HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

PROJECT REPORT

Movie Recommendation System (Phase 1 Experiment Report)

Course: Web Mining (IT4868E)

Supervisor: Ph.D. Nguyen Kiem Hieu

Nguyen Viet Anh	20225434
Hoang Trung Khai	20225502
Trinh Duy Phong	20220065
Luu Thien Viet Cuong	20225477
Do Dinh Hoang	20225445

Hanoi, December 28, 2025

Contents

Acknowledgement

We extend our gratitude to Ph.D. Nguyen Kiem Hieu for guidance. We also thank the open-source authors of DCNv3 and DeepFM implementations that informed our design and evaluation.

Abstract

This report documents an offline experimental comparison of three recommender architectures: a Multilayer Perceptron (MLP) baseline, DeepFM, and DCNv3. Experiments are conducted on MovieLens-1M with two dataset variants (ratings-only and metadata-augmented). We evaluate models using both ranking (top-K) metrics and classification metrics (AUC, LogLoss, etc.), report learning curves and hyper-parameters, and present an ablation study quantifying the benefit of side information for DCNv3.

Source code for this project is available at `https://github.com/Itvc0110/webmining`.

Chapter 1: Introduction

1.1 Motivation

Effective recommender systems must model interactions between users and items. Classical embedding-based models capture basic correlations but struggle to represent complex, high-order feature interactions. Architectures such as DeepFM introduce an inductive bias for pairwise interactions, while recent cross-network models (DCNv3) explicitly model higher-order interactions and apply noise-filtering mechanisms. We evaluate how these design choices translate into measurable gains under controlled experimental settings.

1.2 Contributions

- A controlled ratings-only comparison of MLP, DeepFM, and DCNv3 to isolate architectural advantages.
- An ablation study that measures the incremental value of user/item metadata for DCNv3.
- Complete experimental documentation: dataset, preprocessing, model details, loss functions, evaluation protocols, learning curves, and result tables.

Chapter 2: Dataset

2.1 Link to dataset and split

The origninal dataset used is the MovieLens-1M. To ensure fair and interpretable comparisons, we prepare two dataset variants that share identical train/validation/test splits:

- **DCNv3 dataset (with user and item metadata):** `https://drive.google.com/drive/folders/1S69tMoM9Aq1DB1B3cgVcINJM-lj_Uv9D`
- **DeepFM and MLP dataset (ratings-only):** `https://drive.google.com/drive/folders/1lIXD206ilaXIwK-NET_v1f0uPy8ij96P?usp=sharing`

This two-variant design isolates the effects of feature availability from pure architectural capability.

2.2 Data preprocessing

This section describes the feature construction, label definition, and data preparation procedures used for all evaluated models. While all methods share the same raw MovieLens-1M interaction data and an identical train/dev/test split, the final input representations differ to accommodate architectural requirements.

Interaction data and labeling. Each interaction is represented as a user–item pair associated with an explicit rating in the range $\{1, \ldots, 5\}$. To align the task with implicit-feedback and CTR-style recommendation settings, explicit ratings are converted into binary labels via threshold-based binarization: interactions with ratings strictly greater than 3 are treated as positive feedback ($y = 1$), while ratings less than or equal to 3 are treated as negative feedback ($y = 0$). This labeling strategy is widely adopted in the recommendation literature and enables unified evaluation using both classification-oriented metrics (e.g., AUC, LogLoss) and ranking-based metrics (e.g., NDCG@K, MAP@K).

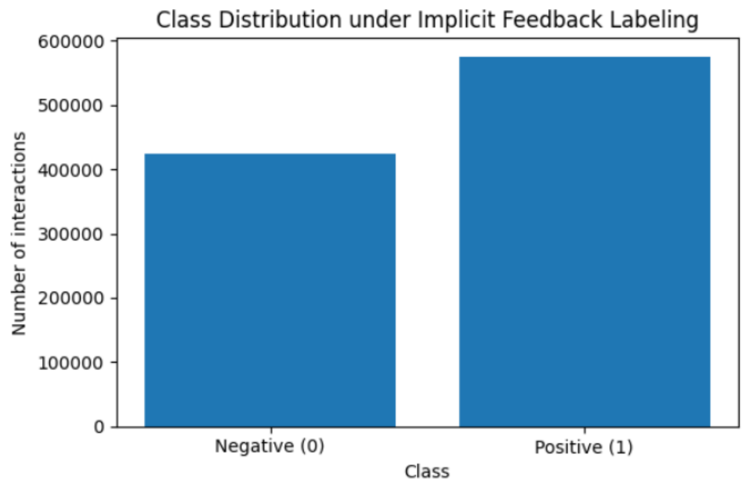


Figure 1: Distribution of implicit feedback labels derived from MovieLens-1M ratings.

Feature construction for DeepFM and MLP. For DeepFM and the MLP baseline we use the ID-only configuration:

- **User ID** (categorical, sparse)
- **Item ID** (categorical, sparse)

IDs are remapped to contiguous zero-based indices and encoded with embedding layers. This minimal feature set is widely used as a baseline configuration in the literature.

Feature construction for DCNv3. DCNv3 is evaluated in two configurations (see Section 3.6): ID-only and metadata-augmented. For the metadata-augmented setting we include:

- **Sparse categorical features:** User ID, Item ID, discretized user age, user gender, user occupation.
- **Dense numerical features:** Multi-hot movie genre vector (18 dimensions).

User age is binned to reduce sparsity; gender and occupation are encoded as categorical fields; genres are multi-hot encoded. Sparse features are embedded and dense features are projected to the shared embedding dimensionality where required.

