Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 18** : 1–11.
ISSN-0037-8712
doi:10.5269/bspm.82713

Invertible Neural Networks for Matrix Factorization in Recommendation Systems

Soukaina Tamim, Atika Radid and Karim Rhofir

ABSTRACT: Recommendation systems play a central role in personalizing digital content by leveraging user–item interactions. Matrix factorization is widely used but has limitations, particularly when integrating both explicit feedback (ratings) and implicit feedback (clicks or lack of interactions). In this work, we propose a framework based on invertible neural networks for matrix factorization. The interaction matrix $A \in \mathbb{R}^{M \times N}$ contains explicit ratings and implicit feedback for M users and N items. The goal is to estimate missing entries to predict preferences and generate personalized recommendations. Unlike classical methods based on dot products of latent vectors, our approach projects users and items into a low-dimensional latent space via a nonlinear and invertible transformation. This design preserves information, reconstructs the original interactions, and simultaneously exploits explicit and implicit feedback. Experiments on benchmark datasets demonstrate the potential of the proposed model. The framework also applies to other loss functions and can integrate heterogeneous data sources (text, social networks, browsing history) in order to improve robustness and personalization.

Keywords: Invertible neural networks, matrix factorization, recommendation systems, implicit feedback.

Contents

1. Introduction

Numerous amounts of data exist now than ever, thus making it increasingly difficult for users to find relevant information quickly and easily [9,14]. One method of assisting users in doing so has been through the use of Recommender Systems (RS), which identify, collect, and recommend items to users based upon their unique preferences [8,10]. RS have become invaluable resources for locating and developing an extensive collection of recommended resources, from recommending books on Amazon to recommending movies on Netflix, recommending music on Last.fm, and recommending scientific articles on CiteULike. By guiding users toward items of interest, RS alleviate cognitive overload, enhance user satisfaction, and improve overall engagement.

2020 *Mathematics Subject Classification:* 68T05, 68T09.
Submitted March 16, 2026. Published June 19, 2026.

Collaborative Filtering (CF) is a very common approach for making recommendations to users, and its popularity can be attributed to how straightforward a technique CF is to implement [9,13]. The CF approach is based on the idea that two different users with similar past preferences will also share similar future preferences. The Classical Matrix Factorization (CMF) techniques used to represent users and items as latent variables are depicted using a common latent space in which a specific user's interaction with various items is encoded as a vector [9,13]. Matrix Factorization techniques, therefore, both offer flexibility and scalability, but rely on having explicit user ratings that are often sparse and incomplete, and therefore do not allow to accurately predict or generalise user interactions to new data.

Many extensions have been proposed to address sparse and incomplete datasets. Social Matrix Factorization (SMF) utilizes social relationships between users to improve latent representations. Meanwhile, content-based methods rely on item attributes (e.g., textual descriptions) or user-generated content (e.g., reviews) to create a deeper understanding of users' preferences [8]. Implicit feedback sources (e.g., clickstreams, browsing history, purchase records) allow for a more complete picture of the user's preference compared to explicit ratings [14]. Newer predictive algorithms, such as SVD++ and Bayesian Personalized Ranking (BPR), incorporate both implicit and explicit feedback while enhancing their ability to create top-N recommendations and provide a greater degree of protection against sparsity [11,12].

The introduction of deep learning has had a major impact on reinventing Recommendation Systems. With the advent of new neural network architectures, including Restricted Boltzmann Machine (RBM) [9], Autoencoder and Denoising Autoencoder [15,10], and Neural Collaborative Filtering (NCF) [3], we can create more advanced and intelligent representations of the relationship between users and items than ever before by fully utilizing deep architecture's ability to learn deep representations of both users and items to build sophisticated models that reflect the complexity and non-linearity of those relationships.

Also, one of the main areas of research in machine learning is the generative model. Generally speaking, the goal of this type of model is to estimate the probability distribution of a set of data in order to create new examples from that probability distribution; for instance, generative models are used to generate new instances of existing objects. Generally, classical generative models are based on the Boltzmann machine and provide a systematic approach to representing probabilities, but in practice these models suffer from computationally expensive calculations and limited ability to represent the data set [2]. Variational Autoencoders (VAEs) allow for efficient learning but are limited to providing only lower bounds on likelihood [6]. Finally, GANs (Generative Adversarial Network) can be used to generate examples that are very close in appearance to examples of existing data and high-dimensional data; however, GANs cannot provide accurate likelihood estimates and therefore are of limited use in probabilistic reasoning.

