0.636	0.646	0.634	0.651	0.636	0.633	0.639	0.65	0.639	0.6409	0.006268
10	0.624	0.646	0.647	0.638	0.643	0.649	0.636	0.636	0.657	0.622	0.6398	0.01039
11	0.62	0.642	0.639	0.626	0.621	0.648	0.638	0.63	0.641	0.625	0.633	0.009306
12	0.604	0.614	0.644	0.628	0.629	0.635	0.645	0.615	0.649	0.606	0.6269	0.015649
13	0.602	0.632	0.634	0.629	0.626	0.624	0.639	0.621	0.644	0.6	0.6251	0.013693
14	0.6	0.625	0.635	0.615	0.629	0.616	0.633	0.624	0.644	0.607	0.6228	0.012742
15	0.602	0.619	0.615	0.601	0.617	0.612	0.637	0.621	0.636	0.6	0.616	0.01253
16	0.586	0.619	0.608	0.608	0.601	0.615	0.625	0.618	0.636	0.601	0.6117	0.013372
17	0.592	0.616	0.606	0.608	0.598	0.612	0.62	0.609	0.619	0.614	0.6094	0.008499
18	0.569	0.609	0.599	0.591	0.595	0.601	0.615	0.611	0.624	0.596	0.601	0.014415
19	0.591	0.609	0.608	0.592	0.59	0.603	0.616	0.602	0.626	0.595	0.6032	0.011214
20	0.59	0.604	0.606	0.58	0.587	0.6	0.599	0.612	0.618	0.587	0.5983	0.011533

Table 33 : HR@10– 64 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.572	0.562	0.56	0.564	0.567	0.576	0.565	0.571	0.551	0.561	0.5649	0.006759
2	0.627	0.619	0.628	0.621	0.617	0.633	0.621	0.613	0.629	0.619	0.6227	0.005934
3	0.644	0.642	0.653	0.638	0.643	0.63	0.649	0.642	0.652	0.645	0.6438	0.006416
4	0.633	0.659	0.667	0.648	0.646	0.667	0.665	0.648	0.648	0.652	0.6533	0.010469
5	0.642	0.654	0.668	0.651	0.655	0.65	0.655	0.648	0.654	0.659	0.6536	0.00653
6	0.657	0.664	0.667	0.629	0.643	0.647	0.647	0.645	0.648	0.649	0.6496	0.010346
7	0.645	0.651	0.639	0.639	0.643	0.627	0.639	0.643	0.621	0.649	0.6396	0.008811
8	0.637	0.661	0.636	0.641	0.645	0.632	0.644	0.628	0.63	0.638	0.6392	0.00902
9	0.626	0.643	0.621	0.625	0.637	0.619	0.635	0.622	0.611	0.63	0.6269	0.009027
10	0.616	0.626	0.629	0.615	0.635	0.615	0.609	0.627	0.601	0.624	0.6197	0.009768
11	0.616	0.614	0.613	0.603	0.634	0.614	0.609	0.61	0.602	0.607	0.6122	0.00853
12	0.607	0.614	0.62	0.603	0.632	0.615	0.589	0.612	0.606	0.613	0.6111	0.010681
13	0.609	0.608	0.604	0.602	0.598	0.601	0.589	0.615	0.6	0.596	0.6022	0.006954
14	0.598	0.6	0.598	0.587	0.586	0.599	0.586	0.594	0.592	0.591	0.5931	0.005243
15	0.594	0.601	0.583	0.594	0.624	0.591	0.575	0.583	0.598	0.589	0.5932	0.012632
16	0.59	0.596	0.587	0.582	0.611	0.604	0.594	0.586	0.6	0.584	0.5934	0.008935
17	0.582	0.603	0.596	0.576	0.617	0.581	0.576	0.586	0.576	0.575	0.5868	0.013437
18	0.584	0.586	0.592	0.581	0.589	0.594	0.584	0.579	0.589	0.569	0.5847	0.006871
19	0.579	0.597	0.586	0.576	0.603	0.587	0.585	0.591	0.58	0.581	0.5865	0.008028
20	0.575	0.58	0.592	0.578	0.597	0.565	0.592	0.585	0.591	0.576	0.5831	0.00947

NDCG@10

Table 34 : NDCG@10– 8 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.286	0.288	0.288	0.281	0.287	0.281	0.285	0.287	0.283	0.217	0.2783	0.020587
2	0.314	0.307	0.315	0.31	0.315	0.305	0.315	0.32	0.313	0.274	0.3088	0.012311
3	0.329	0.328	0.319	0.325	0.326	0.327	0.326	0.332	0.333	0.308	0.3253	0.006842
4	0.335	0.338	0.345	0.341	0.345	0.347	0.34	0.342	0.339	0.32	0.3392	0.007264
5	0.345	0.35	0.36	0.347	0.352	0.355	0.344	0.347	0.352	0.322	0.3474	0.009635
6	0.354	0.356	0.374	0.359	0.361	0.359	0.354	0.358	0.357	0.333	0.3565	0.009521
7	0.362	0.363	0.372	0.366	0.36	0.364	0.357	0.361	0.363	0.338	0.3606	0.008417
8	0.361	0.37	0.366	0.364	0.364	0.369	0.372	0.359	0.362	0.347	0.3634	0.006726
9	0.375	0.365	0.365	0.368	0.36	0.377	0.371	0.367	0.376	0.353	0.3677	0.007142
10	0.371	0.362	0.379	0.366	0.359	0.376	0.375	0.361	0.375	0.359	0.3683	0.007363
11	0.368	0.366	0.383	0.37	0.362	0.373	0.37	0.369	0.369	0.363	0.3693	0.005551
12	0.366	0.369	0.372	0.368	0.366	0.377	0.374	0.366	0.377	0.36	0.3695	0.005182
13	0.359	0.365	0.388	0.369	0.372	0.382	0.372	0.367	0.372	0.356	0.3702	0.009119
14	0.368	0.373	0.374	0.363	0.369	0.383	0.378	0.371	0.378	0.355	0.3712	0.007666
15	0.364	0.367	0.377	0.366	0.359	0.385	0.375	0.37	0.378	0.358	0.3699	0.0083
16	0.369	0.36	0.373	0.364	0.366	0.376	0.373	0.368	0.376	0.357	0.3682	0.006194
17	0.36	0.361	0.374	0.36	0.36	0.379	0.365	0.363	0.375	0.358	0.3655	0.007201
18	0.358	0.356	0.363	0.362	0.364	0.376	0.361	0.364	0.369	0.354	0.3627	0.006051
19	0.36	0.356	0.362	0.36	0.35	0.373	0.363	0.365	0.372	0.366	0.3627	0.006558
20	0.363	0.358	0.366	0.356	0.357	0.37	0.365	0.37	0.363	0.357	0.3625	0.005045

Table 35 : NDCG@10– 16 factor_num

Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.287	0.298	0.291	0.29	0.294	0.296	0.294	0.296	0.3	0.298	0.2944	0.003852
2	0.32	0.324	0.326	0.327	0.321	0.328	0.322	0.32	0.33	0.322	0.324	0.003376
3	0.342	0.347	0.34	0.344	0.345	0.346	0.345	0.337	0.345	0.341	0.3432	0.00296
4	0.352	0.357	0.357	0.352	0.355	0.358	0.348	0.349	0.357	0.352	0.3537	0.003407
5	0.372	0.368	0.358	0.369	0.377	0.365	0.362	0.363	0.361	0.362	0.3657	0.005478
6	0.368	0.372	0.371	0.366	0.373	0.365	0.375	0.365	0.373	0.368	0.3696	0.00347
7	0.377	0.377	0.369	0.371	0.38	0.377	0.384	0.373	0.365	0.371	0.3744	0.005352
8	0.375	0.383	0.377	0.373	0.37	0.385	0.383	0.374	0.368	0.375	0.3763	0.005423
9	0.375	0.379	0.382	0.374	0.379	0.372	0.378	0.37	0.376	0.365	0.375	0.004754
10	0.377	0.378	0.372	0.368	0.371	0.367	0.386	0.368	0.378	0.372	0.3737	0.005675
11	0.376	0.38	0.371	0.375	0.37	0.37	0.378	0.377	0.371	0.37	0.3738	0.003628
12	0.37	0.368	0.374	0.372	0.375	0.369	0.366	0.372	0.37	0.382	0.3718	0.004261
13	0.367	0.375	0.37	0.377	0.37	0.361	0.379	0.358	0.366	0.373	0.3696	0.006422
14	0.363	0.376	0.373	0.368	0.365	0.363	0.372	0.359	0.371	0.367	0.3677	0.005041
15	0.363	0.374	0.374	0.36	0.357	0.367	0.366	0.355	0.369	0.369	0.3654	0.006248
16	0.36	0.364	0.369	0.355	0.363	0.362	0.367	0.354	0.37	0.366	0.363	0.005158
17	0.358	0.354	0.368	0.355	0.357	0.361	0.379	0.357	0.374	0.359	0.3622	0.008109
18	0.359	0.365	0.364	0.354	0.357	0.359	0.37	0.349	0.365	0.358	0.36	0.005814
19	0.365	0.35	0.359	0.349	0.363	0.348	0.364	0.35	0.364	0.36	0.3572	0.006735
20	0.356	0.36	0.358	0.353	0.363	0.35	0.356	0.35	0.358	0.364	0.3568	0.0046

