Eötvös Loránd University

Faculty of Informatics

Dept. of Data Science

Collaborative Filtering Techniques on Alleviating the Cold Start Problem in Music Recommendation

Supervisor:
Kamuzora Adolf Rutebeka
Assistant Lecturer

Author:
Tala Petro
Computer Science MSc

Budapest, 2023

Contents

List of Figures

List of Tables

Acknowledgements

I wish to express my deepest gratitude to my supervisor, Adolf Rutebeka Kamuzora, whose invaluable guidance, assistance, and unwavering support have been instrumental throughout this journey. His efforts in helping crystallize my thoughts into academic discourse have truly shaped this thesis.

I extend heartfelt appreciation to Dr. Bruno Dos Santos. His captivating approach to introducing Neural Networks and the invaluable advice he offered at the outset of my research have significantly contributed to the accomplishment of this work.

I am deeply grateful to the Eötvös Loránd University for providing me with the opportunity to pursue my studies at an esteemed institution. My sincere thanks go out to all members of the Faculty of Informatics and the Department of Data Science and Engineering, whose collective efforts create a conducive environment for academic advancement.

I acknowledge with gratitude the Stipendium Hungaricum scholarship, whose financial assistance has been integral to my ability to study in Hungary.

Lastly, but most importantly, I would like to express my profound appreciation to my family. To my parents, Nabila and Imad, whose unwavering faith in me and unyielding support have been my pillars of strength. To my brothers, Khalil and Waseem, for their encouragement and camaraderie. And to my friends, whose unwavering support and continuous encouragement have been a source of comfort and motivation throughout my years of study. Your collective support has been indispensable to my personal and academic growth.

Chapter 1

Introduction

1.1 Motivation

Music Recommender Systems have seen a significant upsurge in recent years with the proliferation and success of online streaming services. Today, these platforms have virtually global music libraries at the disposal of the user, revolutionizing access to a broad spectrum of music[1].

Current music recommendation systems primarily leverage Collaborative Filtering (CF) techniques, learning from available data to identify similarities between users and songs based on users' listening history[2] where the prediction of whether a user is interested in a song or not is based on using similarities between various user profiles. However, such systems grapple with the cold start problem, i.e., when a new user signs up to the streaming service or a new song is added to the library, the system does not have sufficient data associated with these items/users, as they are unable to effectively recommend new songs without a listening history. To tackle this issue, content-aware recommendation methods incorporate additional content information about the songs into the collaborative filtering process[3]. In recent studies [4][5], the mapping of acoustic features to the acquired item attribute matrix facilitates their use in cold start recommendations. However, the performance can be hindered due to the separate learning phases for user/item embeddings and the deep content feature extractor. To mitigate this, recent studies propose a unified approach where the learning of user/item embeddings and content feature extraction occurs simultaneously, thereby enhancing overall performance. However, most of these approaches still rely on a simple matrix product to model the interaction between users and items, and do not leverage Deep Neural Networks (DNNs) for learning more complex interaction models[6].

A model called Neural Content-Aware Collaborative Filtering (NCACF) was introduced later to fill these gaps [6]. This innovative model signifies a leap forward from traditional content-aware recommendations, offering a unified framework that overcomes previous limitations and extends the benefits of neural collaborative filtering[6]. Central to the NCACF model is a generative design that employs deep learning techniques for two key tasks: extracting content information from low-level acoustic features and modelling the interaction between users and song embeddings.

The NCACF model is unique in its ability to directly predict the item embedding using a deep content feature extractor, or alternatively, to utilize this feature as a regularization prior. This dual functionality results in two distinct variants of the model: the strict and relaxed. Initial experimental results have demonstrated that the NCACF model achieves state-of-the-art performance in cold start music recommendation tasks , and in warm start setting as well[6], warm start setting refers to the case of recommendation where we have some listening history for items. A notable finding is that the utilization of deep neural networks for learning refined user/item interactions significantly outperforms simpler interaction models in a content-aware framework.

NCACF also utilizes a deep interaction model to capture the nonlinear relationships between users and items instead of a simple matrix product for modeling the interaction between users and items [6]. The deep interaction model is essentially a Multi-Layer Perceptron (MLP), which allows for the learning of complex patterns and relations in the data. However, the model's complexity can become a double-edged sword. When an interaction model larger than three layers is used, the system tends to overfit the training data, diminishing its predictive power for unseen data, which we see in [6] results, leading them to pick an MLP with two hidden layers to test, as it yielded better results during the validation phase than having more hidden layers.

In response to these empirical observations, the primary objective of this study is to refine and optimize the performance of NCACF model. Existing research indicates that models utilizing a higher number of hidden layers have the potential to capture more intricate relationships between input and output variables, contingent upon adequately addressing the issue of overfitting [7]. To that end, the forthcoming investigative endeavors will aim at answering the following research questions:

RQ1 Will the incorporation of dropout into the deep interaction model equipped with four hidden layers lead to a discernible enhancement in the model's performance in the cold start setting compared to the performance of the baseline NCACF's performance without dropout with the same number of layers?

RQ2 Can the proposed model, in its 4-layer interaction model variant, surpass the performance benchmarks established by the baseline model (under optimal settings with three hidden layers)?

1.2 Thesis Outline

The remainder of the thesis is organized as follows:

- **Chapter 2: Theoretical Review** provides a comprehensive review of the literature that led to the creation of Neural Content-Aware Collaborative Filtering (NCACF). It includes sections on Collaborative Filtering (Section 2.1), Matrix Factorization (Section 2.2), WMF and Deep Content Features Extraction (Section 2.4), Deep Learning-based Collaborative Filtering (Section 2.5), and an overview of NCACF (Section 2.6).

- **Chapter 3: Methodology** describes the methodology used in this study to investigate and enhance the NCACF model. This chapter provides an in-depth analysis of the baseline NCACF model (Section 3.1). Section 3.2 provides a detailed description of the experiments conducted, including Environment Setup (Section 3.2.1), Limitations and Constraints (Section 3.2.2), Optimal Variant (Section 3.2.3), Dataset and Preprocessing (Section 3.2.4), Training Details (Section 3.2.5), Performance Measures (Section 3.2.6), Baseline Results and Observations (Section 3.2.7), Improvement Proposal (Section 3.2.8), Research Questions (Section 3.2.9), and Implementation of the Improved Model Architecture (Section 3.2.10).

- **Chapter 4: Results and Discussion** presents the results obtained from the enhanced NCACF model, answers the research questions, and provides an insightful discussion on these findings.

- **Chapter 5: Conclusion and Future Work** offers a conclusion based on the findings of the study and suggests potential directions for future research in this area.

Chapter 2

Theoretical Review

In this section, we trace the evolutionary trajectory leading up to the development of the Neural Content-Aware Collaborative Filtering (NCACF) model. We commence our investigation with an exploration of the rudimentary yet fundamental concept of Collaborative Filtering (CF) (see Section 2.1), laying the groundwork for our discussion. This is succeeded by an exposition of Matrix Factorization (MF) (Section 2.2), where we delve into its inherent strengths and perceived limitations. Our discourse then advances to an examination of Weighted Matrix Factorization (WMF), explaining the need of this progression from regular Matrix Factorization (Section 2.3).

Moving along the continuum of development in recommendation systems, we then discuss the integration of deep content features into WMF as a means of enriching side content, thus adding a nuanced layer of complexity to the model (Section 2.4). Subsequently, we delve into the realm of Deep Learning-based Collaborative Filtering (Section 2.5), illuminating how this method harnesses the power of deep learning algorithms to enhance recommendation systems.