By employing the computational efficiency of likelihood calculation and sampling techniques, Normalizing Flows (NF) provide an easy and robust method of generating and manipulating probability distributions [7,5,4] through invertible functions used for transforming simple (e.g., multivariate Gaussian) latent distributions into complex distributions over observed data. Typical architecture for NF have built-in computationally demanding coupling layers that can limit the applicability of NF to high-dimensional spaces or to massive datasets [4].

In this paper, we introduce LU-Net [16], an Invertible Neural Network (INN) based on Normalizing Flows [7,6,4] that achieves high expressiveness at a significantly lower computational cost than other architectures by leveraging the LU Decomposition technique of Fully Connected Layers (in addition to being able to be implemented using simple matrix multiplication) [2,1]. By using an LU Decomposition, all weight matrices of the INN model become constrained to be lower and upper triangular matrices; therefore, both the forward and backward pass can be computed directly with minimal complexity, resulting in a reduction in the computational cost of inversion from $O(D^3)$ to $O(D^2)$, and the cost of determinant calculation from $O(D^3)$ to $O(D)$ [5].

We have implemented LU-Net for recommendation systems by applying a single unified latent representation of explicit ratings and implicit feedback [11] to the user's implicit and explicit behaviours toward the items in our database. The use of LU-Net model achieves an excellent representation of complex interaction mechanisms. Furthermore, by extensively testing our LU-Net model using established benchmarks such as MovieLens 100K and MovieLens 1M, we demonstrated that our LU-Net model achieves competitive results with low computational resources.

The organization of this paper is as follows: In Section 2, we discuss the full architecture for LU-Net and describe how each of the invertible layers is created and how the distribution density and likelihood values are computed for each layer of LU-Net. In Section 3, we outline the experimental configuration, data sets, evaluation methods, and numerical results. Section 4 will serve as a summary and conclusion for the entire paper.

2. Design of the Invertible Model

2.1. Recommendation Systems and Problem Formulation

Recommender systems provide suggestions to users by analyzing their behavior and indicating similar interests. In today's market, there is a large amount of information available online; therefore, recommender systems are important for filtering through all this data for users and creating personalized suggestions.User interactions with different products are often displayed in an interaction matrix shown as $Y \in \mathbb{R}^{M \times N}$. Where M is the number of users, and N is the number of items. Each entry Y_{ij} represents how user u_i interacted with the item i_j, providing valuable insight into what users prefer and items are. Usually, you are able to identify 2 types of feedback, explicit and implicit. Explicit feedback refers to a numerical rating or score, which is provided directly by the user, indicating how strongly he/she feels about something. Although explicit feedback contains useful information, it is typically limited in quantity. Implicit feedback is collected by observing how a user behaves, for example through his/her clicks, views and purchases. This kind of feedback is sometimes uncertain and distorted, therefore learning about what a user does not interact with is complicated because it does not mean the user did not like the content.

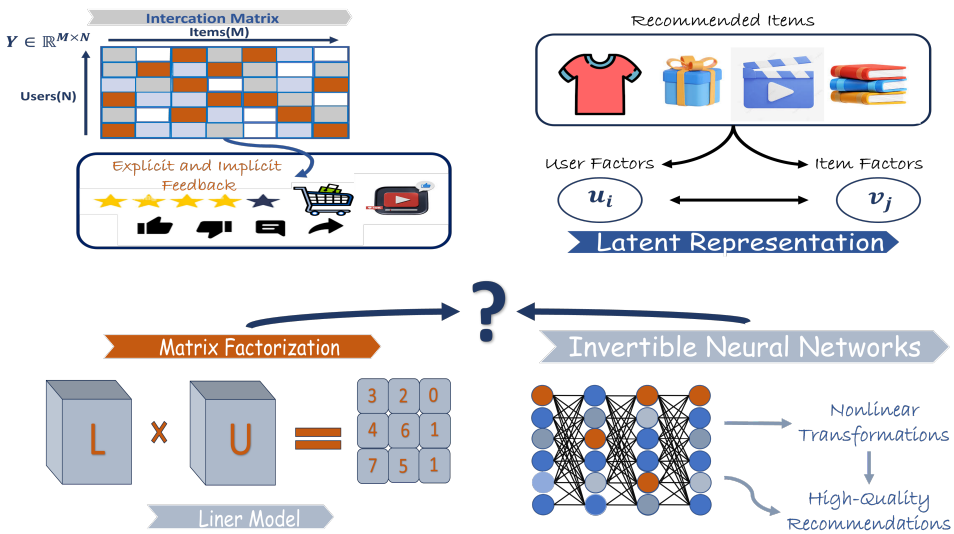

Figure 1: Recommendation System Framework.