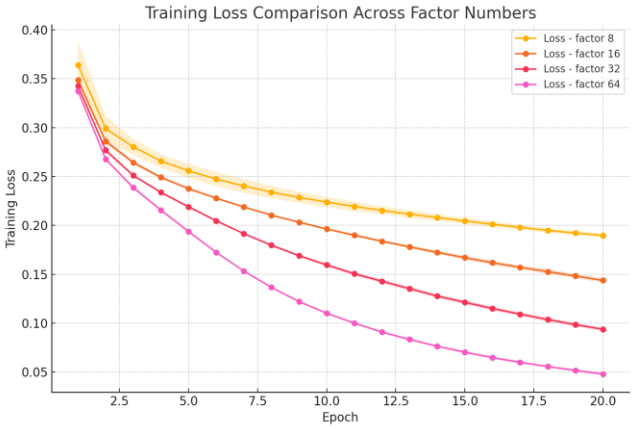
Table 36 : NDCG@10– 32 factor_num

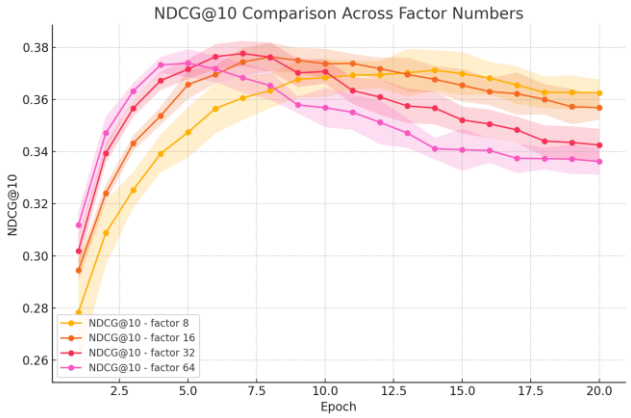
Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.297	0.307	0.294	0.307	0.305	0.292	0.299	0.309	0.305	0.304	0.3019	0.005647
2	0.332	0.343	0.34	0.338	0.336	0.342	0.34	0.346	0.34	0.337	0.3394	0.00372
3	0.351	0.359	0.362	0.358	0.352	0.357	0.358	0.358	0.353	0.358	0.3566	0.003292
4	0.365	0.368	0.377	0.365	0.367	0.363	0.367	0.366	0.367	0.367	0.3672	0.003544
5	0.362	0.374	0.374	0.373	0.371	0.372	0.372	0.375	0.375	0.368	0.3716	0.003774
6	0.375	0.381	0.371	0.374	0.381	0.372	0.376	0.381	0.384	0.369	0.3764	0.004821
7	0.375	0.378	0.379	0.379	0.385	0.369	0.379	0.383	0.379	0.37	0.3776	0.0048
8	0.377	0.368	0.378	0.375	0.383	0.375	0.385	0.373	0.381	0.367	0.3762	0.005618
9	0.374	0.367	0.364	0.362	0.381	0.373	0.368	0.375	0.367	0.371	0.3702	0.005418
10	0.369	0.371	0.368	0.368	0.383	0.373	0.371	0.369	0.37	0.365	0.3707	0.004584
11	0.36	0.365	0.358	0.361	0.374	0.367	0.368	0.356	0.365	0.36	0.3634	0.005142
12	0.357	0.353	0.364	0.361	0.37	0.359	0.365	0.361	0.371	0.348	0.3609	0.006774
13	0.35	0.361	0.356	0.356	0.366	0.348	0.36	0.369	0.359	0.35	0.3575	0.006576
14	0.355	0.36	0.361	0.348	0.367	0.353	0.358	0.362	0.358	0.345	0.3567	0.006294
15	0.347	0.353	0.351	0.345	0.357	0.347	0.361	0.357	0.356	0.347	0.3521	0.005224
16	0.347	0.356	0.348	0.346	0.349	0.347	0.355	0.362	0.353	0.343	0.3506	0.005463
17	0.342	0.353	0.347	0.341	0.345	0.348	0.354	0.358	0.348	0.347	0.3483	0.005061
18	0.334	0.344	0.338	0.339	0.342	0.346	0.349	0.351	0.355	0.342	0.344	0.006066
19	0.34	0.342	0.347	0.335	0.341	0.339	0.353	0.351	0.351	0.336	0.3435	0.006201
20	0.345	0.342	0.344	0.337	0.333	0.338	0.346	0.352	0.352	0.336	0.3425	0.006201

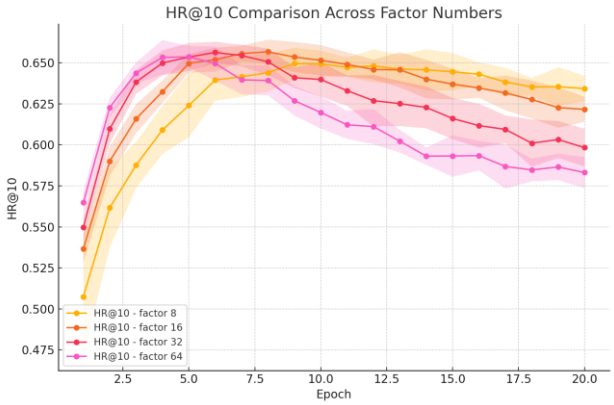
Table 37 : NDCG@10– 64 factor_num

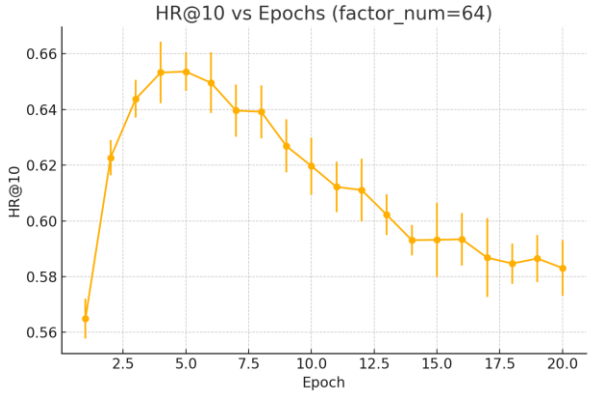
Epoch	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.316	0.312	0.306	0.308	0.312	0.323	0.311	0.315	0.311	0.304	0.3118	0.005134
2	0.349	0.348	0.346	0.338	0.345	0.359	0.345	0.341	0.357	0.343	0.3471	0.006252
3	0.365	0.366	0.362	0.357	0.36	0.365	0.367	0.361	0.366	0.363	0.3632	0.003027
4	0.375	0.371	0.374	0.374	0.368	0.38	0.374	0.373	0.37	0.373	0.3732	0.003059
5	0.367	0.373	0.376	0.375	0.377	0.374	0.375	0.363	0.375	0.384	0.3739	0.005356
6	0.374	0.38	0.371	0.36	0.374	0.369	0.373	0.369	0.376	0.371	0.3717	0.005021
7	0.374	0.365	0.359	0.363	0.376	0.362	0.373	0.368	0.368	0.375	0.3683	0.005693
8	0.36	0.37	0.359	0.359	0.378	0.363	0.365	0.364	0.367	0.368	0.3653	0.005551
9	0.356	0.363	0.354	0.358	0.355	0.355	0.362	0.356	0.358	0.362	0.3579	0.003145
10	0.356	0.361	0.353	0.35	0.364	0.357	0.339	0.361	0.363	0.364	0.3568	0.007454
11	0.356	0.36	0.351	0.348	0.37	0.352	0.346	0.351	0.356	0.361	0.3551	0.006774
12	0.352	0.362	0.349	0.34	0.362	0.347	0.335	0.353	0.359	0.353	0.3512	0.008436
13	0.351	0.353	0.342	0.341	0.349	0.35	0.335	0.348	0.354	0.348	0.3471	0.005665
14	0.347	0.342	0.337	0.337	0.336	0.342	0.341	0.337	0.348	0.344	0.3411	0.00411
15	0.344	0.344	0.329	0.338	0.36	0.338	0.332	0.336	0.344	0.342	0.3407	0.0081
16	0.345	0.345	0.336	0.331	0.347	0.341	0.336	0.339	0.342	0.342	0.3404	0.004695
17	0.341	0.341	0.334	0.331	0.35	0.333	0.33	0.337	0.337	0.34	0.3374	0.005643
18	0.337	0.334	0.337	0.333	0.339	0.335	0.333	0.335	0.346	0.344	0.3373	0.004267
19	0.335	0.342	0.33	0.327	0.338	0.341	0.336	0.334	0.341	0.347	0.3371	0.005665
20	0.335	0.333	0.331	0.332	0.337	0.33	0.339	0.336	0.342	0.347	0.3362	0.005036