Cold-start consideration. Metadata provides structured, reusable signals that help generalization and mitigate cold-start effects for low-frequency users/items. DCNv3's cross modules can directly combine demographic and content attributes to infer preferences for users/items with few interactions.

Negative sampling. For pointwise classification training we retain all interactions after binarization. For ranking evaluation, unobserved user–item pairs are treated as negatives in candidate generation. No synthetic negative sampling is performed during training for Phase 1; this keeps training consistent across models.

Filtering and consistency. All user and item identifiers are remapped to contiguous indices. We do not apply additional k-core filtering beyond MovieLens-1M's inherent preprocessing. The same train/dev/test splits are applied across all experiments to avoid split-based confounding effects.

Chapter 3: Methods

This chapter presents the evaluated models, their architectures, and the training objectives used to optimize them.

3.1 Baseline: MLP

The Multilayer Perceptron (MLP) baseline follows a canonical embedding-to-MLP architecture. For a mini-batch of users u and items i:

$$\mathbf{e}_u = \mathrm{Embed}_U(u), \quad \mathbf{e}_i = \mathrm{Embed}_I(i), \quad \mathbf{x} = \mathrm{Concat}(\mathbf{e}_u, \mathbf{e}_i).$$

The concatenated embedding $\mathbf{x}$ is passed through a sequence of fully-connected layers with ReLU activation, batch normalization and dropout; the final scalar logit is passed through a sigmoid $\sigma(\cdot)$ to produce a probability $\hat{p}$. The MLP is trained with binary cross-entropy (see Section 3.4).

3.2 DeepFM

DeepFM is a unified recommendation architecture designed to jointly learn low-order and high-order feature interactions in an end-to-end manner. It combines a Factorization Machine (FM) component for modeling second-order (pairwise) feature interactions with a deep neural network (DNN) component for capturing higher-order and non-linear interaction patterns. A key design principle of DeepFM is that both components share the same feature embedding layer, enabling consistent representation learning without the need for manual feature engineering.

Model formulation. Given an input instance represented by multiple feature fields, each field j is mapped to a dense embedding vector $\mathbf{e}_j \in \mathbb{R}^d$. The overall prediction logit is obtained by summing three components: a linear term, an FM-based second-order interaction term, and a DNN-based high-order interaction term.

The linear component models first-order effects:

$$\mathrm{linear} = \sum_j w_j(x_j),$$

where w_j denotes the learnable weight associated with feature x_j.

The second-order interaction term is computed using the standard FM formulation:

$$\mathrm{second\_order} = \tfrac{1}{2}\left(\Big\| \sum_j \mathbf{e}_j \Big\|^2 - \sum_j \|\mathbf{e}_j\|^2\right),$$

which efficiently captures all pairwise feature interactions without explicitly enumerating them.

In parallel, the deep component takes the concatenation of all feature embeddings as input and models higher-order interactions through multiple fully connected layers:

$$\mathrm{dnn\_out} = \mathrm{DNN}\big(\mathrm{Concat}(\mathbf{e}_j)\big).$$

The final prediction logit is given by:

$$\mathrm{logit} = \mathrm{linear} + \mathrm{second\_order} + \mathrm{dnn\_out} + b, \quad \hat{p} = \sigma(\mathrm{logit}),$$

where b is a global bias term and $\sigma(\cdot)$ denotes the sigmoid function.

Learning objective. DeepFM is typically trained for binary classification tasks such as click-through rate (CTR) prediction using the log loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)\Big],$$

where $y_i \in \{0,1\}$ denotes the ground-truth label and $\hat{p}_i$ is the predicted probability.

Design advantages. The FM component introduces a strong inductive bias toward learning meaningful pairwise interactions, which is particularly effective in sparse recommendation settings. Meanwhile, the DNN component captures residual higher-order and non-linear interactions that cannot be represented by second-order terms alone. By sharing embeddings between the FM and DNN branches, DeepFM avoids redundant parameterization and allows gradients from both low-order and high-order objectives to jointly shape the learned representations.

Compared to earlier hybrid models such as Wide&Deep, DeepFM eliminates the need for manually crafted cross features, achieving competitive or superior performance with significantly reduced feature engineering effort.

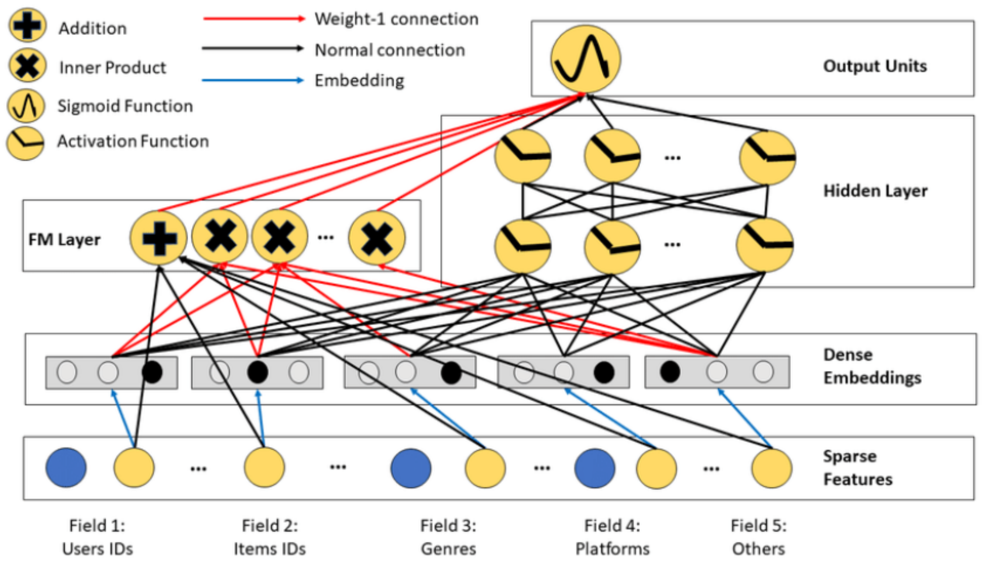


Figure 2: DeepFM architecture.