The objective of a recommendation task is to identify the unknown entries of an interaction matrix so that we can obtain a measurement of each user's preferences and make a prediction regarding which items will be of interest. Matrix factorization is the traditional method of obtaining this information. It does so by representing items and users in a latent space that represents shared characteristics of both users and items (linear interaction). Although effective in providing an overall view of user behaviour through statistical methods, matrix factorization models generally have limited ability to express complex interactions and patterns within large sets of data. This is especially true when most of the data provided by an end-user is implicit.

As a result, researchers have been motivated to create nonlinear models that can better take into account the nature of both explicit and implicit feedback while maintaining useful information about

the data. Using invertible neural networks is now an obvious choice, as they support the creation of highly expressive nonlinear transformations while guaranteeing that all transformations can be accurately reconstructed. This is particularly valuable for recommendation applications needing both high-quality predictions as well as consistency across different latent representations used for each item recommended.

The proposed model relies on fully connected layers modified to ensure invertibility [2,1]. Each layer applies a linear transformation followed by a bijective activation:

$$x \longmapsto f(x) = \phi(LUx + b), \tag{2.1}$$

where L and U are lower and upper triangular matrices with non-zero diagonals, b is a bias vector, and ϕ is a bijective activation function [5,4]. The inverse transformation is:

$$f^{-1}(z) = U^{-1}L^{-1}(\phi^{-1}(z - b)), \tag{2.2}$$

allowing exact reconstruction of latent embeddings and preserving information.

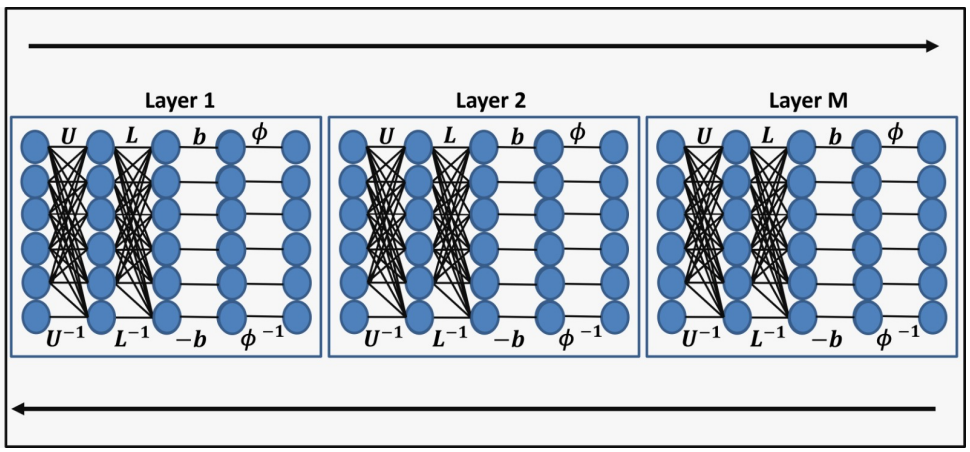

Figure 2: Schematic architecture of the invertible neural network.

2.2. LU Factorization

LU factorization decomposes a square matrix $A \in \mathbb{R}^{d\times d}$ into a product of a lower triangular matrix L with ones on the diagonal and an upper triangular matrix U with non-zero diagonal elements [1,2]:

$$A = LU = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{2,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{d,1} & l_{d,2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,d} \\ 0 & u_{2,2} & \cdots & u_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{d,d} \end{pmatrix} \tag{2.3}$$

where $u_{ii} \neq 0$ for $i = 1, \ldots, d$. The matrix A is assumed to be square and non-singular, which ensures the existence and uniqueness of the LU decomposition without the need for pivoting. This factorization allows efficient computation of the inverse A^{-1} and the determinant [5]:

$$\det(A) = \prod_{i=1}^{d} u_{ii}. \tag{2.4}$$

2.3. Invertible Neural Network Architecture

The proposed LU-Net architecture maps an input vector $x \in \mathbb{R}^d$ to an output of identical dimension through a sequence of M invertible layers [4,6]. Each layer performs a triangular linear transformation followed by a smooth bijective activation. More precisely, the transformation operated at layer m is given by:

$$f^{(m)}(x) = \phi\big(L^{(m)}U^{(m)}x + b^{(m)}\big), \tag{2.5}$$

where $L^{(m)}$ and $U^{(m)}$ denote respectively a lower and an upper triangular matrix, $b^{(m)} \in \mathbb{R}^d$ is a learnable bias vector, and ϕ is the LeakySoftplus activation ensuring the invertibility of the layer.

The choice of the activation function is crucial, as it must preserve both the bijectivity and the expressiveness of the model [2]. To this end, LU-Net employs the *LeakySoftplus* function, defined by:

$$\text{LeakySoftplus}(x) = x + (1-\alpha)\ln\big(1+e^{-x}\big), \qquad 0 < \alpha < 1. \tag{2.6}$$

This activation behaves as a smooth blend of leaky ReLU and softplus. While both of these classical activations are invertible, they also present limitations: leaky ReLU has zero second derivative almost everywhere, which is detrimental for likelihood-based training, and softplus has an inverse defined only on $\mathbb{R}^+$, restricting layer outputs. LeakySoftplus avoids these drawbacks, making it particularly suitable for invertible architectures [6].

For all intermediate layers $m = 1, \ldots, M-1$, we therefore use:

$$\phi^{(m)}(x) = \text{LeakySoftplus}(x).$$

Since our task corresponds to a regression problem, the final layer does not apply any non-linearity:

$$\phi^{(M)}(x) = x,$$

which corresponds to the identity mapping.

Stacking the M layers yields the complete invertible transformation:

$$f(x) = f^{(M)} \circ f^{(M-1)} \circ \cdots \circ f^{(1)}(x), \tag{2.7}$$

whose inverse is analytically available by reversing the sequence of layers:

$$f^{-1}(z) = (f^{(1)})^{-1} \circ (f^{(2)})^{-1} \circ \cdots \circ (f^{(M)})^{-1}(z). \tag{2.8}$$

2.4. Activation Function

We use a bijective nonlinear LeakySoftplus function, defined as [2,6,4]:

$$\phi(x) = \frac{1}{\alpha}\log(1+e^{\alpha x}) + \beta x, \quad \alpha \in (0,1), \tag{2.9}$$

which combines the advantages of leaky ReLU and softplus. This function ensures non-zero gradients during backpropagation and is invertible over $\mathbb{R}$ [6,4].

The addition of the parameters α and β in equations (2.9) allow for flexibility in the shape of the activation function and stability for the optimisation of the network during training. These parameters allow for greater flexibility over the gradient and therefore, allow for faster and more reliable convergence than in the base configuration (i.e. $\alpha = 1$ and $\beta = 0$). In contrast, equation (2.6) represents a more straightforward approach to defining the activation function; however, the parameters available do provide limited control for dynamically altering the training characteristics of the function, as demonstrated in equation (2.9) and the subsections that follow. Although this representation is simpler to interpret than those seen in equation (2.9), they will create limitations on the control that can be exercised over the shape of the activation function that you create. Hence, we will create a generalization of the LeakySoftplus Activation Function to use in our models based on the mathematical equation (2.9) above; as such, the purpose for creating this specific activation function was to provide users with finer control of the gradient

through this activation function for greater accuracy in learning and optimizing the models by ensuring that all hidden layer transformations performed by the model remain invertible.

All hidden layers apply this activation. For the final layer, predicting user–item ratings, we do not apply a nonlinear activation:

$$f^{(M)}(x) = L^{(M)}U^{(M)}x + b^{(M)}. \tag{2.10}$$

2.5. Loss Function and Learning

To jointly exploit explicit ratings and implicit feedback, we adopt a hybrid learning strategy based on a combined loss function. This formulation allows the model to simultaneously predict rating values and estimate the probability of user–item interactions, which is essential in realistic recommendation scenarios where explicit feedback is sparse while implicit signals are abundant [3,11,15].