Finally, we providee with a detailed introduction to NCACF (Section 2.6), shedding light on how this state-of-the-art model synthesizes the strengths of its precursors. The intention of this systematic exploration is to underscore the ingenuity of the NCACF model and highlight its unique place within the broader context of recommendation system models.

2.1 Collaborative Filtering

The concept of Collaborative Filtering (CF) forms the backbone of many contemporary recommendation systems.CF methods generate recommendations based on patterns of user behavior, typically through the application of matrix factorization techniques. CF methods are founded on the premise that if two users agree on one issue, they are likely to agree on others as well [8].

The CF techniques are classified into various categories, mainly: Memory-Based CF, Model-Based CF, and finally Hybrid CF.

The neighborhood-based Collaborative Filtering (CF) method, a commonly utilized memory-based CF algorithm, functions according to the following steps: it first computes the similarity or weight, denoted by $w_{i,j}$, signifying the distance, correlation, or weight between two users or items (i and j). It then formulates a prediction for the current user by computing the weighted average of all the ratings of the user or item with respect to a specific item or user, or by employing a basic weighted average. For tasks necessitating the generation of a Top-N recommendation, the method initially identifies the k most similar users or items (nearest neighbors) post similarity calculations. It then amalgamates these neighbors to derive the Top-N most recurrent items for recommendation. [9]

Memory-based CF is easy to implement and simple, but it suffers from being highly dependent on human ratings, as well as the cold start problem. It is also limited by the sparsity of the rating matrix, and the limited scalability for large datasets [8].

Model-Based CF seems to solve some of these problems, as it better addresses the sparsity and scalability problem, and it also improves the prediction performance[8]. Model-based CF uses the users feedback whether explicit(provided by users in form of likes or ratings), or implicit (such as playcounts) to make predictions on user preferences. Examples of Model-Based CF include Bayesian Belief nets, Clustering CF and Matrix Factorization.

2.2 Matrix Factorization

Following the exploration of collaborative filtering, a significant development was the introduction of matrix factorization (MF) techniques. Matrix factorization can be viewed as an advanced model-based form of collaborative filtering, specifically designed to handle the sparsity and scalability issues that can hamper simpler collaborative filtering approaches [8]. The fundamental idea behind matrix factorization is to represent the user-item interaction matrix as the product of lower-dimensionality latent factor matrices.

Say we have a big data matrix $Y \in R^{U \times I}$, showing the interactions between a collection of U users and I items. Here, $y_{u,i}$ symbolizes the interaction between user u and item i. The principle of Matrix Factorization, as proposed by [10] and [11], involves decomposing the data Y into the multiplication of two lower-dimension factors. These include a transposed matrix of user preferences, $W \in R^{K \times U}$, and a matrix of item attributes, $H \in R^{K \times I}$, such that $Y \approx W^T H$. The rank K of the decomposition is selected such that $K(U + I) \ll UI$, thereby ensuring a substantial reduction in dimensionality.

This powerful technique of matrix factorization has shown its efficacy in handling high-dimensional, sparse data and providing personalized recommendations. Nonetheless, MF techniques are often used in cases with explicit ratings, which lead to the development of Wieghted Matrix Factorization (WMF).

2.3 Weighted Matrix Factorization

Weighted Matrix Factorization (WMF), as introduced by [12] and [13], is a specialized form of matrix factorization, suited for processing implicit feedback data. In this model, factorization is conducted on a binary interaction data matrix R where $r_{u,i} = 1$ if there's an interaction between user u and item i, and 0 otherwise.

Estimation of the WMF model can be performed using the Alternating Least Squares (ALS) method, referenced in works by [11] and [14]. The ALS algorithm allows for the derivation of closed-form updates for user and item factors. This method alternates between fixing the user factors and solving for the item factors, and vice versa, which simplifies the complex problem into a series of easier ones. Once the user and item factors are estimated, the recommendation can be accomplished using the predicted interaction defined as $\hat{r_{u,i}} = w_u^T h_i$. A recommendation list for each user is created by selecting items with the highest predicted interaction. Despite WMF's effective performance in recommendation systems, as shown by [11] , it's limited to songs with available listening history, thereby facing the cold-start problem.

2.4 WMF and Deep Content Features Extraction

To mitigate the cold-start issue mentioned earlier, numerous studies have suggested the utilization of auxiliary content information about songs within a factorization-based framework. Some research has relied on closed-form models such as matrix co-factorization [15]. However, our focus lies on techniques that employ deep learning in combination with WMF, as proposed by [16] and [4], with these approaches illustrated in Figure 2.1 and discussed below.

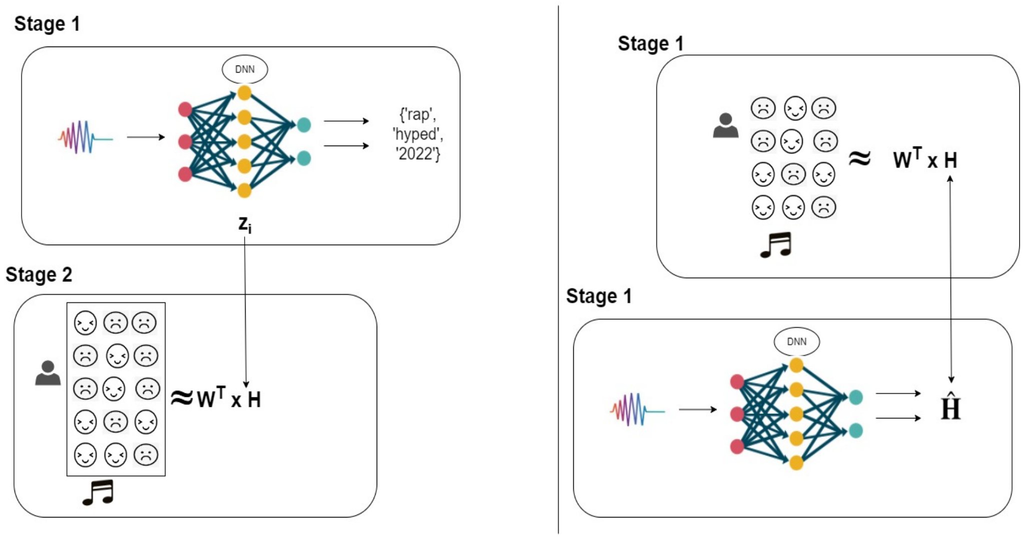

Figure 2.1: Content-aware WMF methods using deep content features. [16] (left) extract content features using an auto-tagging DNN and then incorporated it in WMF. [4] (right) first obtain the WMF decomposition, and then use the learned attribute matrix as target for training the deep content extractor

In their study, [16] propose embedding some content information into WMF. Specifically, they contemplate a prior on the item attribute matrix in the form $h_i \approx \Phi z_i$, where z_i is the content feature vector and Φ linearly maps this vector into the item embedding space. For the content feature vector z_i, the authors extract the last hidden representation from a pre-trained auto-tagging Deep Neural Network (DNN). This strategy helps to address the cold-start problem: even for new items lacking listening history, binarized playcounts can be predicted using $\hat{r_{u,i}} = w_u^T \Phi z_i$, allowing for recommendation generation. Nonetheless, the quality of the content features largely depends on the performance of the auto-tagging network. While these methods have seen significant improvements in recent years[17] [18], they remain constrained by the noisy nature of user-annotated tags. Moreover, the derived features may not necessarily be pertinent for the task of recommendation, given that they are neither estimated together with the user/item embeddings nor specifically fine-tuned for this specific task.[6]

In [4] we see the authors employ an alternative approach for content-based recommendation utilizing WMF. Initially, the factors W and H are derived from the listening data using ALS updates, as outlined in Sect. 2.3. Subsequently, they train a Deep Neural Network (DNN) ϕ_θ with parameters θ, which aims to map input acoustic features x_i to the pre-estimated attributes h_i. This DNN can then be utilized for predicting item attributes for new songs, enabling recommendations in a cold-start situation via $\hat{r_{u,i}} = w_u^T \phi_\theta(x_i)$. However, this strategy significantly hinges on the accurate estimation of the WMF factors W and H, which are essential for the training of ϕ_θ. As the authors don't alternate updates on the factors and the network parameters, the quality of the initial factorization inherently poses a performance constraint on this method.