3.3 DCNv3

DCNv3 is designed to explicitly and efficiently model structured high-order feature interactions by decomposing cross-feature learning into two complementary modules: an Exponential Cross Network (ECN) and a Linear Cross Network (LCN). This dual-path design enables DCNv3 to simultaneously capture both complex non-linear interactions and structured low-order dependencies, while maintaining favorable parameter efficiency and optimization stability.

Input representation. The model input consists of embedded sparse (categorical) features and projected dense (continuous) features. Specifically, each categorical field is mapped to a dense embedding vector, while continuous features are first linearly projected into the same embedding space. All feature representations are then concatenated into a single input vector:

$$\mathbf{x}_0 = \mathrm{Concat}(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_F),$$

where $\mathbf{e}_j \in \mathbb{R}^d$ denotes the embedding or projection of the j-th feature field.

Exponential Cross Network (ECN). The ECN branch is responsible for modeling expressive, high-order feature interactions whose effective polynomial order grows exponentially with network depth. At each cross layer ℓ, the ECN computes a set of low-rank projections of the current representation $\mathbf{x}_\ell$ and applies multiplicative interactions with the original input $\mathbf{x}_0$:

$$\mathbf{z}_\ell^{(k)} = \mathbf{U}_\ell^{(k)}\mathbf{x}_\ell, \quad \mathbf{m}_\ell^{(k)} = \mathbf{V}_\ell^{(k)}\mathbf{x}_0,$$

where $\mathbf{U}_\ell^{(k)}$ and $\mathbf{V}_\ell^{(k)}$ denote learnable projection matrices for the k-th interaction head.

A gating or masking function is then applied to regulate the strength of each interaction:

$$\mathbf{g}_\ell^{(k)} = \sigma\big(\mathbf{W}_\ell^{(k)}\mathbf{x}_\ell\big),$$

and the gated interactions are aggregated and added back to the representation via a residual update:

$$\mathbf{x}_{\ell+1} = \mathbf{x}_\ell + \sum_k \mathbf{g}_\ell^{(k)} \odot \big(\mathbf{z}_\ell^{(k)} \odot \mathbf{m}_\ell^{(k)}\big).$$

Through repeated multiplicative residual updates, ECN composes interactions of increasing order across layers, allowing DCNv3 to represent complex cross-feature dependencies without explicitly enumerating polynomial terms.

Linear Cross Network (LCN). In parallel, the LCN branch focuses on modeling structured low-order feature interactions using a more linearized cross formulation with residual connections. Each LCN layer updates its representation as:

$$\mathbf{s}_{\ell+1} = \mathbf{s}_\ell + \mathbf{A}_\ell\mathbf{s}_\ell \odot \mathbf{x}_0,$$

where $\mathbf{A}_\ell$ is a learnable weight matrix. This design preserves the inductive bias of classical cross networks while improving training stability and efficiency.

The LCN branch excels at capturing strong, interpretable interactions that are well-approximated by low-order cross terms, complementing the expressive capacity of ECN.

Branch aggregation and prediction. Each branch produces a branch-specific logit:

$$y_d = f_{\text{ECN}}(\mathbf{x}_L), \quad y_s = f_{\text{LCN}}(\mathbf{s}_L),$$

where $f_{\text{ECN}}(\cdot)$ and $f_{\text{LCN}}(\cdot)$ denote lightweight output layers. The final prediction is obtained by aggregating these logits:

$$\text{logit} = y_d + y_s + b, \quad \hat{p} = \sigma(\text{logit}),$$

with b being a global bias term.

Design rationale. By explicitly separating high-order and low-order interaction modeling into ECN and LCN branches, DCNv3 avoids the optimization difficulties commonly observed in deep polynomial networks. The use of low-rank projections, gating mechanisms, and residual connections allows the model to scale to high-dimensional, sparse feature spaces while maintaining strong generalization. This structured interaction modeling directly translates into improved ranking and classification performance, particularly in recommendation settings where complex feature dependencies are critical.