Formally, the overall objective function is defined as a weighted combination of a mean squared error loss for explicit ratings and a cross-entropy loss for implicit interactions:

$$\mathcal{L}_{\text{combined}} = \gamma\, \mathcal{L}_{\text{MSE}} + (1-\gamma)\, \mathcal{L}_{\text{CE}}, \tag{2.11}$$

where the parameter $\gamma \in [0,1]$ controls the trade-off between the two learning objectives.

The mean squared error loss is computed over the set of observed explicit interactions Y^+ and is defined as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{|Y^+|} \sum_{(i,j)\in Y^+} \left(Y_{ij} - \hat{Y}_{ij}\right)^2, \tag{2.12}$$

where Y_{ij} denotes the true rating provided by user u_i for item i_j, and $\hat{Y}_{ij}$ is the corresponding rating predicted by the model.

For implicit feedback, user behavior is modeled as a binary interaction signal. The cross-entropy loss is therefore defined over both observed interactions Y^+ and sampled non-interactions Y^- as:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{|Y^+ \cup Y^-|} \sum_{(i,j)\in Y^+\cup Y^-} \Big[Y_{ij} \log \hat{Y}_{ij} + (1 - Y_{ij}) \log(1 - \hat{Y}_{ij})\Big]. \tag{2.13}$$

In this setting, the network output $\hat{Y}_{ij} \in (0,1)$ is interpreted as the probability that user u_i interacts with item i_j. This probability is obtained by applying a sigmoid function to the output of the final linear layer:

$$\hat{Y}_{ij} = \sigma(f(x_{ij})) = \frac{1}{1 + \exp(-f(x_{ij}))}, \tag{2.14}$$

where $f(\cdot)$ denotes the mapping implemented by the invertible network.

For explicit feedback, the predicted value $\hat{Y}_{ij}$ directly represents the estimated rating, whereas for implicit feedback it corresponds to the probability of interaction (e.g., a click or a purchase). The accuracy of explicit rating prediction is evaluated using the Root Mean Squared Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{|Y^+|} \sum_{(i,j)\in Y^+} \left(Y_{ij} - \hat{Y}_{ij}\right)^2}. \tag{2.15}$$

For implicit feedback, performance is assessed through ranking-oriented metrics. In particular, the precision at top-K recommendations is defined as:

$$\text{Precision@K} = \frac{1}{K} \sum_{j\in\mathcal{R}_i^K} \mathbb{I}(Y_{ij} = 1), \tag{2.16}$$

where $\mathcal{R}_i^K$ denotes the set of the K highest-ranked items recommended to user u_i, and $\mathbb{I}(\cdot)$ is the indicator function defined as

$$\mathbb{I}(Y_{ij} = 1) = \begin{cases} 1, & \text{if user } u_i \text{ has effectively interacted with item } v_j, \\ 0, & \text{otherwise.} \end{cases}$$

This hybrid learning framework enables the model to achieve a balanced optimization between accurate rating prediction and reliable interaction modeling, leading to improved recommendation quality in scenarios involving both explicit and implicit feedback [15,3].

3. Experiments and Results

3.1. Experimental Settings

3.1.1. Datasets. We evaluate our models on two widely used datasets in recommender systems: MovieLens 100K (ML100k) and MovieLens 1M (ML1m) [3,11]. Both datasets are publicly available at https://grouplens.org/datasets/movielens/. No additional preprocessing is required as the datasets are already filtered; only users with at least one rating and items with at least one rating are included.

The statistics of the two datasets are summarized in Table 1.

Table 1: Statistics of the MovieLens 100K and 1M datasets.

Dataset	ML100k	ML1m
Users	943	6,040
Items	1,682	3,952
Ratings	100,000	1,000,209
Rating Density	0.0629	0.0447

3.1.2. Evaluation Protocol and Experimental Setup. To evaluate our model we use a conventional train-test split alongside a leave-one-out approach. This means that 80% of the user's past activity is used for training and the remaining 20% is evaluated during testing. As part of the leave-one-out approach, we retain the most recent activity from every user before training and then rank them against a sample of 100 random items that are not in the user's history. The purpose of this evaluation is to determine the model's performance in making relevant recommendations over a pool of many other items. Hit Ratio (HR@10) and Normalized Discounted Cumulative Gain (NDCG@10) were employed to evaluate the ranking performance. It was determined whether or not the held-out item appeared within the top ten recommendations (HR@10), while NDCG@10 measured how far from the top a correct recommendation was located within the ranked list. Both metrics heavily favor items closer to the top. Moreover, the following measures were reported for evaluating the quality of the representations created from learning: Mean Squared Error (MSE) and the embedding reconstruction error.