In essence, these techniques are predominantly restricted by their two-stage operation, implying that the content feature extractor and the factorization model are not jointly trained. Consequently, in [16] the features extracted might not be entirely relevant for the task of recommendation, and in [4], the effectiveness of the deep content extractor is intrinsically limited by the initial factorization.

2.5 Deep Learning-based Collaborative Filtering

As we progress towards more sophisticated recommendation models, the employment of deep learning in collaborative filtering-based recommender systems has taken center stage[19]. Rather than being limited to the shallow embeddings and linear interaction models of matrix factorization, deep learning paves the way for more nuanced user/item embeddings [20] and complex interaction models [21] [22].

A key example of this advancement is the Neural Collaborative Filtering (NCF) framework introduced by [21]. NCF adapts the standard collaborative filtering methodology by incorporating a deep neural network to learn the intricate interactions between user and item embeddings. Specifically, in their proposed model termed Generalized Matrix Factorization (GMF), the predicted user-item interactions are modeled as:

$$\hat{r}_{ui} = \sigma(a^T(w_u \odot h_i)), \tag{2.1}$$

where σ is a non-linear activation function, replacing the identity function employed in the standard matrix factorization, and a is a weight vector.

In another model variant, they consider a multi-layer perceptron (MLP) atop a concatenated user and item factor input, further enhancing the capacity to capture complex interactions between user and item factors.

MLP represents a specific type of artificial neural network characterized by having at least three node layers: an input layer, one or several hidden layers, and an output layer. All nodes in a given layer establish connections with every node in the subsequent layer, with specific weights allocated to these connections. The neurons utilize a nonlinear function—usually a Sigmoid function or a Rectified Linear Unit (ReLU)—to the sum of inputs that have been subjected to weight adjustments. This result is subsequently forwarded to the neurons in the following layer. The MLP's structure is conducive to learning sophisticated, non-linear correlations between inputs and outputs, empowering it to tackle problems that are not linearly separable. MLPs use backpropagation—a supervised learning technique—for training. Coupled with an optimization procedure such as Gradient Descent(GD), backpropagation enables the MLP to refine the weights based on output error, calculated as the deviation between the predicted and actual output. [23]

The integration of these two architectures culminates in the Neural Matrix Factorization (NeuMF) model, which combines the advantages of linearity of GMF and non-linearity of MLP for modeling user-item latent structures.

The NCF framework has demonstrated superiority over shallow matrix factorization models, providing more accurate recommendations. However, NCF in its original form still struggles with the cold-start problem, as it does not incorporate any supplementary content information to support in case of new items.

While attempts have been made to incorporate content information into this framework [24], they have yet to fully leverage the expansive potential of deep learning. Furthermore, while the NCF approach has shown its promise for traditional collaborative filtering tasks, its performance in cold-start recommendation scenarios leaves room for improvement.

2.6 Neural Content-Aware Collaborative Filtering

Presented by [6] ,the strategy of Neural Content-Aware Collaborative Filtering (NCACF) was introduced to address the shortcomings of existing collaborative filtering techniques in dealing with the cold-start problem. Essentially, NCACF is a comprehensive framework that extends the Neural Collaborative Filtering (NCF) model by assimilating a content-aware aspect. It leverages deep learning for the dual purpose of refining user/item embeddings and extracting content information from low-level acoustic features.

Within the NCACF framework, each item (e.g., a music track) is associated with a vector of acoustic features, denoted as x_i. A Deep Content Feature Extractor (DCFE) is applied to these acoustic features to extract higher-level content features, represented as $\hat{h} = DCFE(x_i)$. The content feature vectors $\hat{h}_i$ and user embeddings w_u then become the inputs to the interaction model, yielding predicted ratings or play counts.

Notably, the DCFE can operate in either a strict or a relaxed mode, creating two variants of the NCACF model. In the strict variant, the DCFE directly predicts item embeddings, that is $h_i = DCFE(x_i)$, where h_i is the item latent vector. In the relaxed variant, the DCFE serves as a regularization prior, and the final item embedding h_i is a combination of the content feature vector and a learned latent vector.

NCACF also utilizes a deep interaction model, inspired by NCF [21], to capture the nonlinear relationships between users and items instead of a simple matrix product for modeling the interactions between users and items. The deep interaction model is essentially a Multi-Layer Perceptron which allows for the learning of complex patterns and relations in the data.

The effectiveness of NCACF has been demonstrated in the domain of cold-start music recommendation, where it has reached state-of-the-art results. It also showed state-of-the-art results in the warm setting. However, there are areas within the model that can be improved, specifically , when the interaction model's MLP contains more than 3 hidden layers, the model's complexity can act as both an advantage and a detriment. When an interaction model larger than three layers is used, the model tends to overfit the training data, diminishing its predictive power for unseen data. And this thesis aims to scrutinize and enhance the performance and applicability of the NCACF approach in the context of cold-start scenarios.

In terms of performance, both the strict and relaxed variants of NCACF demonstrate significant improvements over traditional models in cold-start music recommendation tasks. The relaxed variant, in particular, tends to achieve superior results, as it retains the flexibility of learning a part of the item embeddings directly from the data, in addition to leveraging content information. The additional degree of freedom provided by the learned latent vector z_i tends to enhance the adaptability of the model, resulting in better overall performance.

Empirical results from the authors of [6] show that the NCACF models outperform their shallow counterparts in various metrics, providing concrete evidence for the benefit of integrating deep learning in user/item interactions and deep content-aware recommendation models, especially for the cold start setting. Nevertheless, there is a potential for further performance gains, and this thesis will dive deeper into strategies to optimise these interactions and to explore other areas of improvement within the NCACF framework.

Chapter 3

Methodology

The goal of this chapter is to provide a comprehensive account of the experimental methodology and procedures undertaken during this study, exhibiting a systematic approach to investigating and enhancing the Neural Content-Aware Collaborative Filtering (NCACF) model. The experiments are sequential, following the research's chronological progression and mirroring the different stages it went through.

In Section 3.1 NCACF is discussed in details. Subsequently, Section 3.2 unfolds the methodological framework of the experiments. The discourse initiates in Section 3.2.1 with detailed elaboration on the environment in which the experiments were conducted.The following Section 3.2.2 discusses "Limitations and Constraints" encountered throughout the study. This involves discussing the constraints faced while trying to reimplement and enhance the original NCACF model which enforced some decisions regarding the experimentation process. This is succeeded by Section 3.2.4, titled "Dataset and Preprocessing," wherein the sources of the dataset used in the study are shown and the steps taken for data preprocessing are explained. Section 3.2.5, "Training Details," explores the specifics of the training phase. Afterward, "Performance Measures" is discussed in Section 3.2.6, conveying the evaluation metrics utilized to determine the model's performance. Section 3.2.7, "Baseline Results and Observations," reports the outcomes obtained from the baseline NCACF model implementation, paving the way for the development of an improvement proposal. This proposal, discussed in Section 3.2.8 as "Improvement Proposal: Architecture Update and Dropout Layer Addition," advocates for an enhancement to the model based on baseline results, focusing on the integration of dropout into the architecture.