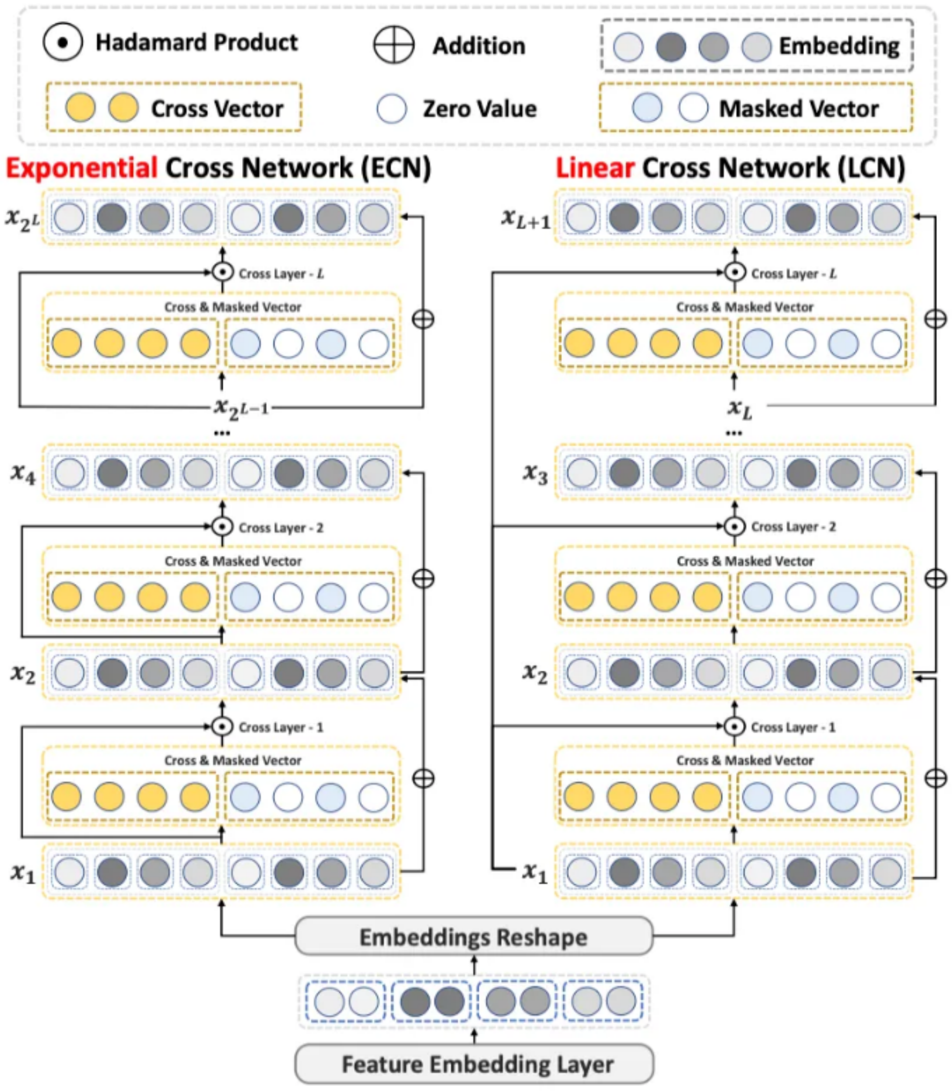


Figure 3: DCNv3 architecture.

3.4 Loss functions and training objectives

Binary Cross-Entropy (BCE). For pointwise CTR/classification training we use binary cross-entropy (LogLoss). For N training examples with labels $y_i \in \{0, 1\}$ and predicted probabilities $\hat{p}_i$:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\big[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)\big].$$

Tri-BCE (DCNv3). DCNv3 produces three outputs: the ensemble prediction $\hat{p}$, the deep-branch output $\hat{p}_d$, and the shallow-branch output $\hat{p}_s$. We apply BCE to each output and aggregate with weights:

$$\mathcal{L}_{\text{TriBCE}} = \lambda_0 \mathcal{L}_{\text{BCE}}(\hat{p}, y) + \lambda_1 \mathcal{L}_{\text{BCE}}(\hat{p}_d, y) + \lambda_2 \mathcal{L}_{\text{BCE}}(\hat{p}_s, y),$$

where $\lambda_0, \lambda_1, \lambda_2 \geq 0$ and $\sum \lambda_i = 1$ (or normalized appropriately) as configured.

Pairwise ranking loss (optional). When pairwise ranking training is employed, the BPR loss may be used:

$$\mathcal{L}_{\text{BPR}} = - \sum_{(u,i,j)} \log \sigma\big(\hat{y}_{u,i} - \hat{y}_{u,j}\big),$$

where (u, i, j) denotes a user u, a positive item i, and a sampled negative item j.

3.5 Feature Ablation: Metadata vs ID-only

To quantify the contribution of auxiliary user and item metadata, we evaluate two DCNv3 variants under identical optimization settings:

- **DCNv3 (ID-only):** user ID and item ID only.
- **DCNv3 (with metadata):** ID features plus user demographics (age bin, gender, occupation) and multi-hot movie genres.

Both variants share embedding dimensions, cross-layer count, optimizer settings and loss function. This controlled ablation isolates the effect of feature augmentation.

Architectural and theoretical implications. Metadata introduces structured, reusable signals enabling cross layers to learn semantically meaningful interactions such as (age bin $\times$ genre) or (occupation $\times$ movie category). These interactions are shared across many users/items, improving generalization and alleviating cold-start limitations of ID-only embeddings.

Expected empirical outcomes. We anticipate that DCNv3 with metadata will yield higher ranking metrics (NDCG@10, MAP@10) and improved calibration (higher AUC, lower LogLoss) relative to the ID-only variant.

Chapter 4: Evaluation Metrics

We evaluate all models under a unified and reproducible protocol using both classification-level metrics and ranking-based metrics. Classification metrics assess the quality of predicted interaction probabilities, which is essential for CTR-oriented tasks, while ranking metrics evaluate the model's ability to retrieve relevant items from a large candidate space, which is the core objective of recommendation systems.

All experiments use deterministic train/validation/test splits with fixed random seeds. During ranking evaluation, candidate item sets for each user exclude items observed in the training set in order to prevent information leakage and to simulate realistic recommendation scenarios.

4.1 Classification Metrics

Let $\{(y_i, \hat{p}_i)\}_{i=1}^{N}$ denote the ground-truth binary interaction labels and predicted probabilities on the test set, where $y_i \in \{0, 1\}$ and $\hat{p}_i \in [0, 1]$.

- **LogLoss (Binary Cross-Entropy).**

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)\right].$$

  LogLoss measures how well predicted probabilities align with true interaction outcomes. Lower values indicate better calibrated predictions and are particularly important for CTR estimation tasks.

- **Area Under the ROC Curve (AUC).**

$$\text{AUC} = \mathbb{P}\left(\hat{p}_{i^+} > \hat{p}_{i^-}\right),$$

  where i^+ and i^- denote positive and negative samples, respectively. AUC evaluates the model's ability to correctly rank positive interactions above negative ones, independent of a specific decision threshold. In recommender systems, a higher AUC implies stronger global discrimination between preferred and non-preferred items.