The experiments that everyone did were done using Python 3.10 and used PyTorch Version 2.0, NumPy and SciPy and Pandas. In training these models, Adam was the optimizer with a learning rate of 10^{-3}, a batch size of 128 and trained for 50 epochs on data using an Intel Core® i3 CPU with 8 GB of RAM and an NVIDIA® GPU (if available). This hardware set up would give the best chance that experiments would be reproducible, and it would also allow someone to compare two different models fairly and use different datasets.

3.2. Loss Evolution

3.3. Performance Comparison

Matrix Factorization-based Recommendation Models currently used are based upon Matrix Factorization (MF). Most MF-based models rely on linear interaction between both latent user and latent item representation. This linearity and constraint creates limitations for capturing complex preference patterns. Neural based recommendation approaches (e.g. MLP-Based models, Neural Collaborative Filtering) enable the creation of non-linear models, however they lack an inverse distribution resulting in loss of all information contained in learned representation features.Autoencoder approaches add an additional level of expressiveness over the previous non-linear model types, but because they utilize compressed latent variables as part of their procedures, accurate reconstruction cannot occur using the skilled autoencoder.

Normalizing Flow models attempt to address the limitations associated with inverse models by applying a constraint on the structure of the transformations by forcing all transformations to be bijective; however, the high computational cost associated with using coupling-layer based architectures limits the practical applications for which Normalizing Flow models are applicable. The proposed LU-Net model uses LU-based invertible transformations as the basis for a new architecture that utilizes not only an effective means of reconstructing latent embeddings exactly but also retains a capacity for nonlinear modeling with predictive performance comparable to current methods.

Table 2: Performance comparison between a classical MLP and the proposed invertible model.

Model	Test MSE	Embedding Reconstruction Error
Classic MLP	0.9200	0.0050
Invertible Model (ML-100K)	0.8880	0.0000
Invertible Model (ML-1M)	0.7955	0.0000

4. Discussion

According to the presented results, the invertible architecture proposed is able to recreate the latent embeddings with perfect accuracy (Embedding Reconstruction Error 0). Furthermore, the invertible model achieves lower test MSE while perfectly reconstructing the latent embeddings, and still produces good results on the rating prediction problems [3,15].In addition, the proposed invertible architecture is able to remain stable across multiple configurations of the experimental environment; this means that by varying the degree of dimensionality in the embedding, and varying the amount of LU-blocks,it is still able to maintain the same performance levels. Therefore, it can be concluded that the invertible architecture proposed is quite robust to variations in hyperparameters.

An additional point to be taken into consideration when evaluating the proposed model for the rating prediction problem is that the rating prediction error remains relatively high (approximately 0.8). The magnitude of this error is reasonable for two primary reasons:(1) the MovieLens datasets are exceptionally sparse, and (2) they contain a significant amount of noise in user-item interactions. Consequently, limitations in predictive precision arise from larger fluctuations in user behavior, which lead to increased variance in ratings within the same scale. The main objective of LU-Net is not only to reduce prediction error, but also to preserve the invertibility of the latent space through LU-based transformations. As a result these structural constraints may slightly reduce prediction accuracy while ensuring full preservation of the encoded information. Therefore, an error close to 0.8 can be interpreted as a reasonable trade-off between prediction performance, predictive stability, and invertibility.