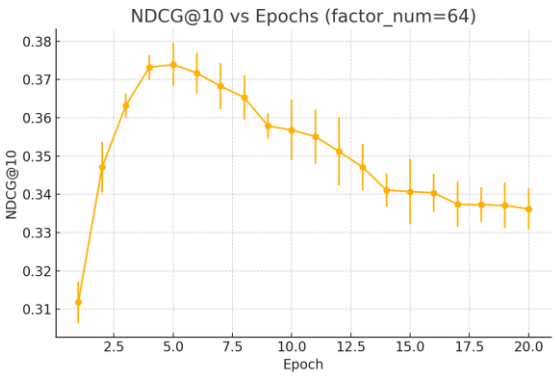
Figure 11: Results represented in figures

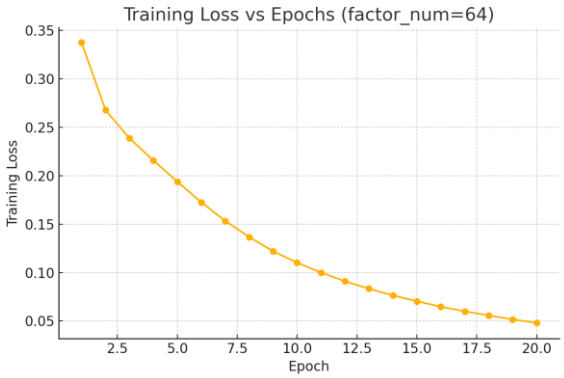


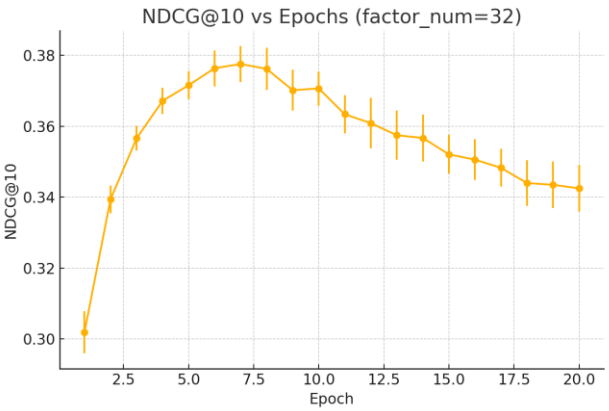


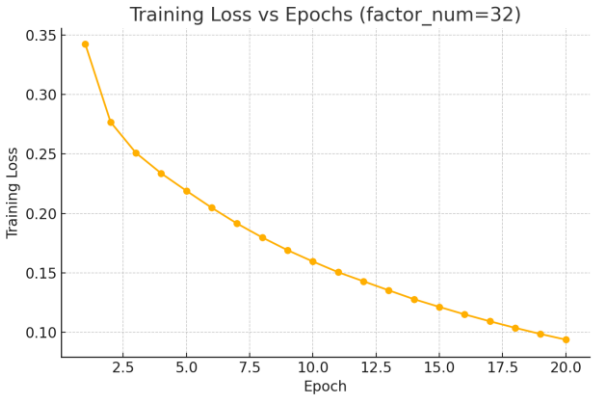


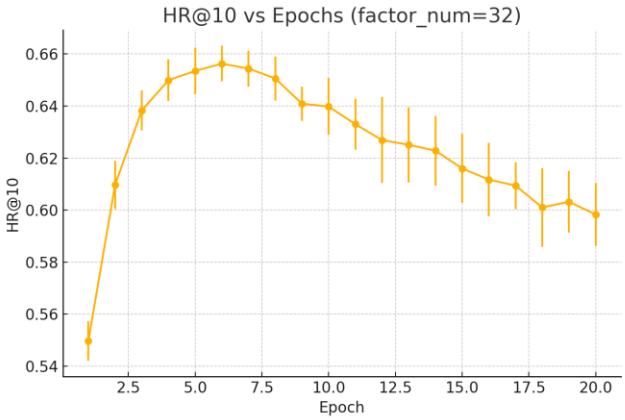


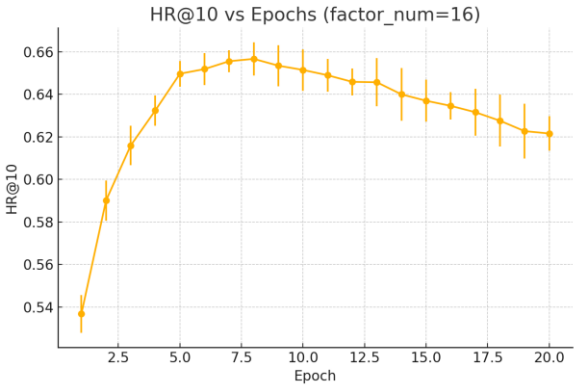


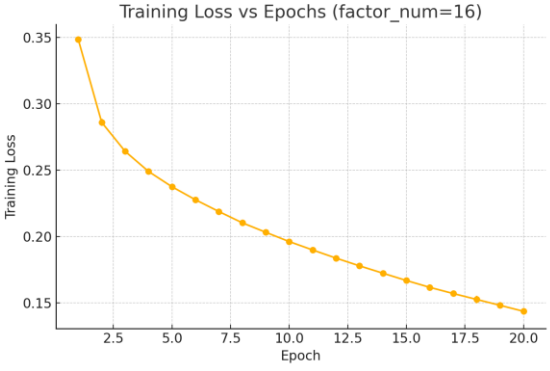


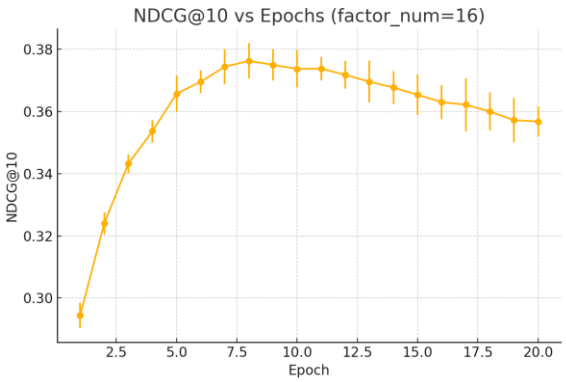

NDCG@10 vs Epochs (factor_num=32)
NDCG@10
Epoch
0.38
0.36
0.34
0.32
0.30
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

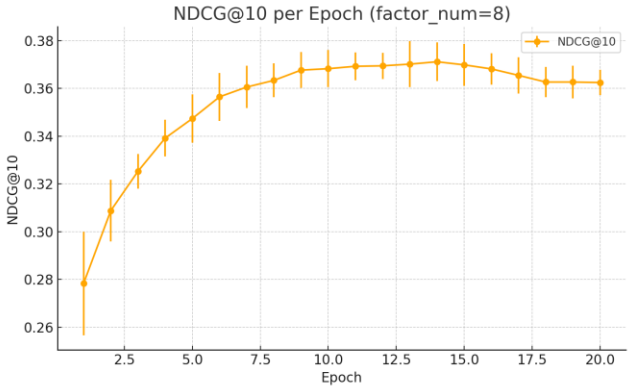

Training Loss vs Epochs (factor_num=32)
Training Loss
Epoch
0.35
0.30
0.25
0.20
0.15
0.10
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