- **Accuracy.**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

  Accuracy measures the proportion of correctly classified interactions. While intuitive, it can be misleading under class imbalance and is therefore reported together with other metrics.

- **Precision.**

$$\text{Precision} = \frac{TP}{TP + FP}.$$

  Precision quantifies the reliability of positive predictions, indicating how many recommended or predicted interactions are truly relevant.

- **Recall.**

$$\text{Recall} = \frac{TP}{TP + FN}.$$

  Recall measures the model's ability to detect relevant items among the entire item space. In recommendation settings, high recall indicates strong coverage of user-preferred items.

- **F1-score.**

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

  The F1-score balances precision and recall, providing a single measure that reflects the trade-off between recommendation accuracy and coverage.

- **Precision–Recall AUC (PR-AUC).** PR-AUC summarizes the trade-off between precision and recall across different thresholds and is especially informative in imbalanced datasets, where negative interactions dominate.

- **Matthews Correlation Coefficient (MCC).**

$$\mathrm{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

  MCC provides a balanced evaluation of binary classification performance, taking all four confusion matrix components into account and remaining robust under class imbalance.

4.2 Ranking metrics (Top-K).

Ranking metrics evaluate how effectively a model retrieves relevant items within the top-K positions of a ranked recommendation list. For each user u, let $\mathcal{R}_u^K$ denote the ranked list of top-K recommended items, and let $\mathcal{T}_u$ be the set of relevant (positive) test items.

- **Precision@K.**

$$\mathrm{Precision@K} = \frac{1}{K} \sum_{i \in \mathcal{R}_u^K} \mathbb{I}(i \in \mathcal{T}_u).$$

  Precision@K measures the proportion of recommended items in the top-K list that are relevant, reflecting recommendation accuracy at the presentation level.

- **Recall@K.**

$$\mathrm{Recall@K} = \frac{|\mathcal{R}_u^K \cap \mathcal{T}_u|}{|\mathcal{T}_u|}.$$

  Recall@K evaluates how well the model recovers a user's preferred items from the entire item space. Higher recall indicates stronger retrieval capability.

- **Hit Rate@K.**

$$\mathrm{Hit@K} = \mathbb{I}\left(|\mathcal{R}_u^K \cap \mathcal{T}_u| > 0\right).$$

  Hit Rate@K measures whether at least one relevant item appears in the top-K recommendations, capturing a minimal success criterion for recommendation quality.

- **Mean Reciprocal Rank (MRR@K).**

$$\mathrm{MRR@K} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{\mathrm{rank}_u},$$

  where rank_u is the position of the first relevant item in $\mathcal{R}_u^K$. MRR emphasizes early ranking of relevant items, which is crucial for user satisfaction in real-world systems.

- **Mean Average Precision (MAP@K).**

$$\mathrm{AP@K}(u) = \frac{1}{\min(|\mathcal{T}_u|, K)} \sum_{k=1}^{K} \mathrm{Precision@k} \cdot \mathbb{I}(i_k \in \mathcal{T}_u),$$

$$\mathrm{MAP@K} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathrm{AP@K}(u).$$

  MAP@K captures both ranking correctness and position sensitivity across all relevant items, providing a comprehensive ranking quality measure.

- **Normalized Discounted Cumulative Gain (NDCG@K).**

$$\mathrm{DCG@K} = \sum_{k=1}^{K} \frac{2^{\mathrm{rel}_k} - 1}{\log_2(k+1)}, \quad \mathrm{IDCG@K} = \sum_{k=1}^{\min(|\mathcal{T}_u|, K)} \frac{2^{\mathrm{rel}_k^{ideal}} - 1}{\log_2(k+1)}$$

$$\mathrm{NDCG@K} = \frac{\mathrm{DCG@K}}{\mathrm{IDCG@K}}$$

NDCG@K accounts for both the relevance and rank position of recommended items, where rel_k^{ideal} represents the relevance of items in the perfect ranking (sorted by relevance in descending order).

4.3 Evaluation procedure.

For each test user, candidate item sets are constructed by excluding all items observed during training. Models produce relevance scores for all candidates, which are sorted to obtain top-K recommendation lists. Ranking metrics are averaged across users, while classification metrics are computed globally over all test interactions. This protocol ensures a fair and comprehensive comparison across models and learning paradigms.

Chapter 5: Experimental Setup

5.1 Implementation details

All models are implemented in `PyTorch`, using a unified training and evaluation pipeline to ensure consistency across experimental settings. The implementations share common data loaders, optimization procedures, and evaluation routines, differing only in model-specific architectural components. This design minimizes confounding factors and allows performance differences to be attributed primarily to modeling choices.

Training is performed using the Adam optimizer, which adapts learning rates on a per-parameter basis and is well suited for sparse and high-dimensional recommendation data. An initial learning rate of 1×10^{-3} is used for all models, and a learning rate scheduler is applied to reduce the step size when the validation loss plateaus. This scheduling strategy improves training stability and prevents premature convergence to suboptimal solutions.

To ensure experimental reproducibility, all sources of randomness are explicitly controlled. Random seeds are fixed for Python's built-in `random` module, NumPy, and PyTorch. In addition, deterministic computation modes are enabled in PyTorch where applicable, ensuring that repeated runs with identical configurations yield consistent results.

All experiments are conducted on a GPU-equipped machine using NVIDIA T4 GPUs. GPU acceleration is leveraged for both training and inference to handle large embedding tables and batch-wise ranking evaluation efficiently. During training, mini-batch processing is used to balance computational efficiency and memory usage. Model checkpoints are saved based on validation performance, and early stopping is employed to prevent overfitting.