Finally, experiments conducted on the ML-1M dataset confirm that the model scales effectively to larger datasets, maintaining reliable rating predictions and near-perfect reconstruction of latent represen-

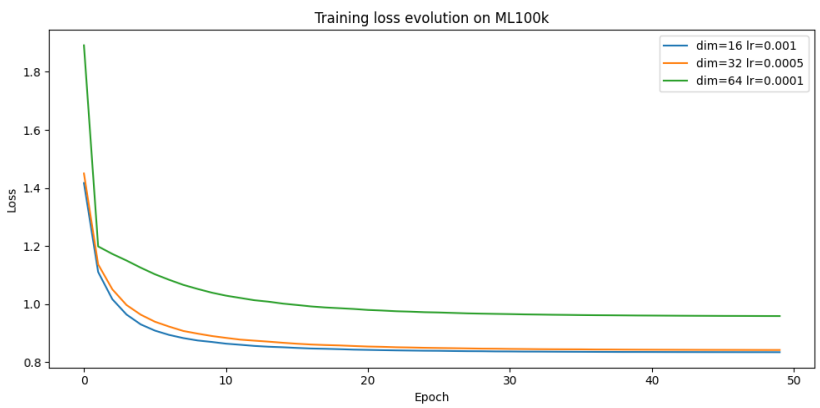

Figure 3: Training loss evolution on ML100k.

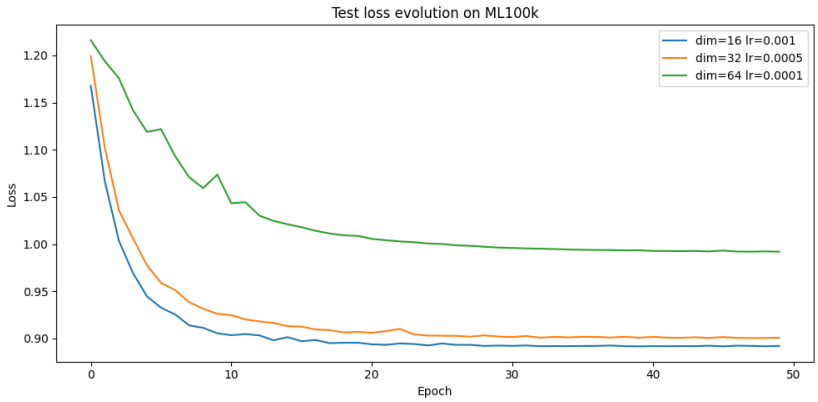

Figure 4: Test loss evolution on ML100k.

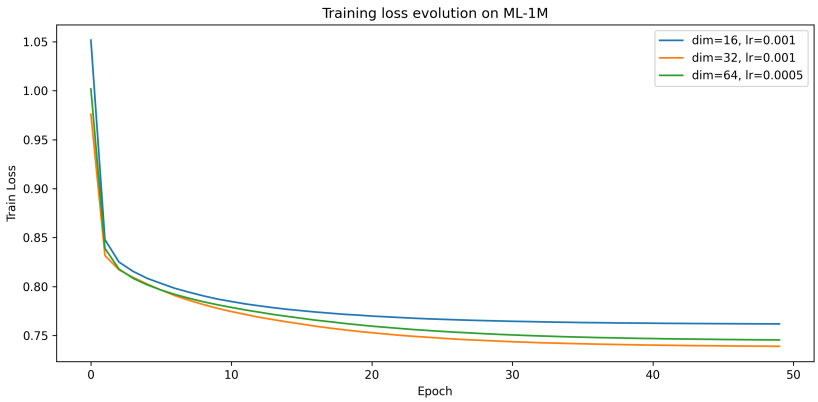

Figure 5: Training loss evolution on ML1m.

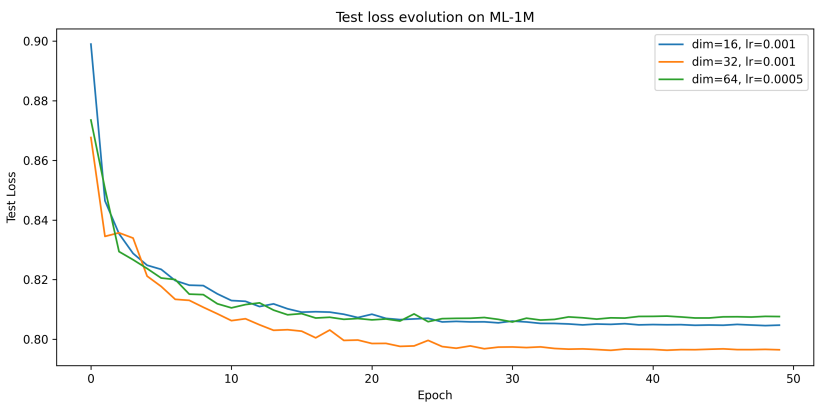

Figure 6: Test loss evolution on ML1m.

tations [1,5]. This behavior is directly related to the invertible nature of the architecture, which relies on information-preserving transformations. Consequently, LU-Net is particularly well suited for applications that require both generative modeling capabilities and robust representation learning [6,7,5].