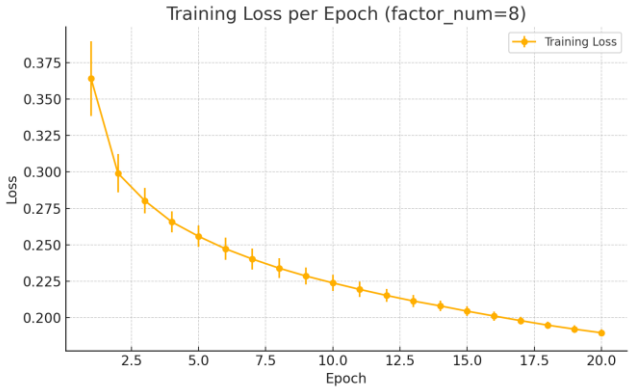

HR@10 vs Epochs (factor_num=32)
HR@10
Epoch
0.66
0.64
0.62
0.60
0.58
0.56
0.54
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

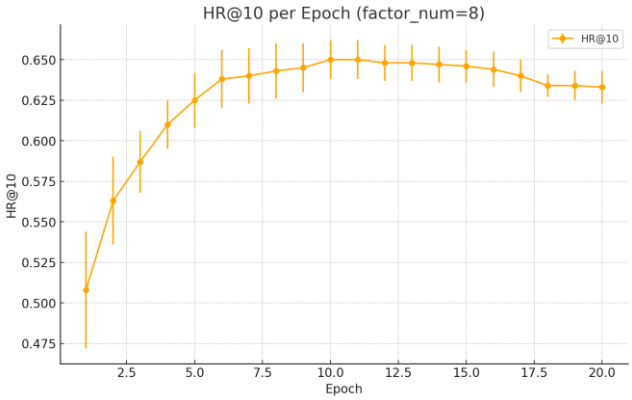

HR@10 vs Epochs (factor_num=16)
HR@10
Epoch
0.66
0.64
0.62
0.60
0.58
0.56
0.54
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training Loss vs Epochs (factor_num=16)
Training Loss
Epoch
0.35
0.30
0.25
0.20
0.15
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


NDCG@10 vs Epochs (factor_num=16)
NDCG@10
Epoch
0.38
0.36
0.34
0.32
0.30
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


NDCG@10 per Epoch (factor_num=8)
NDCG@10
NDCG@10
Epoch
0.38
0.36
0.34
0.32
0.30
0.28
0.26
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training Loss per Epoch (factor_num=8)
Training Loss
Loss
Epoch
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


HR@10 per Epoch (factor_num=8)
HR@10
HR@10
Epoch
0.650
0.625
0.600
0.575
0.550
0.525
0.500
0.475
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

To answer this question, I trained the model with different factor_num values (8, 16, 32, 64) and tracked how the training loss, HR@10, and NDCG@10 changed over 20 epochs. For each setting, I ran the training 10 times and averaged the results to make the graphs more reliable.

From the training loss plot, it's clear that models with higher factor_num (especially 64) managed to reduce the loss much faster and ended up with the lowest loss overall. So, larger factor sizes help the model fit the training data better.

However, when I looked at HR@10 and NDCG@10, the results weren't as straightforward. While factor 64 started strong, it didn't always maintain the best performance later in training. In fact, I noticed some signs of overfitting—its HR and NDCG dropped a bit in the later epochs. On the other hand, factor values like 16 and 32 showed more stable and consistently good performance across epochs.

So overall, even though a bigger factor_num gives lower training loss, it doesn't always mean better recommendation quality. Based on the results, I'd say that factor_num = 16 or 32 is probably the best choice because it gives a good balance between learning and generalization.

Best parameter setting chosen

For the questions below, I chose this final (best) parameter setting for NeuMF:

Batch size: 256

Learning rate: 0.001

Iterations: 20

Number of Factors: 32

MLP Layers: 3

5) HR@K vs K: Top-K Recommendation Performance

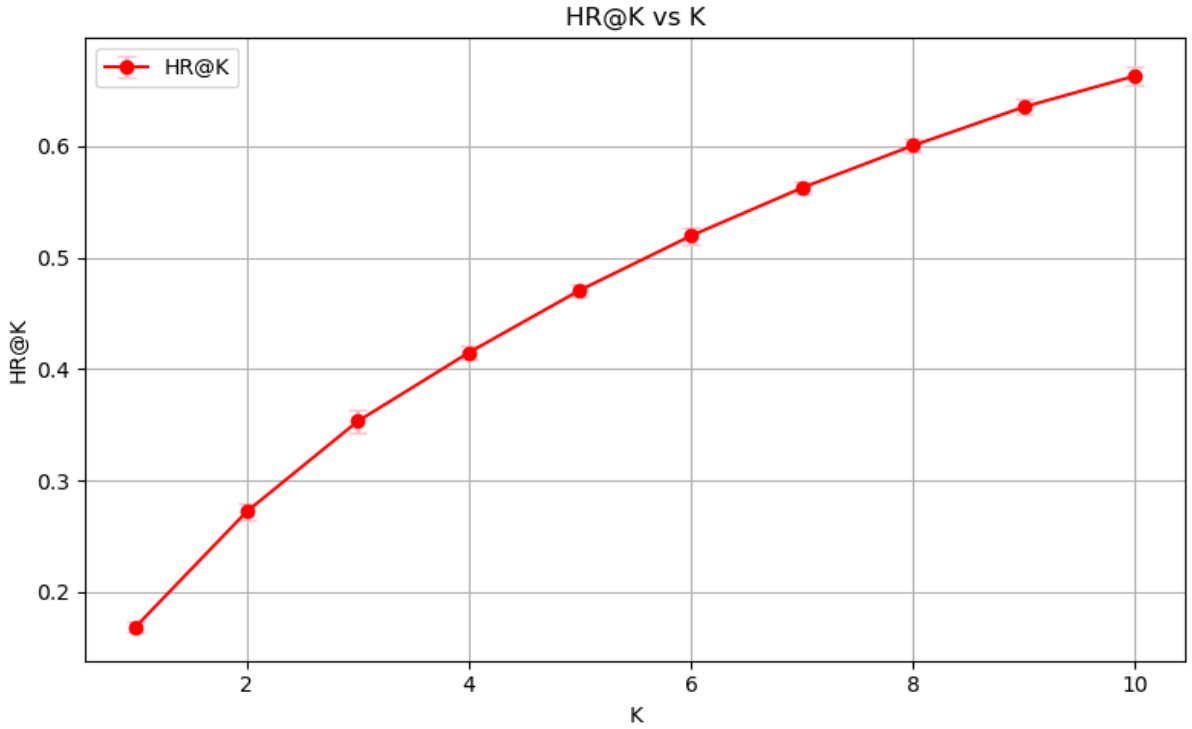
In this experiment, I tested how the NeuMF model performs in Top-K recommendation by changing K from 1 to 10. For each value of K, I ran the model 10 times and calculated the average HR@K, as well as the errorbars to see how much they varied.

As expected, HR@K increases as K gets bigger which makes sense, since it's more likely that the correct item is included when we recommend more options. The model showed stable performance across runs, with only small differences between the lowest and highest values. This means NeuMF is quite reliable and good at ranking relevant items near the top.