In Section 3.2.9, "Research Questions," a connection is established between the initial research questions, as outlined in the introduction, and the subsequent experiments conducted. Concluding the section, in 3.2.10, "Implementation of the Improved Model Architecture," details of the enhanced model are provided.

Section 3.2.10, "Model Setup," covers the modifications applied to the architecture and their deviations from the baseline NCACF model. Finally, in Section 3.2.10, "Evaluation," the evaluation process of the updated NCACF model against the baseline model is delineated.

This comprehensive overview provides a thorough understanding of the systematic approach adopted in the study, ensuring transparency and reproducibility. The results of the experiments inform the subsequent discussions and evaluations in later chapters.

3.1 Baseline Model: NCACF

In this section, we look at the underlying concepts and methodologies of the NCACF model shown in Figure 3.1. The NCACF framework is a robust and flexible model that addresses many of the common issues encountered with traditional CF methods. It optimizes a CF model based on maximum a posteriori probability with a novel deep content feature extractor, offering significant improvements in the cold start scenario. The discussions will encompass four main aspects of the NCACF model: its structure, the estimation, the recommendation process, the unified learning approach, the incorporation of the deep interaction model, and the selection of optimal model variants.

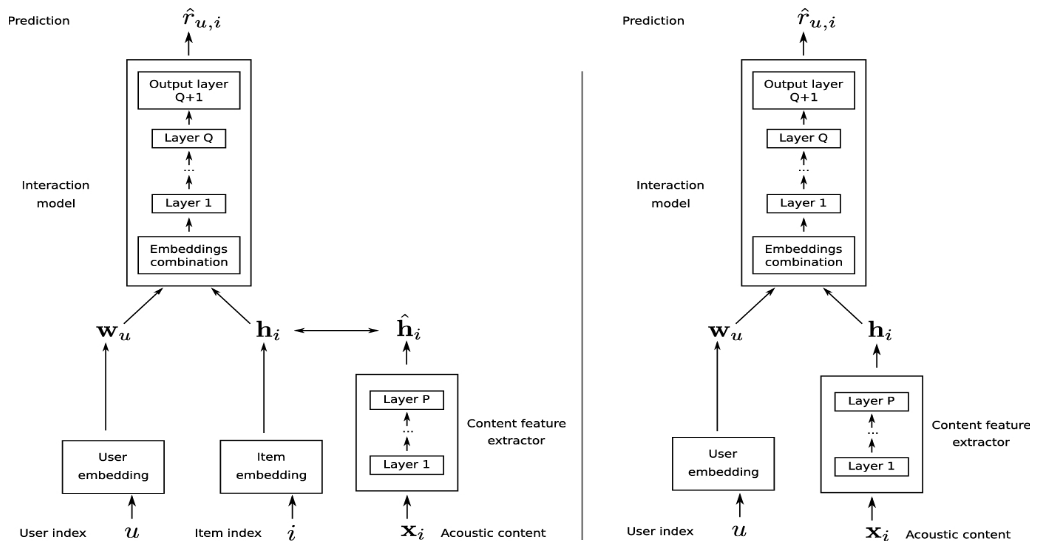

Figure 3.1: NCACF model as proposed by authors in its relaxed variant (left) and strict variant (right)

3.1.1 Data

The data available from the source comes in form of implicit feedback data represented by a playcount matrix $Y \in \mathbb{R}^{U \times I}$. To effectively handle the over-dispersion in this sort of data, it is standard practice to use a binarized playcount matrix R, rather than the raw listening history. Matrix R indicates if a user has listened to a song more than a certain threshold τ (as suggested by [16], [25]; [26]):

$$\forall(u,i), \quad r_{u,i} = \begin{cases} 1 & \text{if} \quad y_{u,i} \geq \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

To retain some information about the raw playcounts (which is discarded when binarization is applied), confidence is used and calculated as follows:

$$\forall(u,i), \quad c_{u,i} = 1 + \alpha \log\left(1 + \frac{y_{u,i}}{\epsilon}\right), \tag{3.2}$$

where α and ϵ are hyperparameters. This function assures that the confidence score increases with the playcount, while the logarithmic term mitigates the influence of extremely large playcounts.

The authors of NCACF used the typical values for these parameters, $\alpha = 2$ and $\epsilon = 10^{-6}$, backed by the recommendations in [27], [16].

3.1.2 Generative Process

The authors of NCACF propose modeling the binary playcounts as an outcome of the interaction between the factors of the user and the item, a principle akin to the Weighted Matrix Factorization (WMF). A Gaussian generative model is chosen to represent these interactions, which allows for the distribution of the binary playcounts to be characterized by a mean and a standard deviation. This model is defined as follows:

$$\forall(u,i), \quad r_{u,i} \sim \mathcal{N}(\psi_\gamma(w_u, h_i), c_{u,i}^{-1}) \tag{3.3}$$

In Equation 3.3, $c_{u,i}$ symbolizes the confidence, which is defined in Equation 3.2 and ψ_γ signifies the interaction model which will be discussed in Section 3.1.7.

The user factor in the model is governed by the following prior, in an approach similar to that taken by [16]:

$$\forall u \in \{1, ..., U\}, \quad w_u \sim \mathcal{N}(0, \lambda_W^{-1} I_K) \tag{3.4}$$

In Equation 3.4, λ_W is a regularization hyperparameter that helps prevent overfitting by imposing a penalty on the magnitude of the model parameters.

To incorporate content information, the authors introduce an additional assumption about the item factor. Initially, they propose a relaxed version where the content information is incorporated as a prior:

$$\forall i \in \{1, ..., I\}, \quad h_i \sim \mathcal{N}(\phi_\theta(x_i), \lambda_H^{-1} I_K) \tag{3.5}$$

However, the authors of NCACF also propose a stricter formulation where the item attribute is directly predicted by the deep content feature extractor:

$$\forall i \in \{1, ..., I\}, \quad h_i = \phi_\theta(x_i) \tag{3.6}$$

Equations 3.5 and 3.6 depict the relaxed and strict scenarios, with ϕ_θ as the deep content feature extractor and x_i as a vector of low-level acoustic features for item i. λ_H is another regularization hyperparameter.

3.1.3 Deep Content Feature Extractor

The authors employ a Multilayer Perceptron (MLP) architecture for the deep content feature extractor, denoted by ϕ_θ. This MLP consists of P layers, which can be formally described as:

$$\phi_\theta(x_i) = \phi^P(\phi^{P-1}(...\phi^1(x_i))) \tag{3.7}$$

In Equation 3.7, x_i represents a set of low-level acoustic features, and ϕ^p indicates the p-th layer of ϕ_θ. Each ϕ^p is a function of some underlying parameters, θ^p, but these are not explicitly mentioned for the sake of simplicity.

Each layer, ϕ_p, in the MLP can be defined by:

$$\forall p \in \{1, ..., P\}, \quad \phi^p(z) = \sigma_p(A^p z + b^p) \tag{3.8}$$

In Equation 3.8, A^p, b^p, and σ^p denote the weights, biases, and activation function of the p-th layer, respectively. The activation function, σ_p, is chosen as the Rectified Linear Unit (ReLU) function for all layers except for the last one, which uses the identity function. The choice of ReLU is inspired by its superior performance demonstrated in various studies and its ability to alleviate gradient vanishing and overfitting issues, as noted by [28].