5. Conclusion

In this paper, we introduced LU-Net, a novel architecture of invertible neural networks for matrix factorization in recommender systems. We make use of bijective transformations in the model to ensure preserving mapping and parameters for each pair through a shared model reconstructed user-item embeddings with nontrivial, nonlinear interactions among the users and items.

We experiment and analyze the benchmark datasets MovieLens 100K and 1M, and it is shown that LU-Net reaches the competitive performance between prediction accuracy and latent representation fidelity while remaining computationally efficient.

This inversibility structure guarantees that no information is lost during the mapping from input to latent spaces, suitable for recommendation tasks and models that need generative modeling or for accurate density estimation. Extensions to our work could include future research into the integration of side modelling temporal dynamics or LU-Net can be applied to other domain where scalable and expressive invertible networks may have benets.

References

1. Palm, G., *On representation and approximation of nonlinear systems: part II: discrete time*, Biol. Cybernet., **34**(1), 49–52, (1979).
2. Cybenko, G., *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Syst., **2**(4), 303–314, (1989).
3. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S., *Neural collaborative filtering*, arXiv preprint arXiv:1708.05031, (2017).
4. Ardizzone, L., Kruse, J., Wirkert, S., Rahner, D., Pellegrini, E. W., Klessen, R. S., Maier-Hein, L., Rother, C., Köthe, U., *Analyzing inverse problems with invertible neural networks*, arXiv preprint arXiv:1808.04730, (2018).
5. Kingma, D. P., Dhariwal, P., *Glow: generative flow with invertible 1×1 convolutions*, arXiv preprint arXiv:1807.03039, (2018).
6. Rezende, D. J., Mohamed, S., *Variational inference with normalizing flows*, arXiv preprint arXiv:1505.05770, (2015).
7. Dinh, L., Sohl-Dickstein, J., Bengio, S., *Density estimation using real NVP*, arXiv preprint arXiv:1605.08803, (2016).
8. Bao, Y., Fang, H., Zhang, J., *TopicMF: simultaneously exploiting ratings and reviews for recommendation*, in *Proc. AAAI*, (2014).
9. Billsus, D., Pazzani, M. J., *Learning collaborative information filters*, in *Proc. ICML*, (1998).
10. Elkahky, A. M., Song, Y., He, X., *A multi-view deep learning approach for cross domain user modeling in recommendation systems*, in *Proc. WWW*, 278–288, ACM, (2015).
11. He, R., McAuley, J., *VBPR: visual Bayesian personalized ranking from implicit feedback*, arXiv preprint arXiv:1510.01784, (2015).
12. He, X., Chen, T., Kan, M.-Y., Chen, X., *TriRank: review-aware explainable recommendation by modeling aspects*, in *Proc. CIKM*, 1661–1670, ACM, (2015).
13. He, X., Zhang, H., Kan, M.-Y., Chua, T.-S., *Fast matrix factorization for online recommendation with implicit feedback*, in *Proc. SIGIR*, 549–558, ACM, (2016).
14. Hu, Y., Koren, Y., Volinsky, C., *Collaborative filtering for implicit feedback datasets*, in *Proc. ICDM*, 263–272, IEEE, (2008).
15. Wu, Y., DuBois, C., et al., *Collaborative denoising auto-encoders for top-N recommender systems*, in *Proc. WSDM*, 153–162, ACM, (2016).
16. Chan, R., Penquitt, S., Gottschalk, H., *LU-Net: invertible neural networks based on matrix factorization*, arXiv preprint arXiv:2302.10524, (2023).
17. Xue, H.-J., Dai, X.-Y., Zhang, J., Huang, S., Chen, J., *Deep matrix factorization models for recommender systems*, in *Proc. IJCAI*, 3203–3209, (2017).

Soukaina Tamim, Atika Radid, Karim Rhofir,
Mathematical Analysis, Algebra and Applications Laboratory (LAM2A)
Faculty of Sciences Ain Chock, Hassan II University,
Casablanca, Morocco.
E-mail address: tamimsoukaina0@gmail.com; atikaradid@gmail.com; k.rhofir@usms.ma