Table 38: HR@K values for every run , mean and std

K	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.162	0.168	0.163	0.165	0.165	0.173	0.172	0.17	0.179	0.168	0.1685	0.0049244
2	0.273	0.271	0.284	0.262	0.276	0.281	0.264	0.267	0.265	0.276	0.2719	0.0070491
3	0.349	0.347	0.349	0.361	0.343	0.373	0.347	0.362	0.361	0.338	0.353	0.0101882
4	0.417	0.423	0.406	0.406	0.41	0.411	0.424	0.416	0.416	0.417	0.4146	0.0059699
5	0.472	0.458	0.475	0.469	0.472	0.48	0.472	0.473	0.472	0.463	0.4706	0.0058344
6	0.52	0.522	0.519	0.515	0.516	0.524	0.501	0.525	0.52	0.532	0.5194	0.0076707
7	0.577	0.56	0.565	0.555	0.561	0.563	0.566	0.561	0.56	0.556	0.5624	0.0058686
8	0.611	0.603	0.6	0.6	0.589	0.598	0.595	0.604	0.593	0.611	0.6004	0.0068147
9	0.622	0.642	0.638	0.643	0.641	0.635	0.636	0.63	0.638	0.626	0.6351	0.0066551
10	0.66	0.659	0.672	0.661	0.673	0.657	0.677	0.648	0.656	0.666	0.6629	0.0085141

Figure 12 : HR@K vs K



6) NDCG@K vs K: Ranking Quality Across Different K

In this experiment, I evaluated how the ranking quality of the NeuMF model changes as the number of recommended items (K) increases from 1 to 10. I calculated the average NDCG@K over 10 runs and also visualized the errorbars.

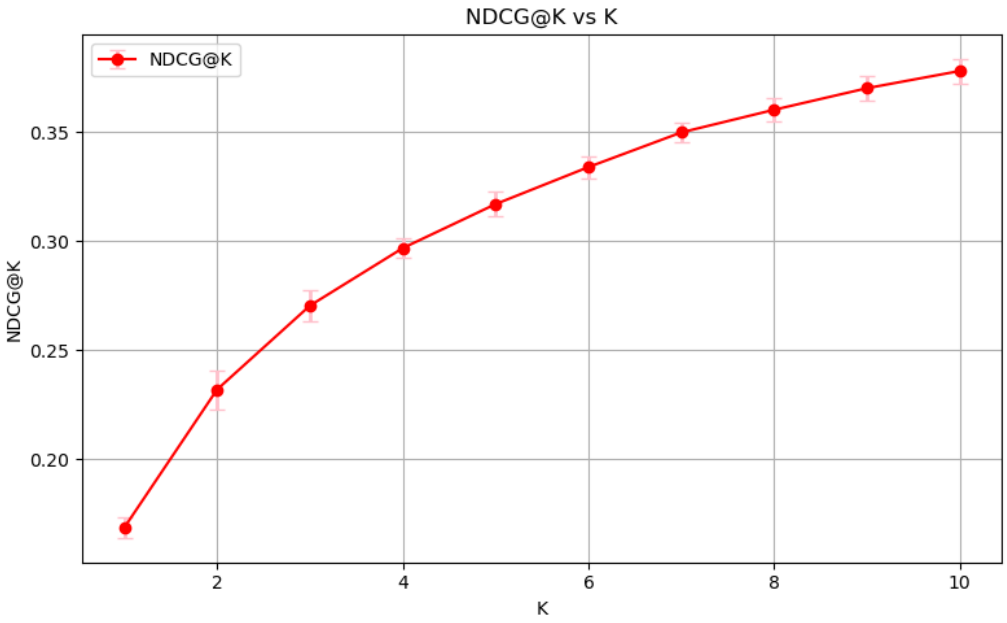
As K increases, NDCG@K also increases, which means that relevant items are not only present in the recommendation list but tend to appear near the top. However, the improvement becomes smaller after K

> 5, indicating that the model already ranks the most relevant items early in the list. The variation across runs is relatively small, showing consistent performance.

Table 39: NDCG@K *values for every run , mean and std*

K	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.162	0.168	0.163	0.165	0.165	0.173	0.172	0.17	0.179	0.168	0.1685	0.0049244
2	0.23	0.225	0.25	0.222	0.238	0.241	0.222	0.229	0.224	0.238	0.2319	0.0089605
3	0.27	0.265	0.263	0.276	0.266	0.288	0.266	0.273	0.274	0.263	0.2704	0.0073376
4	0.301	0.301	0.288	0.297	0.292	0.294	0.303	0.298	0.296	0.298	0.2968	0.0043081
5	0.322	0.312	0.324	0.312	0.318	0.325	0.32	0.308	0.318	0.311	0.317	0.0056214
6	0.33	0.342	0.331	0.327	0.334	0.336	0.33	0.34	0.33	0.34	0.334	0.0049598
7	0.356	0.353	0.351	0.34	0.35	0.351	0.352	0.352	0.349	0.345	0.3499	0.0042532
8	0.366	0.37	0.356	0.357	0.355	0.362	0.354	0.365	0.355	0.363	0.3603	0.0053301
9	0.358	0.377	0.373	0.373	0.369	0.375	0.377	0.367	0.37	0.363	0.3702	0.0058617
10	0.379	0.386	0.377	0.374	0.385	0.37	0.383	0.369	0.377	0.381	0.3781	0.0055758

Figure 13: NDCG@K vs K



7) HR@10 vs Number of Negative Samples

In this experiment, I tested how the number of negative samples affects the HR@10 of the NeuMF model. I increased the number of negatives from 1 to 10 and calculated the average HR@10 and std across 10 runs.

The results show that HR@10 gradually improves as the number of negatives increases up to around 6 or 7. The highest HR@10 was observed when using 7 negative samples. However, the improvement

compared to 5, 6, or even 10 negatives is very small. After a certain point, adding more negatives doesn't significantly help and may introduce noise instead.

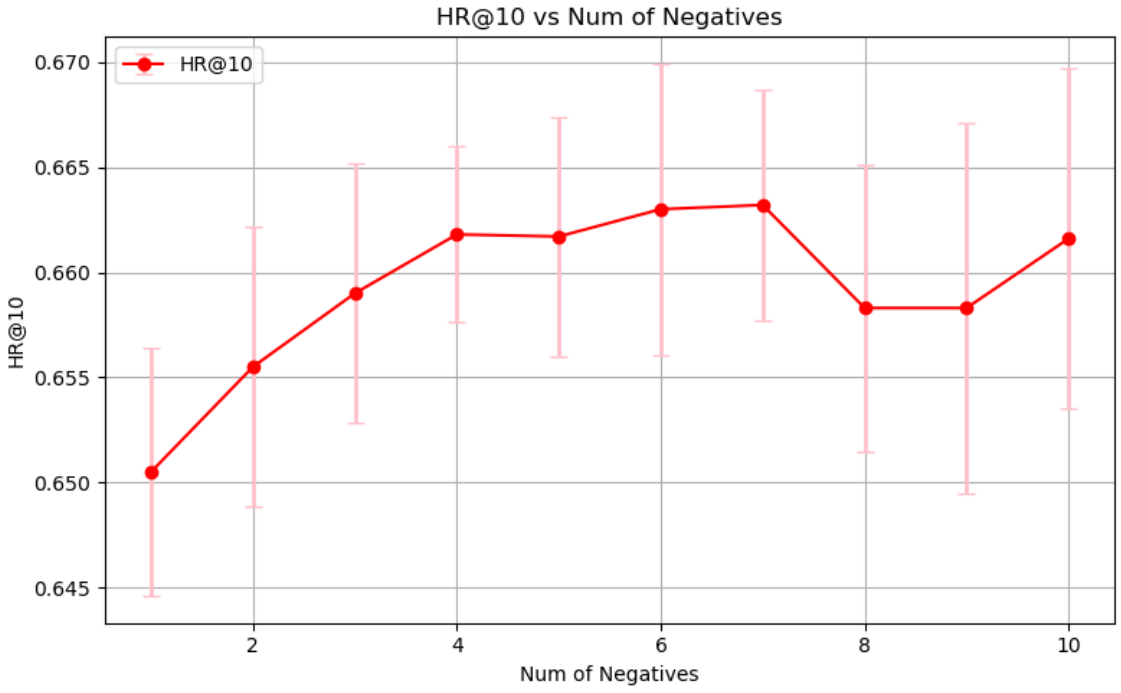
Additionally, the error bars are quite large for all settings, especially with more negatives, which suggests that the model's performance varies more between runs when too many negative samples are used.

In summary, a moderate number of negatives (around 4–7) seems to offer the best balance between performance and stability.