Preliminary validation experiments and previous work (such as that by [16]) backed the authors choice to employ 1024 neurons in each layer, with the exception of the final layer which outputs the item factor of dimension K. A total of $P = 3$ layers were utilized in [6].

3.1.4 Estimation

In the context of optimizing the complete model according to maximum a posteriori probability, we encounter two variants: the relaxed variant and the strict variant.

For the relaxed variant, the optimization problem can be expressed as:

$$\min_{\xi_R} L_{\text{NCACF-R}}(\xi_R) = \sum_{u,i} c_{u,i}(r_{u,i} - \psi_\gamma(w_u, h_i))^2 + \lambda_W \sum_u |w_u|^2 + \lambda_H \sum_i |h_i - \phi_\theta(x_i)|^2 \tag{3.9}$$

where $\xi_R = \theta,\ \gamma,\ W,\ H$ encompasses the entire parameter set. Here, $c_{u,i}$ is the confidence level, $r_{u,i}$ the interaction between user u and item i, ψ_γ the interaction model, w_u the user factor, h_i the item factor, ϕ_θ the deep content feature extractor, and x_i a vector of low-level acoustic features for item i. The terms λ_W and λ_H are regularization hyperparameters that prevent overfitting by imposing a penalty on the magnitude of the parameters.[6]

For the strict variant, the optimization problem is similar but directly incorporates the deep content feature extractor in the interaction model:

$$\min_{\xi_S} L_{\text{NCACF-S}}(\xi_S) = \sum_{u,i} c_{u,i}(r_{u,i} - \psi_\gamma(w_u, \phi_\theta(x_i)))^2 + \lambda_W \sum_u |w_u|^2 \tag{3.10}$$

with $\xi_S = \theta,\ \gamma,\ W$.

Optimization approaches for problems (3.9) and (3.10) generally rely on a gradient descent strategy.

3.1.5 Recommendation

After the learning phase, the model can be utilized for addressing the cold start recommendation scenario by evaluating the expected binary interactions for each user-item pair as follows:

$$\hat{r}_{u,i} = \psi_\gamma(w_u, \phi_\theta(x_i)). \tag{3.11}$$

The suggested model is not only useful for cold start scenarios but can also serve in standard collaborative filtering applications that do not face the cold start issue. The computation of recommendations in warm start conditions remains the same as in Equation 3.11 for the strict variant. Yet, in the case of the relaxed variant, following the practices in existing literature [16], it's usually preferred to directly leverage the item embedding obtained from the collaborative filtering, denoted as h_i, which was also confirmed by the authors of NCACF [6]. Consequently, the recommendation computation can be expressed as:

$$\hat{r}_{u,i} = \psi_\gamma(w_u, h_i), \tag{3.12}$$

This approach was utilized by the authors of [6] in the warm start setting experiments.

3.1.6 Unified Learning

The training of the NCACF model is centered around the utilization of a Gradient Descent (GD) algorithm, a well-established optimization algorithm often used in machine learning and data science. The essence of the GD algorithm lies in iteratively adjusting the parameters of a model to minimize a defined function, typically a loss or cost function. [29]

In the context of the NCACF model, employing a singular GD algorithm facilitates a unified learning strategy, ensuring a comprehensive and harmonized adjustment of all model parameters. This is advantageous as it enables simultaneous learning and adjustment, fostering a more cohesive interplay between the variables. As a result, this method could potentially increase the accuracy and efficiency of the learning process, by mitigating issues that might arise from using different algorithms for different parameters.[6]

3.1.7 Deep Interaction Model

NCACF leverages a deep interaction model instead of the shallow dot product in order to learn more refined interactions between the embedding vectors [21] [20] as seen in Figure 3.1.

In the first step, the embeddings associated with the user and item are merged into a single vector. Building on the work of[16], [21], [20], the authors suggest two methods for combining these embeddings, specifically through multiplication or concatenation:

$$\forall(u,i), \quad v_{u,i} = \begin{cases} w_u \odot h_i, & \text{(multiplication)} \\ \begin{bmatrix} w_u \\ h_i \end{bmatrix}, & \text{(concatenation)} \end{cases} \tag{3.13}$$

The outcome is a combined vector $v_{u,i}$ of length K', which equates to K in the case of multiplication or $2K$ when concatenation is employed. Previous research such as [22] has indicated that the concatenation method generally provides superior results compared to multiplication, irrespective of the depth and size of the subsequent network layers. Nevertheless, the experiments conducted by the authors of NCACF showed otherwise, given the cold start setting. Results have shown that multiplication yielded better results [6] as will be discussed in Section 3.2.3.

This combined vector serves as an input to an MLP with Q hidden layers to yield the predicted binarized playcounts. The method used to calculate the binarized playcounts resembles the deep content feature extractor:

$$r^{u,i} = \psi^{Q+1}(\psi^{Q}(\dots\psi^{1}(v_{u,i}))), \tag{3.14}$$

In Equation 3.14, ψ^q represents the q-th layer of the network ψ_γ, defined as follows:

$$\forall q \in \{1, \dots, Q+1\}, \quad \psi^q(z) = \sigma^q(A^q z + b^q), \tag{3.15}$$

In Equation 3.15, A^q, b^q, and σ^q denote the weights, biases, and activation function of the q-th layer, respectively. Despite the similarity in notation to the deep content feature extractor ϕ_θ, the corresponding parameters are distinct.

The final layer, or output layer, deviates from the standard structure in that it excludes biases (i.e., $b_{Q+1} = 0$) and employs a single neuron to provide the predicted binarized playcount of dimension 1. The authors chose ReLU as the activation function for the Q hidden layers and the sigmoid function for the output layer to ensure that the predicted binarized playcounts fall within the 0 to 1 range[6][21][22].

To determine the appropriate number of neurons for each hidden layer, the authors of NCACF adopt the "tower" structure, frequently used in deep collaborative filtering models [30] [22]. This structure involves halving the size of each subsequent higher layer to facilitate learning more abstract representations in the upper layers of the network [21]. In practical terms, the q-th hidden layer utilizes $K'/2^{q-1}$ neurons.

These models, labeled as NCACF-Relaxed and NCACF-Strict, are optimized to solve the problems outlined in Equation 3.9 and Equation 3.10, respectively. This is achieved through the implementation of a Gradient Descent (GD) algorithm, leading to the subsequent updates:

$$\xi_R \leftarrow \xi_R - \eta \nabla_{\xi_R} L_{\text{NCACF-R}}(\xi_R), \tag{3.16}$$

and

$$\xi_S \leftarrow \xi_S - \eta \nabla_{\xi_S} L_{\text{NCACF-S}}(\xi_S), \tag{3.17}$$

In Equations 3.16 and 3.17, $L_{\text{NCACF-R}}$ and $L_{\text{NCACF-S}}$ denote the losses as defined in Equation 3.9 and Equation 3.10, respectively.

3.2 Experiments, Materials and Methods

This section will go through the details of the development and experimentation process that lead to the findings, and eventually to the improvement proposal.