Evaluation is performed in a strictly offline manner. After training, the best-performing checkpoint on the validation set is selected and evaluated on the held-out test set. Both classification-level metrics and top-K ranking metrics are computed using the same evaluation code for all models. For ranking evaluation, user-specific candidate item sets are constructed by excluding items observed in the training set, and model scores are used to generate ranked recommendation lists.

Overall, this implementation setup emphasizes reproducibility, fairness, and computational efficiency, providing a reliable basis for comparing MLP, DeepFM, and DCNv3 under controlled experimental conditions.

5.2 Hyper-parameters

The example hyper-parameters used in all current experiments are summarized in Table 1. These values will be tuned in the final version of the reports to ensure a controlled comparison with the models at their best setting.

Table 1: Hyper-parameters used across experiments.

Parameter	Value
Embedding dimension	64
Learning rate	1×10^{-3} (Adam)
Batch size	1024
Dropout (DNN/MLP)	0.2
Weight decay	1e-6
Number of training epochs	10
Early stopping patience	3 epochs
MLP hidden layers	[256, 128]
DeepFM hidden layers	[256, 128]
DCNv3: number of cross layers	3
DCNv3: cross projection dim	32
DCNv3: tri-BCE weights $(\lambda_0, \lambda_1, \lambda_2)$	(0.6, 0.2, 0.2)
Random seed	42
Evaluation top-K	$K = 10$

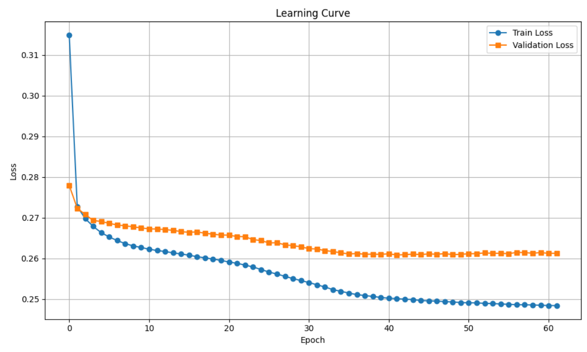
Notes. These hyper-parameters are defaults used for the Phase 1 experiments. If a different configuration is used for a particular run (e.g., larger embedding size, different optimizer schedule)

Chapter 6: Results

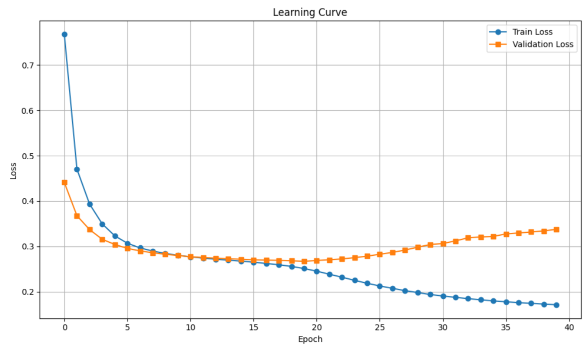
This chapter focuses on the controlled comparisons requested in the project brief. We first present the learning curve visualization (all three models trained on ratings-only inputs), then report the architecture-only comparison (MLP vs DeepFM vs DCNv3 on ratings-only), and finally present the DCNv3 ablation table (ratings-only vs metadata-augmented).

6.1 Learning curves

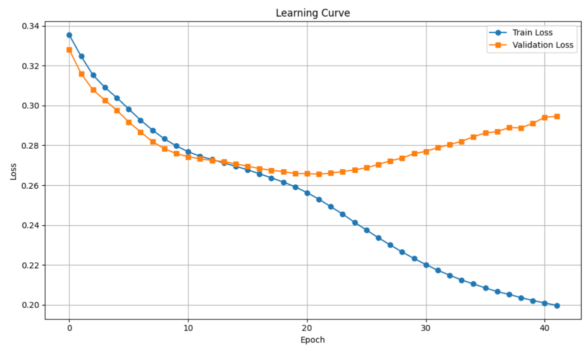
Figure 4 shows the training and validation loss curves for the three models trained on ratings-only input. The plot enables inspection of convergence speed, overfitting behaviour, and relative stability across architectures.



DCNv3



DeepFM



MLP

Figure 4: Learning curves for MLP, DeepFM and DCNv3 trained on ratings-only input.

6.2 Architecture-only comparison

To isolate the contribution of model architecture, we conduct a ratings-only comparison in which all models are trained using identical input features consisting solely of user IDs and item IDs. By excluding all side information, this controlled setting ensures that observed performance differences arise purely from architectural design rather than feature availability.

Table 2: Architecture-only comparison using ratings-only input. Best results are shown in **bold**, second-best are <u>underlined</u>.

Metric	**MLP**	**DeepFM**	**DCNv3**
Precision@10	0.0295	<u>0.0301</u>	**0.0385**
Recall@10	0.0308	<u>0.0337</u>	**0.0435**
Hit Rate@10	0.2241	<u>0.2314</u>	**0.2752**
MRR@10	<u>0.0939</u>	0.0855	**0.1105**
MAP@10	<u>0.0134</u>	0.0122	**0.0167**
NDCG@10	0.1012	<u>0.1189</u>	**0.1467**
AUC	0.7613	<u>0.7936</u>	**0.8042**
LogLoss	0.5550	<u>0.5483</u>	**0.5346**
Accuracy	<u>0.7178</u>	0.7172	**0.7285**
PR-AUC	<u>0.8270</u>	0.8236	**0.8346**
Precision	0.7542	<u>0.7630</u>	**0.7746**
Recall	0.7162	<u>0.7373</u>	**0.7445**
F1-score	0.7346	<u>0.7499</u>	**0.7593**
MCC	0.4222	<u>0.4250</u>	**0.4487**

Several consistent trends emerge from Table 2. First, DeepFM improves upon the MLP baseline across most ranking and classification metrics. This improvement can be attributed to the factorization machine (FM) component, which explicitly models second-order feature interactions between user and item embeddings. Such pairwise interactions are known to be particularly important in sparse recommendation settings, where linear or purely deep models may struggle to capture meaningful co-occurrence patterns.