Table 40: HR@10 values for every run , mean and std for different numbers of negatives

Num Of Negatives	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.662	0.65	0.655	0.643	0.644	0.653	0.656	0.648	0.651	0.643	0.6505	0.0059203
2	0.655	0.646	0.664	0.656	0.652	0.654	0.646	0.666	0.653	0.663	0.6555	0.006637
3	0.67	0.655	0.653	0.655	0.652	0.653	0.668	0.662	0.659	0.663	0.659	0.0061644
4	0.666	0.654	0.657	0.662	0.661	0.663	0.662	0.663	0.66	0.67	0.6618	0.0041905
5	0.653	0.666	0.669	0.653	0.657	0.659	0.668	0.661	0.667	0.664	0.6617	0.0057105
6	0.681	0.656	0.657	0.657	0.66	0.665	0.66	0.663	0.665	0.666	0.663	0.0069282
7	0.662	0.661	0.662	0.662	0.677	0.667	0.655	0.663	0.664	0.659	0.6632	0.0054736
8	0.661	0.657	0.653	0.654	0.664	0.643	0.657	0.668	0.664	0.662	0.6583	0.0068125
9	0.662	0.668	0.663	0.657	0.659	0.674	0.648	0.654	0.642	0.656	0.6583	0.0087983
10	0.659	0.665	0.659	0.664	0.672	0.647	0.67	0.649	0.67	0.661	0.6616	0.0080771

Figure 14: HR@10 vs num of negatives



8) NDCG@10 vs Number of Negative Samples

In this experiment, I examined how the number of negative samples affects the NDCG@10 metric. I increased the number of negatives from 1 to 10 and measured the model's ranking quality over 10 runs.

The results show that NDCG@10 improves steadily up to around 4–6 negatives. After that, it stays mostly flat with very small changes. This suggests that the model learns to rank relevant items better when trained with a moderate number of negatives, but adding more doesn't make a big difference.

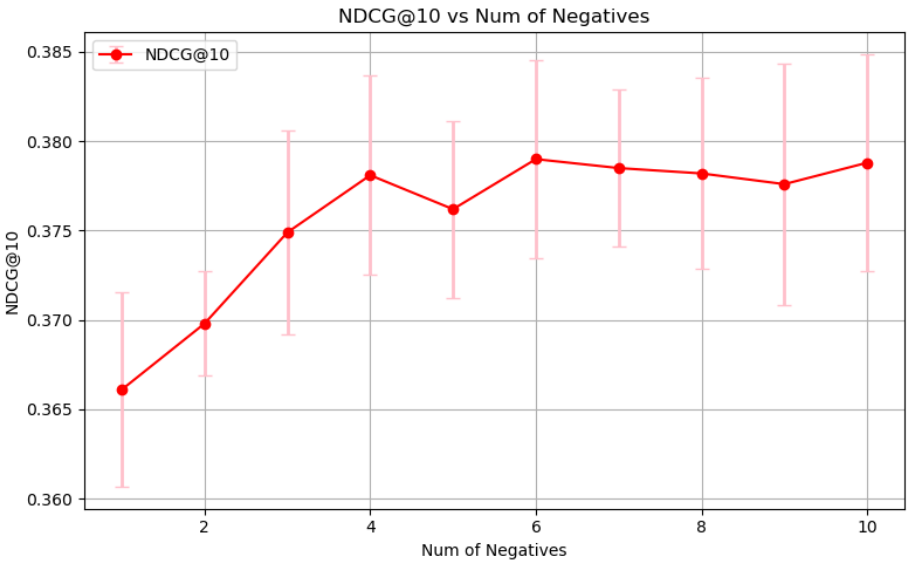
The highest NDCG@10 value was observed at 6 negatives, but the performance from 4 to 10 is very close. Also, the error bars are relatively large across all settings, which shows some variability across runs.

In general, using around 4 to 6 negative samples seems to be a good balance between ranking quality and training stability.

Table 41: NDCG@10 *values for every run , mean and std for different numbers of negatives*

Num Of Negatives	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.373	0.365	0.361	0.357	0.366	0.37	0.372	0.369	0.37	0.358	0.3661	0.0054489
2	0.369	0.367	0.373	0.373	0.363	0.372	0.37	0.369	0.37	0.372	0.3698	0.0029257
3	0.382	0.379	0.367	0.368	0.367	0.373	0.379	0.373	0.381	0.38	0.3749	0.0057175
4	0.385	0.37	0.371	0.382	0.382	0.374	0.377	0.379	0.374	0.387	0.3781	0.0055579
5	0.369	0.38	0.374	0.371	0.376	0.37	0.38	0.376	0.382	0.384	0.3762	0.0049558
6	0.39	0.374	0.377	0.378	0.373	0.382	0.37	0.381	0.384	0.381	0.379	0.0055678
7	0.375	0.369	0.381	0.382	0.383	0.384	0.379	0.38	0.374	0.378	0.3785	0.0044102
8	0.375	0.372	0.371	0.377	0.379	0.374	0.379	0.389	0.383	0.383	0.3782	0.0053254
9	0.378	0.375	0.38	0.383	0.382	0.388	0.369	0.367	0.37	0.384	0.3776	0.0067409
10	0.381	0.387	0.382	0.386	0.377	0.364	0.377	0.376	0.378	0.38	0.3788	0.0060795

Figure 15 : NDCG@10 vs num of negatives



9) Non-negative Matrix Factorization (NMF) and Its Impact on NDCG@10

In this experiment, I tested how the number of latent factors affects the NDCG@10 score of the NMF model. I ran the model with latent factors ranging from 1 to 30 (step 5), and calculated the average NDCG@10 over 10 runs for each setting.

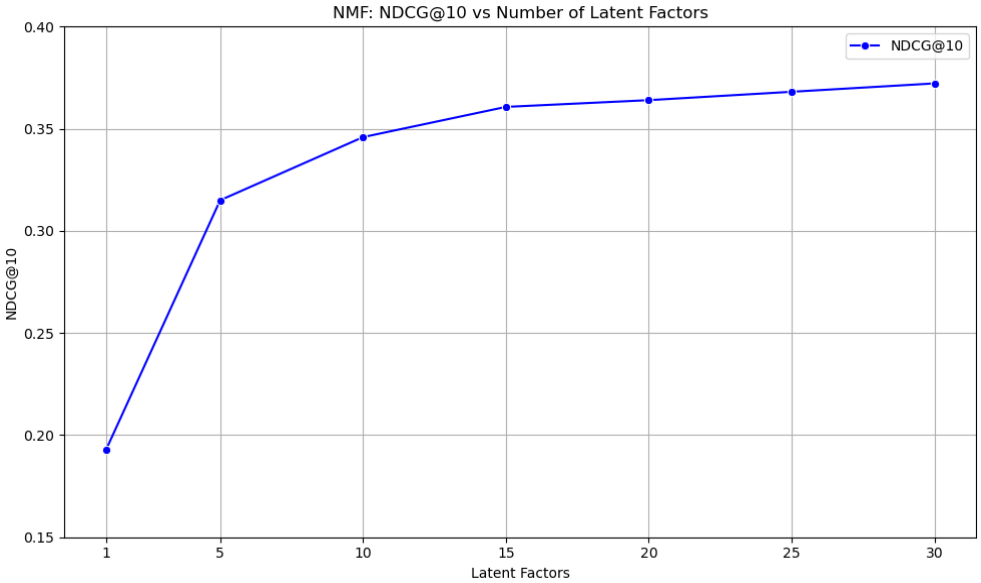
The results show that NDCG@10 increases as the number of factors grows. The improvement is especially fast in the beginning — going from 0.1926 at 1 factor to 0.3150 at just 5 factors. After that, the performance continues to improve but at a slower and more gradual rate, reaching 0.3722 at 30 factors.

This means that even a small number of latent factors can lead to a big boost in performance, while adding more offers smaller but steady gains.