3.2.1 Environment Setup

The experimental implementation was carried out utilizing PyTorch, a widely renowned and open-source machine learning library developed by Facebook's AI Research lab (FAIR). PyTorch offers extensive support for GPU-accelerated tensor computations and builds deep neural networks on a tape-based autodiff system. It stands out for its ability to provide maximum flexibility and speed during implementing and building the neural network models, thereby making it a popular choice for deep learning research[31]. All computations were executed using Google Colaboratory Pro, colloquially known as Google Colab Pro, which is an interactive, cloud-based Jupyter notebook environment that allows the writing and execution of Python code through the browser. This environment, hosted by Google, provides access to computing resources including GPUs, making it an advantageous tool for machine learning practitioners, particularly for training neural networks. The GPU provided by Google Colab during computation varies, alternating primarily between Tesla T4 and V100.

3.2.2 Limitations and Constraints

In the process of conducting this research, a number of constraints were encountered. The most significant of these was that my personal device didn't have a GPU. To tackle this, Google Colab's free edition was utilized to run the experiments. However, this quickly proved to be impractical due to the platform's policy of disconnecting if not interacted with every 30-60 minutes. Considering that the training and testing phases required approximately two hours for validation and six hours for testing (one of the down sides of using k-fold validation). The time needed to complete training and testing varies depending on the number of hidden layers, increasing the training time as the hidden layers increase [7]. Maintaining constant interaction was not possible, considering the need to run multiple rounds of training and testing during experiments. Adding to these challenges, The amount of time needed to run multiple runs of training and validation overran the deadline for this study, which imposed a time constraint on our study.

The first approach to overcome these challenge was by upgrading to Google Colab Pro, but this too came with limitations. Specifically, the computing unit cap of this upgraded service was set at 100 units. When these units were rapidly consumed,additional computing units were purchased. to continue the research.

Therefore, there was a need for strategic decisions. It was decided that some parameters were not to be validated to save time on the validation phase (more on that in Section 3.2.3. Furthermore, the number of training epochs were reduced from the 100 epochs used in the original study to 70, a number more manageable within the time and GPU constraints.

3.2.3 Optimal Variant

In this section, the focus will be on the selection of optimal variants of the NCACF model in the cold start scenario. Due to constraints regarding time and GPU resources (mentioned in Section 3.2.2), a strategic selection was made in order to ensure meaningful and efficient experimentation.

The authors of NCACF conducted extensive trials, and the results yielded valuable insights to inform the selection of model variants and hyperparameters. One important finding was the contrast between the relaxed and strict variants. It was found that the relaxed variant consistently outperformed the strict variant. Notably, the strict variant was found to be prone to overfitting, particularly as the number of interaction layers increased[6]. As a result, the relaxed variant was selected for this study.

Regarding the interaction models, two approaches were tested: multiplication and concatenation of the user and item embeddings. The outcomes suggested that the multiplication approach outperformed concatenation across all tested variants in the cold start setting, often by a large margin[6]. Therefore, the multiplication approach was chosen for the interaction model in this experiment.

The impact of the regularization hyperparameters λ_W and λ_H on the performance of both relaxed and strict variants was also examined. The performance was found to be relatively unaffected by the choice of these hyperparameters. Given this observation and the need to manage resource constraints, it was decided not to experiment further with these hyperparameters. Instead, the hyperparameters that yielded optimal results in the original paper were used in the course of this study[6].

Lastly, despite the resource limitations, validation was conducted on the depth of the interaction model. The primary objective was to pinpoint the specific depth at which the model begins to exhibit overfitting tendencies. It should be noted that the findings may exhibit variations, as the number of training epochs was intentionally reduced this study.

3.2.4 Dataset and Preprocessing

The source for data used for this study is the Million Song Dataset [32], which is a freely available collection of audio features and metadata for a million contemporary popular music tracks. The dataset was created by Echo Nest, a music intelligence platform, in collaboration with LabROSA, a research lab at Columbia University. It contains information on songs' track IDs, artist names, song titles, and a set of features extracted from the audio signal such as loudness, tempo, and key. Additionally, it provides user listening history in the form of user-song play counts.

The primary dataset used in this study is the Million Song Dataset Taste Profile (MSD-TP) , which contains user-song interactions in the form of play counts. The dataset comprises over 48 million triplets of user ID, song ID, and play count.

In addition to the MSD-TP, the study also utilizes the MSD Allmusic Style/Genre Dataset (MSD-SSD) , which provides the features associated with each song. The dataset contains 168 features that describe various aspects of the songs, such as tempo, key, and timbre. The features are extracted using the Echo Nest Analyzer, and the dataset is stored in Attribute-Relation File Format (ARFF).

In preprocessing we prepare the dataset for effective use in training and evaluating a collaborative filtering model for music recommendation. The preprocessing steps were:

1. Filtering Taste Profile data: The first step in preprocessing involves filtering the Taste Profile dataset. The Taste Profile data is loaded, and only the songs that are present in the unique_tracks.txt file are retained. This file contains a list of song IDs in the dataset. Additionally, any ratings with play counts

below a threshold value (min_c = 7) are removed. Removing these ratings is done to ensure that only reliable data is used in the study, as low play counts might not accurately represent user preferences [26]. Top songs and users are retained, and inactive users and items are removed. (songs which have been listened to by at least 50 users, and users who listened to at least 20 songs.

2. Loading and filtering record features: After filtering the Taste Profile data, the next step is to load and filter the corresponding features from the MSD Allmusic Style/Genre Dataset (MSD-SSD). The unique song IDs from the filtered Taste Profile data are used to extract the relevant features from the MSD-SSD. Once the features are extracted, they are scaled using appropriate techniques (e.g., min-max scaling, standardization) to ensure that they are on a comparable scale. This step helps in improving the performance of machine learning models that are sensitive to feature scales. The features are then stored in a Pandas DataFrame for further processing.

3. Updating Taste Profile records: The Taste Profile dataset is updated to include only the songs with available features. This step ensures that the final dataset has complete information for both user-song interactions and song features. The dataset is then shuffled to randomize the order of records, which helps avoid potential biases during model training. Dictionaries are created to map user and song IDs to integers, which simplifies data representation and handling. These dictionaries are stored as JSON files for future use.

4. Splitting the data: To implement model training and assessment, the authors utilized a 10-fold cross-validation technique [33]. This method, particularly effective in the cold start scenario, reserves 20% of the songs for validation. The remainder of the songs is randomly divided into ten equally distributed subsets, 90% of these songs serve as the training set, and 10% serve as the test set. Playcounts corresponding to the training set are employed to learn the different models. The validation set, held in reserve, is leveraged for hyperparameter tuning, specifically in the course of this experiment, for different depths of the interaction model, and the remaining test set is utilized for model comparisons. Evaluations conducted on the validation and test sets simulate a cold start scenario, given that the models have not been trained on the playcounts corresponding to the songs they encompass. To alleviate computational demands,

the authors tune hyperparameters exclusively on the initial training/testing split. The optimized values derived from this process are subsequently applied to all remaining nine splits during testing. For the cold start recommendation scenario, the splitting strategy implemented is particularly efficient. This strategy involves partitioning the songs instead of non-zero playcounts. By doing so, all songs in the validation and test sets are guaranteed to be songs that the model has not encountered during training. This simulates a realistic cold start scenario, where the recommendation system encounters new items that it has not seen before. For a warm start situation, a different strategy would be necessary, where the non-zero playcounts are split instead of the songs. In that case, it would ensure that all songs in the validation and test sets also have some non-zero interaction in the training set. This strategy would allow the system to make recommendations based on previously observed interactions. However, as the focus of this work is on the cold start problem, the specifics of warm start recommendation strategies are not discussed further. The characteristics of the resulting datasets from this splitting strategy are detailed in Table 3.1.