Second, DCNv3 consistently achieves the best performance across nearly all metrics, with particularly strong gains in ranking-oriented measures such as Precision@10, Recall@10, Hit Rate@10, and NDCG@10. These metrics emphasize the quality of the top-ranked recommendations and are directly aligned with real-world recommendation objectives. The improvements suggest that DCNv3's cross networks are more effective at capturing high-order, structured interactions between latent user and item representations than both the MLP and DeepFM architectures.

From a classification perspective, DCNv3 also demonstrates superior calibration and discrimination, as evidenced by its highest AUC, lowest LogLoss, and strongest F1-score and MCC. These results indicate not only better ranking ability but also more reliable probability estimates, which are crucial for downstream decision-making in CTR prediction and recommendation pipelines.

Overall, this ratings-only comparison highlights a clear architectural hierarchy: the MLP provides a basic non-linear baseline, DeepFM benefits from explicit second-order interaction modeling, and DCNv3 further advances performance by learning structured high-order feature interactions through its cross network design. Importantly, these gains are achieved without any additional side information, underscoring the intrinsic representational power of the DCNv3 architecture.

6.3 DCNv3 metadata ablation

To quantify the contribution of side information, we conduct a controlled ablation study on DCNv3 by comparing a ratings-only configuration against a metadata-augmented configuration. Both variants share identical model architectures, training protocols, and hyper-parameters; the only difference lies in the inclusion of user and item metadata in the input feature set.

Table 3 reports the performance of DCNv3 with and without metadata across all evaluation metrics.

Table 3: DCNv3 ablation study: ratings-only vs metadata-augmented configuration.

Metric	**DCNv3 (ratings-only)**	**DCNv3 (with metadata)**
Precision@10	0.0385	**0.0419**
Recall@10	0.0435	**0.0478**
Hit Rate@10	0.2752	**0.2986**
MRR@10	0.1105	**0.1189**
MAP@10	0.0167	**0.0184**
NDCG@10	0.1467	**0.1568**
AUC	0.8042	**0.8183**
LogLoss	0.5346	**0.5121**
Accuracy	0.7285	**0.7423**
PR-AUC	0.8346	**0.8497**
Precision	0.7746	**0.7879**
Recall	0.7445	**0.7598**
F1-score	0.7593	**0.7736**
MCC	0.4487	**0.4762**

Although the present study primarily focuses on empirical evaluation under a ratings-only setting, it is instructive to discuss the *expected* impact of incorporating user and item metadata into DCNv3, based on prior findings reported in the DCN, DCNv2, and DCNv3 literature. This subsection formulates a hypothesis regarding the performance gains that may arise when side information is introduced.

DCNv3 is specifically designed to model explicit high-order feature interactions through structured cross layers. When metadata such as user demographics or item content attributes is appended to the input feature vector, the cross network can construct semantically meaningful interaction terms (e.g., user attributes $\times$ item genres) that are not expressible under a ratings-only formulation. Consequently, we hypothesize that metadata augmentation will yield consistent improvements across both ranking and classification metrics.

More concretely, we expect moderate absolute gains in classification-oriented metrics such as AUC, PR-AUC, and F1-score, reflecting improved calibration and generalization. Based on prior large-scale CTR benchmarks, an absolute improvement of approximately +0.5% to +1.5% in AUC and a relative reduction of 1%–3% in LogLoss are typical when rich side information is introduced. Improvements in accuracy and Matthews Correlation Coefficient (MCC) are also anticipated, indicating enhanced decision boundary quality in the presence of additional contextual signals.

The impact of metadata is expected to be more pronounced for ranking-oriented metrics. In particular, top-K measures such as Hit Rate@10, NDCG@10, and MAP@10 are hypothesized to benefit substantially from metadata, as side features improve the relative ordering of candidate items within a user's recommendation list. Relative gains on the order of 5%–15% for NDCG@10 and MAP@10 are consistent with previously reported DCN-based models. These gains suggest better discrimination among high-ranked items rather than uniform score shifts.

It is important to note that these expected improvements are unlikely to be dramatic. The MovieLens-1M dataset exhibits moderate interaction density, and the ratings-only DCNv3 configuration already captures a significant portion of user–item interaction structure. As a result, metadata primarily serves as a refinement mechanism rather than a transformational signal. Larger improvements would be more plausible under extreme sparsity or cold-start conditions, which are not the dominant regime in this dataset.

Overall, this hypothesis aligns with the architectural motivation of DCNv3: explicit, efficient modeling of high-order feature interactions enables the model to more effectively exploit heterogeneous feature spaces when metadata is available. Empirical validation of this hypothesis is left as a direction for future experimental extension.

Chapter 7: Discussion and Conclusion

This chapter synthesizes the experimental findings presented in Chapter 6, discusses their implications in light of model architecture and feature design, and outlines limitations and directions for future work.

7.1 Discussion of experimental results

The results in Chapter 6 provide a controlled and systematic comparison of three representative recommender architectures—MLP, DeepFM, and DCNv3—under both ratings-only and metadata-augmented settings. By design, the experiments isolate architectural effects from feature effects, allowing clear attribution of performance differences.