Table 42 : NMF NDCG@10 *values for every run*

Latent Factors	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	Std
1	0.1926	0.1926	0.1926	0.1926	0.1926	0.1926	0.1926	0.1926	0.1926	0.1926	0.1926	2.776E-17
5	0.3135	0.3167	0.3157	0.3168	0.3166	0.3123	0.3123	0.317	0.3125	0.3169	0.315	0.0019965
10	0.349	0.3445	0.3449	0.3471	0.3477	0.3439	0.3463	0.3472	0.3452	0.3433	0.3459	0.0017421
15	0.3571	0.3647	0.3648	0.3643	0.3576	0.3575	0.3589	0.3593	0.3657	0.3576	0.3607	0.0034404
20	0.3636	0.3627	0.3635	0.3618	0.3633	0.3635	0.3642	0.3657	0.3681	0.3634	0.364	0.0016666
25	0.3651	0.3709	0.3664	0.3709	0.367	0.3698	0.3633	0.3664	0.3696	0.3714	0.3681	0.0026596
30	0.369	0.3643	0.3724	0.3678	0.3694	0.3749	0.3796	0.3688	0.3735	0.3818	0.3722	0.0051703

Figure 16: NDCG@10 vs latent factors



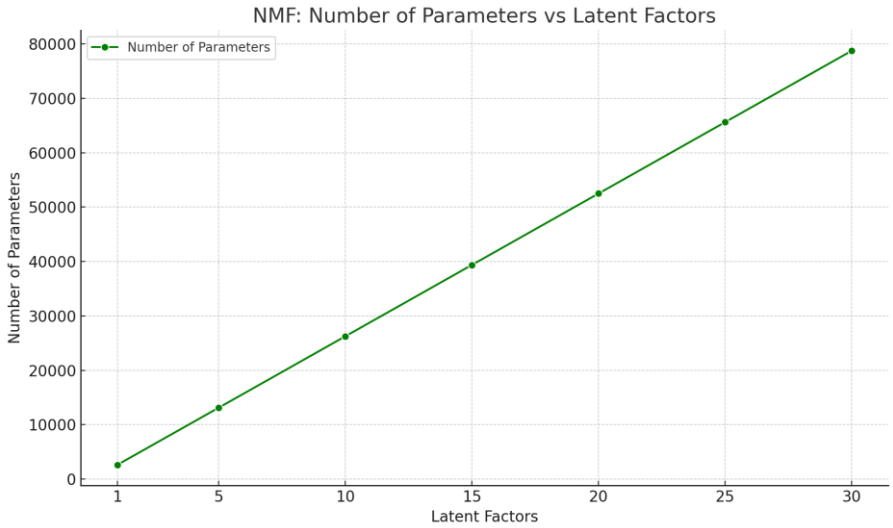
10) Effect of Latent Factor Size on NMF Model Complexity

In this experiment, I examined how the number of model parameters changes as the number of latent factors increases in the NMF model.

The results show a steep and almost linear growth in the number of parameters. Starting from just 2,625 parameters with 1 latent factor, the number increases rapidly to 78,750 at 30 latent factors. This means that every increase in the latent dimension significantly adds to the model size.

Although higher latent factors can improve recommendation performance (as seen in Question 9), they come with a clear cost in terms of memory and computation. Therefore, when deploying NMF in a real-world system, it's important to balance between accuracy and efficiency.

Figure 17: Number of parameters vs latent factors



11) Comparison of NeuMF and NMF in Terms of Accuracy and Model Size

For this question, I ran both NeuMF and NMF again (using the dataset split from Question 1) with their best parameter settings determined.

For NeuMF:

Batch size: 256 Learning rate: 0.001 Number of Negatives: 4 Number of MLP layers: 3, Number of Factors: 32 Iterations: 20

For NMF:

Iterations: 2000, Number of Latent Factors: 30 ,Random initialization

Based on the 10 runs, NeuMF gave slightly better results than NMF in both HR@10 and NDCG@10. NeuMF had an average HR@10 of 0.665 compared to 0.652 for NMF, and NDCG@10 of 0.3781 compared to 0.3722.

The results were also more stable for NeuMF (lower standard deviation in HR). However, NeuMF uses a lot more parameters (463k vs 78k), so it's heavier.

In short, NeuMF is more accurate but NMF is much lighter and still performs well.

Table 43: NeuMF vs NMF

	NeuMF		NMF	
Run	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**
1	**0.672**	**0.379**	**0.6532**	**0.369**
2	**0.661**	**0.386**	**0.649**	**0.3643**
3	**0.672**	**0.377**	**0.6511**	**0.3724**
4	**0.662**	**0.374**	**0.6416**	**0.3678**
5	**0.665**	**0.385**	**0.6522**	**0.3694**
6	**0.662**	**0.37**	**0.6532**	**0.3749**
7	**0.662**	**0.383**	**0.6596**	**0.3796**
8	**0.668**	**0.369**	**0.6469**	**0.3688**
9	**0.665**	**0.377**	**0.6554**	**0.3735**
10	**0.659**	**0.381**	**0.6575**	**0.3818**
Mean	**0.665**	**0.3781**	**0.652**	**0.3722**
Std	**0.004**	**0.0055758**	**0.0052**	**0.0055**
Weights	**463297**		**78250**	

Figure 18 : Comparison of NeuMF and NFM

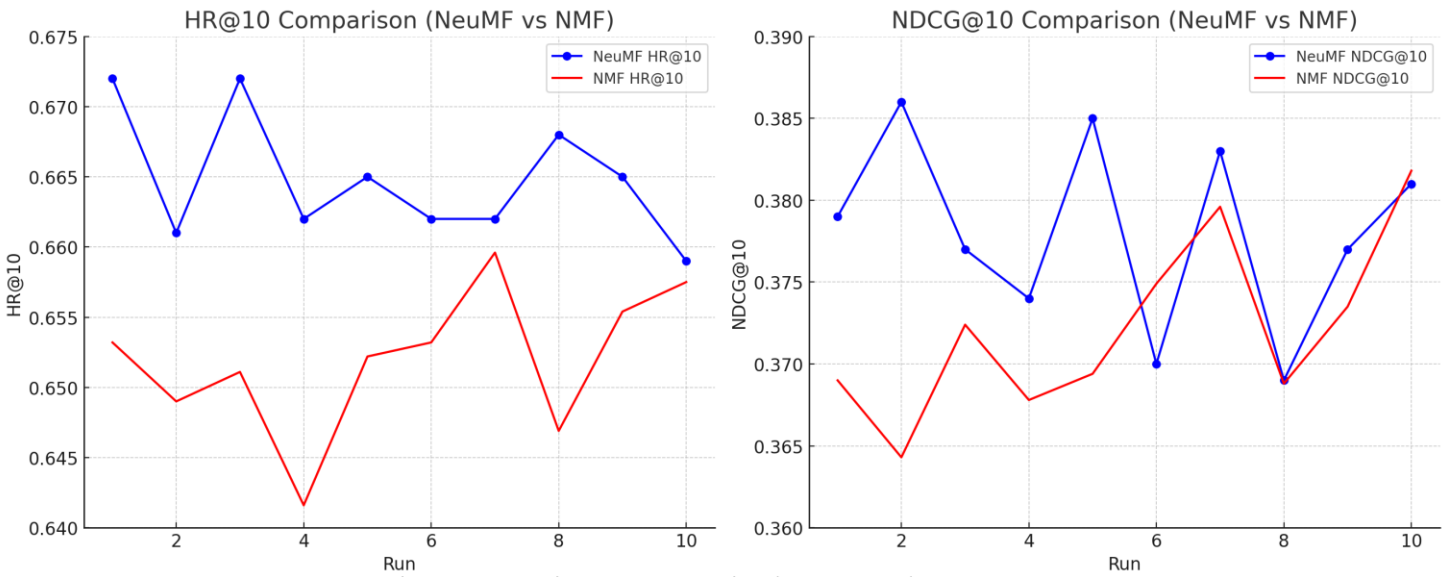


Figure: HR@10 and NDCG@10 Comparison (NeuMF vs NMF) Across 10 Runs

12) Applying Knowledge Distillation to NeuMF for Model Compression

I selected three distinct knowledge distillation techniques: Logits Matching, Soft Labels, and Feature Matching. The goal was to explore different types of knowledge transfer from the teacher model (NeuMF-end) to lighter student models.

I first chose Logits Matching because it is a simple and commonly used baseline. It directly aligns the raw output values (logits) of the student and teacher using Mean Squared Error (MSE). This method is easy to implement and serves as a starting point for comparing more advanced techniques.

Next, I implemented Soft Labels Distillation, inspired by Hinton et al. This method uses a temperature-scaled sigmoid to create soft targets from the teacher, capturing more nuanced information about the teacher's confidence across possible outputs. It provides richer supervision and often leads to better generalization in the student.

Lastly, I experimented with Feature Matching, where the student is trained to mimic the intermediate feature representations of the teacher's. This approach allows the student to learn internal representations similar to the teacher, not just output behavior.