	Total	**Training**	**Validation**	**Test**
Users	17,572	17,572	17,572	17,572
Songs	13,532	9,743	2,706	1,083
Interactions	509,177	355,891	105,897	47,389

Table 3.1: Dataset statistics for the first split (interactions shows the number of non-zero playcounts)

5. Data Numerization: Finally, the data is numerized to convert user and song IDs to integers using the previously created dictionaries. Generating numerized training, validation, and testing sets for each fold is a necessary as machine learning models in Pytorch expect numerized IDs and can't parse with string IDs as provided in the dataset.

After preprocessing, the dataset consists of 509,177 triplets (user ID, song ID, playcount) from 17,572 users and 13,532 songs, resulting in a density level of 0.214%.

3.2.5 Training Details

In alignment with prior work focused on music recommendation utilizing the same dataset [26][6], the dimensionality for user and item embeddings was designated as $K = 128$. Initialization of user/item embeddings was accomplished with random values originating from a normally distributed, centered set with a standard deviation of 10^{-2}. The deep content feature extractor and deep interaction model parameters, θ and γ, were initialized following the Le Cun's initialization scheme [34], excluding the output layer of the interaction model in NCACF , where weights were initialized to ones. In LeCun initialization, the weights are initialized from a normal distribution with mean 0 and standard deviation $\sqrt{1/N}$, where N is the number of inputs to the neuron (LeCun Normal), or a uniform distribution within the interval $[-\sqrt{3/N}, \sqrt{3/N}]$ (LeCun Uniform) [34]. Gradient Descent was performed employing the Adam algorithm [29] with a learning rate of 10^{-4} and a batch size of 128. Adam, short for Adaptive Moment Estimation, is a popular optimization algorithm used in deep learning tasks. Unlike traditional stochastic gradient descent, which maintains a single learning rate for all weight updates, Adam implements an adaptive learning rate. It accomplishes this by calculating individual adaptive learning rates for different parameters. Adam combines the advantages of two extensions of stochastic gradient descent: AdaGrad and RMSProp. AdaGrad adjusts the learning rate adaptively for each coefficient in the model, amplifying the weights with lower gradients. RMSProp also adapts the learning rate but in a different manner; it uses a moving average of squared gradients to normalize the gradient itself. That gives a balanced step size – not too small, not too large. Consequently, Adam optimizes the best properties of these two algorithms to provide an optimization method that can handle sparse gradients and noisy data[29]. Hyperparameters λ_W and λ_H were set according to the authors of NCACF validation experiments, setting $\lambda_W = .01$ and $\lambda_H = 10$. To conserve computational resources, training went on for a of 70 epochs. The hyperparameters are presented in Table 3.2.

3.2.6 Performance Measure

Normalized Discounted Cumulative Gain

In the experiments, Normalized Discounted Cumulative Gain (NDCG) was employed as articulated by [35], as the evaluation metric for the effectiveness of the

Hyperparameter	**Description**	**Value**
item/user embedding size	size of the embedding of user and item vectors	128
θ initialization	deep content feature extractor parameters (weights and biases)	Le Cun's initialization scheme
γ initialization	deep interaction model parameters (weights and biases)	Le Cun's initialization scheme
batch size	size of batch for training	128
λ_W	user regularization term	0.01
λ_H	item regularization term	10
epochs	number of epochs for training	70
learning rate	learning rate of gradient descent	10^{-4}
optimization	GD optimization algorithm	Adam

Table 3.2: Hyperparameters and their corresponding descriptions and values.

recommendations. For each user u, a list of items is created, sorted based on the anticipated ratings r^u, from the evaluation set (either validation or testing). The relevancy of this list is then calculated in accordance to the actual preferences, i.e., the real user/item interactions in the evaluation set. The relevancy of an item i for a user u as $rel_{u,i}$ is denoted, which is 1 if item i is in user u's listening history (in the evaluation set), and 0 otherwise.

To give priority to recommendations that put the evaluation items at the top of the list, discounted weight to the relevance is applied. This results in the Discounted Cumulative Gain (DCG):

$$DCG_u = \sum_{i=1}^{I'} \frac{rel_{u,i}}{\log 2(i+1)}, \tag{3.18}$$

where I' symbolizes the length of the ranked item list. Ideally, I' should match the number of songs in the evaluation set for a thorough evaluation. However, this causes a substantial increase in computational resources needed for the evaluation. As a result, it is common to only consider the top-I' items [25] . Following earlier research on this dataset [26], the authors chose $I' = 50$ in their tests, and we followed in their steps to replicate the experiments. This choice is practical for a music recommendation system and also significantly reduces computational load.

The NDCG is then calculated by normalizing the DCG:

$$NDCG_u = \frac{DCG_u}{IDCG_u}, \tag{3.19}$$

where IDCG represents the Ideal DCG, that is, the DCG of a perfectly ordered list. Finally, these scores are averaged across users to obtain an overall measure of recommendation performance. The final NDCG scores range from 0 to 1 (with 1 being the best), and these scores will be expressed as percentages for the ease of interpretation in our experiments.

Loss Function

The weighted prediction error was utilized, primarily due to the inherent characteristics of the Gaussian generative model which allows for seamless integration of prior information. This particular capability of the Gaussian generative model has been a key factor motivating previous studies to adopt this framework in the context of content-aware collaborative filtering [36][37]. Furthermore, it enhances the model's ability to ascertain more nuanced and accurate predictions by leveraging pre-existing knowledge. Thus, the use of weighted prediction error provides a significant edge in obtaining more refined model outcomes.

In the learning process for the proposed models, the optimization objective is to minimize a weighted prediction error:

$$\sum_{(u,i)\in B} c_{u,i}(r_{u,i} - \hat{r}_{u,i})^2. \tag{3.20}$$

This task is typically split over batches of data, $B \subset 1, ..., U \times 1, ..., I$, with each batch being processed at each iteration of the gradient descent algorithm.

To alleviate the high computational burden that arises from processing all available data, especially for large datasets, a sampling strategy is employed. This strategy involves considering only batches of items, $B_i \subset 1, ..., I$, for each sample $i \in B_i$. Despite the necessity of using a smaller batch size with this approach, it yields a significantly better performance across all tested methods[6]. Hence, the results obtained using this sampling strategy are presented in this study.

3.2.7 Baseline Results and Observations

The performance of the model was scrutinized under various depths of the interaction model $Q \in \{0, 1, 2, 3, 4\}$, where $Q = 0$ signifies a shallow interaction model, $Q = 1, 2, 3, 4$ means an interaction model with 1,2,3 or 4 hidden layers. This evaluation was conducted using the validation set, the results of which can be found in Figure 3.2. We can observe the NDCG metric commenced at an initial value of 17. As Q incremented from 0 to 3, an evident increase in NDCG was observed, indicating an enhancement in the model's performance with the deep interaction model's layers. At this point, the model achieved an optimal NDCG. Conversely, when Q was further increased to 4, there was a significant decrease in the NDCG value, suggesting a performance dip. [35]

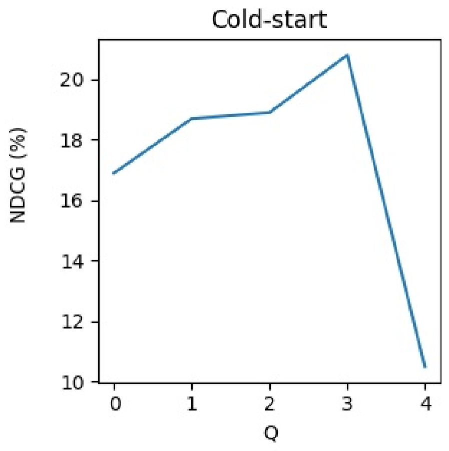

Figure 3.2: Validation Results for Baseline NCACF on various depths (Q) for the deep interaction model

It is worth mentioning that this is not consistent with the results of the authors of NCACF, as they yielded better results at $Q = 2$, the reason for that must be the change in the number of epochs, which had to be cut down to 70 in our experiments due to the limitations (refer back to Section 3.2.2).