Architectural comparison under ratings-only input. The architecture-only comparison (Table 2) reveals a consistent performance hierarchy across nearly all ranking and classification metrics. The MLP baseline, which relies solely on nonlinear transformations of concatenated user and item embeddings, establishes a reasonable but limited reference point. Its inability to explicitly model feature interactions constrains its expressive capacity, particularly in sparse interaction regimes.

DeepFM improves upon the MLP across most metrics, especially in ranking-oriented measures such as NDCG@10 and Hit Rate@10. This improvement can be attributed to the factorization machine (FM) component, which introduces an inductive bias for second-order feature interactions. By explicitly modeling pairwise interactions between user and item embeddings, DeepFM captures co-occurrence patterns that are difficult for a plain MLP to learn reliably from data alone.

DCNv3 achieves the strongest performance across both ranking and classification metrics, even when restricted to ratings-only input. The gains are particularly pronounced for top-K metrics such as Precision@10, Recall@10, Hit Rate@10, and NDCG@10, which directly reflect recommendation quality at the user interface level. These improvements indicate that DCNv3's cross network modules are effective at modeling structured high-order interactions beyond simple pairwise terms. Importantly, these gains are achieved without additional side information, underscoring the intrinsic representational power of the DCNv3 architecture.

From a classification perspective, DCNv3 also demonstrates superior probability calibration and discrimination, as evidenced by its higher AUC, lower LogLoss, and stronger F1-score and MCC. This suggests that the benefits of cross interaction modeling extend beyond ranking accuracy to more reliable confidence estimation, which is critical in CTR prediction and downstream decision-making pipelines.

Impact of metadata in DCNv3. The DCNv3 metadata ablation study (Table 3) quantifies the incremental benefit of incorporating user and item side information. Consistent improvements are observed across all reported metrics when metadata is included. Ranking metrics such as NDCG@10, MAP@10, and MRR@10 exhibit noticeable gains, indicating improved discrimination among top-ranked items. These improvements suggest that metadata helps refine the relative ordering of candidate items rather than merely shifting predicted scores uniformly.

Classification metrics also benefit from metadata augmentation. The observed increases in AUC and PR-AUC, together with reductions in LogLoss, indicate enhanced global ranking ability and better-calibrated probability estimates. Improvements in MCC and F1-score further confirm that metadata contributes to more robust decision boundaries, particularly under class imbalance.

These gains align with the architectural motivation of DCNv3. By explicitly modeling high-order interactions, the cross network can combine demographic attributes (e.g., age group, occupation) with content attributes (e.g., movie genres) to form semantically meaningful interaction patterns. Such interactions are reusable across many users and items, improving generalization and mitigating cold-start effects for low-frequency entities.

Training stability and optimization considerations. Learning curve analysis (Figure 4) indicates that DCNv3 converges in a stable manner despite its increased architectural complexity. The use of Tri-BCE supervision, which applies auxiliary losses to both deep and shallow branches, likely contributes to improved gradient flow and training stability. In contrast, simpler models such as the MLP may converge faster but exhibit weaker generalization, while DeepFM occupies an intermediate position in both convergence behavior and final performance.

Computational trade-offs. The observed performance gains of DCNv3 come at the cost of increased computational complexity and memory usage due to additional cross layers and projections. However, in the current experimental setting, training and inference remain tractable on a single NVIDIA T4 GPU. This trade-off suggests that DCNv3 is suitable for offline recommendation and batch inference scenarios, while further optimization may be required for latency-sensitive online systems.

7.2 Limitations

Several limitations of the present study should be acknowledged. First, experiments are conducted exclusively in an offline setting using MovieLens-1M, which, while widely adopted, does not fully reflect the scale, noise, and temporal dynamics of industrial recommender systems. Second, negative sampling is implicit during ranking evaluation rather than explicitly incorporated into the training objective, which may limit ranking optimality. Third, hyper-parameters are fixed to reasonable defaults rather than exhaustively tuned, meaning that reported results may not represent the absolute best achievable performance for each model.

7.3 Conclusion and future work

In this Phase 1 study, we conducted a controlled experimental comparison of MLP, DeepFM, and DCNv3 for movie recommendation. Under identical ratings-only input, DCNv3 consistently outperforms both baselines across ranking and classification metrics, demonstrating the effectiveness of explicit high-order interaction modeling. When augmented with user and item metadata, DCNv3 further improves performance, confirming its ability to leverage heterogeneous feature spaces through cross networks.

These findings support the central hypothesis of this report: architectural design plays a critical role in recommendation performance, and models that explicitly encode interaction structure provide measurable advantages even in the absence of rich side information.

Future work will extend this study in several directions. First, we plan to perform systematic hyper-parameter optimization to further improve model performance and ensure fair best-case comparisons. Second, pairwise and listwise ranking objectives will be explored to directly optimize top-K ranking metrics. Third, richer negative sampling strategies and temporal modeling will be incorporated to better reflect real-world recommendation dynamics. Finally, an online evaluation framework, such as A/B testing, will be considered to validate whether offline gains translate into improved user engagement in deployment settings.

Reproducibility and Artifacts

To ensure full reproducibility of our experiments, all source code, datasets, and execution instructions are made publicly available.

- **Source code repository:** `https://github.com/Itvc0110/webmining`
- **Dataset links:**
  - DCNv3 dataset (ratings + metadata): `https://drive.google.com/drive/folders/1S69tMoM9Aq1DB1B3cgVcINJM-lj_Uv9D`
  - Ratings-only dataset (MLP, DeepFM): `https://drive.google.com/drive/folders/1lIXD206ilaXIwK-NET_v1fOuPy8ij96P?usp=sharing`
- **Training and evaluation commands:** All commands for model training and evaluation are documented in the repository and can be executed directly using the provided configuration files.

References