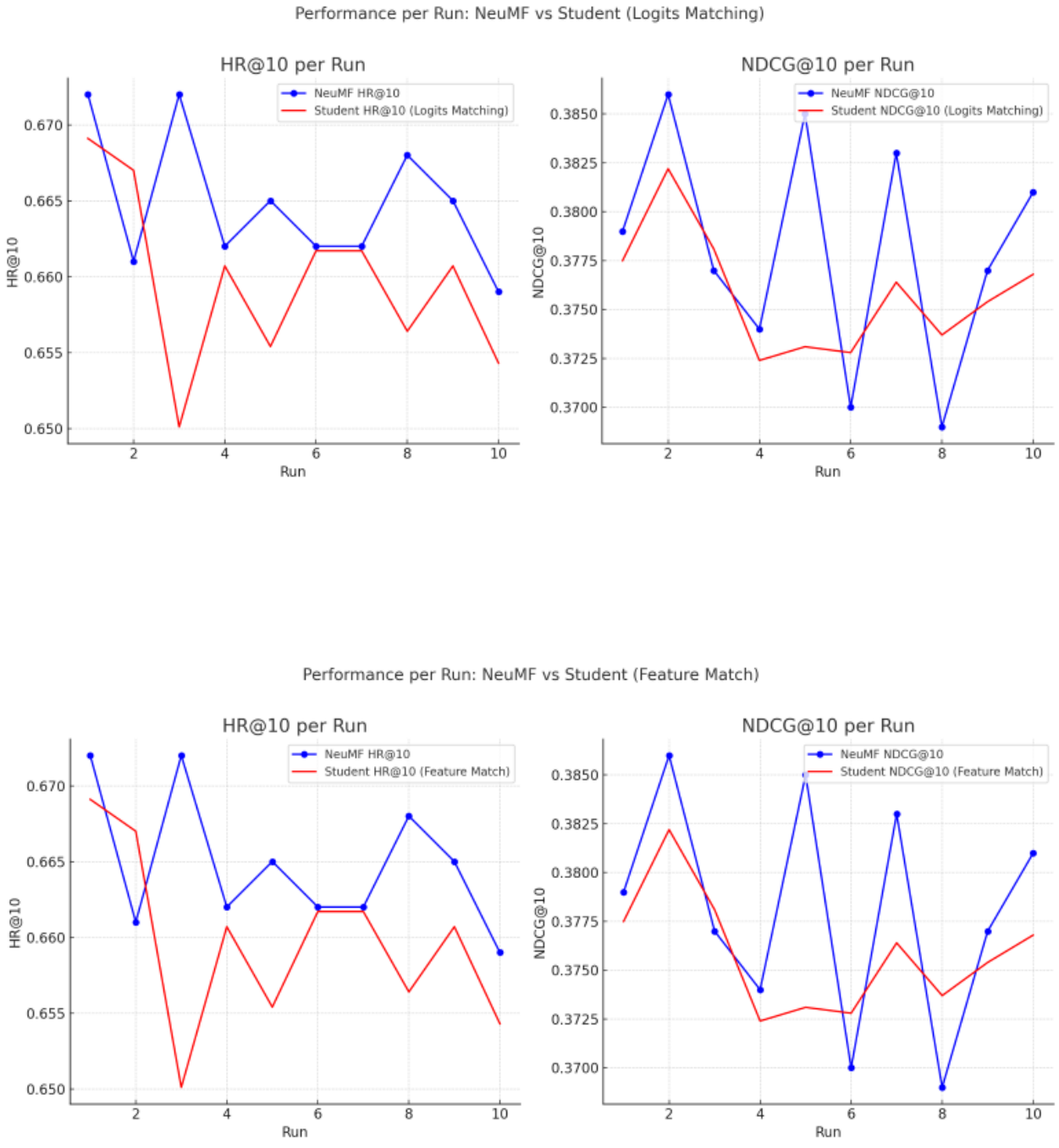
Overall, these three techniques were chosen to represent a balance between output-level, probability-level, and feature-level knowledge transfer, enabling a meaningful comparison across different distillation strategies.

For all knowledge distillation experiments, I used a fully trained NeuMF-end model as the teacher.

For each distillation technique, I designed a student model with reduced capacity:

- Embedding factor size: 8 (instead of 32)
- Number of MLP layers: 1 (instead of 3)

Figure 19: Knowledge distillation results



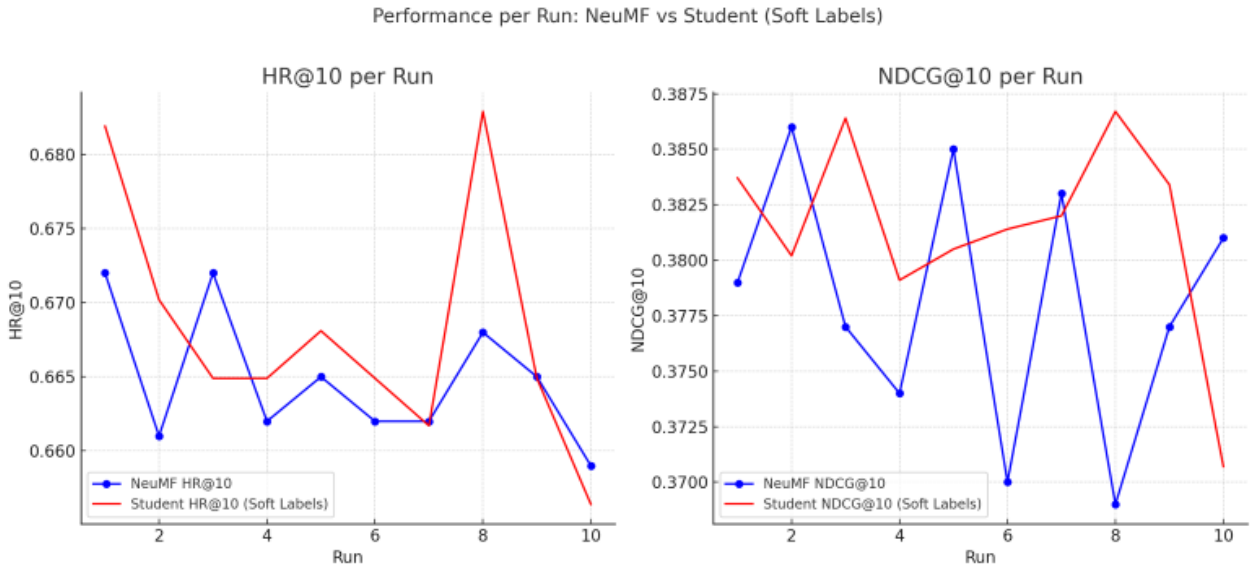


Table 44: Knowledge distillation results(best epoch for evet run)

	NeuMF		Soft Labels		Logits		Feature Match	
Run	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
1	0.672	0.379	0.6819	0.3837	0.6691	0.3775	0.6691	0.3758
2	0.661	0.386	0.6702	0.3802	0.667	0.3822	0.667	0.3791
3	0.672	0.377	0.6649	0.3864	0.6501	0.3781	0.6554	0.3716
4	0.662	0.374	0.6649	0.3791	0.6607	0.3724	0.6522	0.371
5	0.665	0.385	0.6681	0.3805	0.6554	0.3731	0.6638	0.3798
6	0.662	0.37	0.6649	0.3814	0.6617	0.3728	0.6702	0.377
7	0.662	0.383	0.6617	0.382	0.6617	0.3764	0.6543	0.373
8	0.668	0.369	0.6829	0.3867	0.6564	0.3737	0.6649	0.3753
9	0.665	0.377	0.6649	0.3834	0.6607	0.3754	0.666	0.3766
10	0.659	0.381	0.6564	0.3707	0.6543	0.3768	0.6522	0.3705
Mean	0.665	0.3781	0.6681	0.3814	0.6597	0.3758	0.6615	0.375
Std	0.004	0.0055758	0.0079	0.0043	0.0055	0.0029	0.0068	0.0031

All student models achieved performance close to the NeuMF teacher. Among the three techniques, Soft Labels distillation performed best, slightly outperforming the teacher in average HR@10 and NDCG@10, and showing more stable results across runs. Logits Matching and Feature Matching had similar performance, confirming they extract comparable information in this setup. Overall, the results demonstrate that knowledge distillation can significantly compress the model without a noticeable drop in recommendation accuracy.