To better comprehend this observation, a juxtaposition of training loss against validation NDCG was carried out. As depicted in Figure 3.3, a steady decrease in loss is evident across the span of 70 epochs, despite an observable downward trend in NDCG. This phenomenon strongly suggests that the model tends to overfit as the complexity of the interaction model (i.e., the depth Q) is augmented[7], we tend to see this behaviour when the network matches the data so closely, hence the downward trend of the loss, that it becomes unable to generalize over the data to be tested [7], hence the downward trend of the NDCG [23].

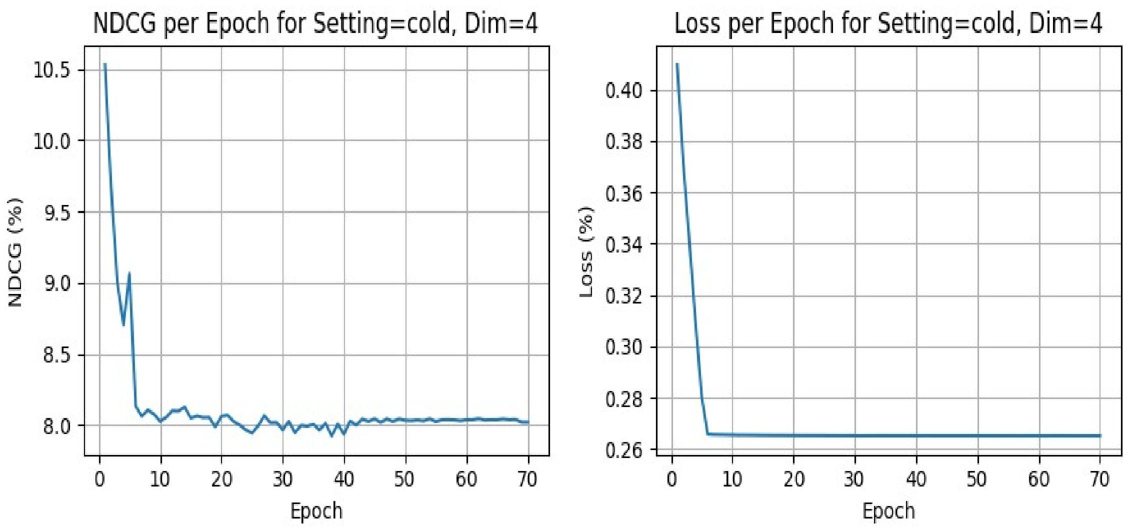

Figure 3.3: NDCG during validation (left) and loss during training (right) for MLP with 4 hidden layers

When the model was subsequently tested for the optimal depth $Q = 3$, the resulting NDCG averaged to 25.5 across all validation splits, reinforcing the previous conclusion regarding the model's optimal complexity. For the sake of experimenting and comparison with the anticipated improvement, the model was also tested for $Q = 4$. As anticipated from the validation performance, the mean NDCG for $Q = 4$ dropped to 18.5 across all splits, further underscoring the observed overfitting issue at this depth.

3.2.8 Improvement Proposal: Dropout Layer Addition

While the increase in the number of hidden layers has been associated with enhanced model performance [7],the findings here indicate the onset of overfitting when transitioning from a three-layer to a four-layer architecture. This is despite employing L2 regularization coefficients for user and item embeddings, and leveraging k-fold splits, in alignment with the approach put forth by the authors of NCACF.

The utilization of a three-layer hidden architecture of the deep interaction model has been observed to yield good results in terms of NDCG in NCACF. However, there is evidence suggesting that models with a higher number of hidden layers can encapsulate more intricate relationships between their inputs and outputs, given the adequate complexity of the problem and an appropriately large dataset, provided that overfitting is kept in check[38][7].

Dropout is a strategy conceived to tackle issues of overfitting and to facilitate the approximation of an amalgamation of diverse neural network architectures efficiently [38]. The principle entails the random omission, or "dropping out," of units (both hidden and visible) within a neural network. Each unit is independently dropped with a fixed probability p, which is typically chosen via a validation set or simply fixed at a value such as 0.5. Notably, for input units, the optimal retention probability is usually closer to 1 than 0.5 [38].

The application of dropout generates a "thinned" network, a subset of the original neural network, which incorporates all units that survived the dropout procedure, as seen in Figure3.4. Given a neural network with n units, one can envision it as a collection of 2^n potential thinned networks. Training a neural network with dropout is akin to training a set of 2^n thinned networks that share weights and are each trained very rarely, if at all [38][39].

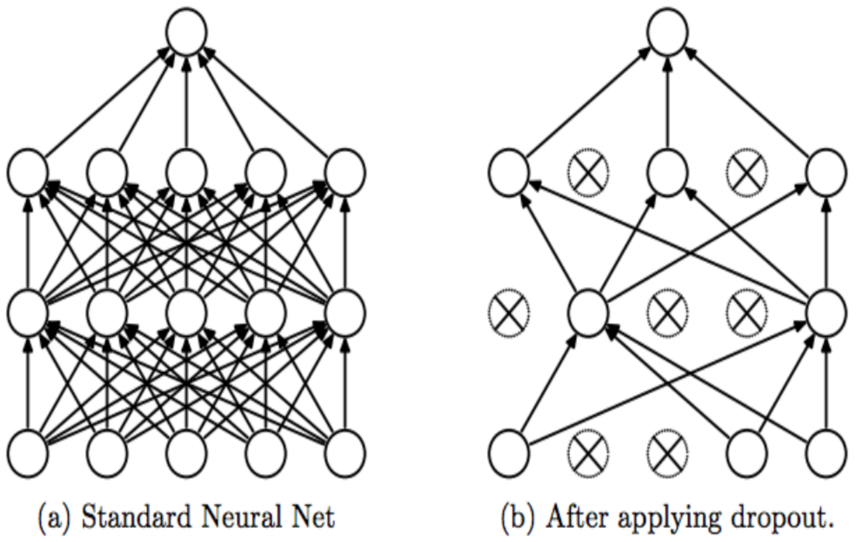

Figure 3.4: A standard neural net before dropout (left), and an example of what a thinned network would look like after dropout (right)

A notable aspect of the dropout method is that it allows each neuron to independently acquire valuable attributes without being overly reliant on the corrective functionality of other neurons [40]. In other words, it reduces co-adaptation among neurons. It is important to highlight that training with dropout is equivalent to the introduction of noise into the neural network, a concept analogous to that employed in Denoising Autoencoders [40].

Upon completion of training, these disparate thinned networks are combined into a single neural network. It is impractical at test time to explicitly average the predictions from exponentially many thinned models. Instead, a straightforward approximate averaging method is utilized, where a single network is employed without dropout, and the weights of this network are scaled-down versions of the trained weights as shown in Figure 3.5. This scaled-down version ensures that for any hidden unit, the expected output is the same as the actual output at test time [38][39]. The application of this method has been shown to yield significantly lower generalization error on a wide variety of classification problems compared to training with other regularization methods [38].

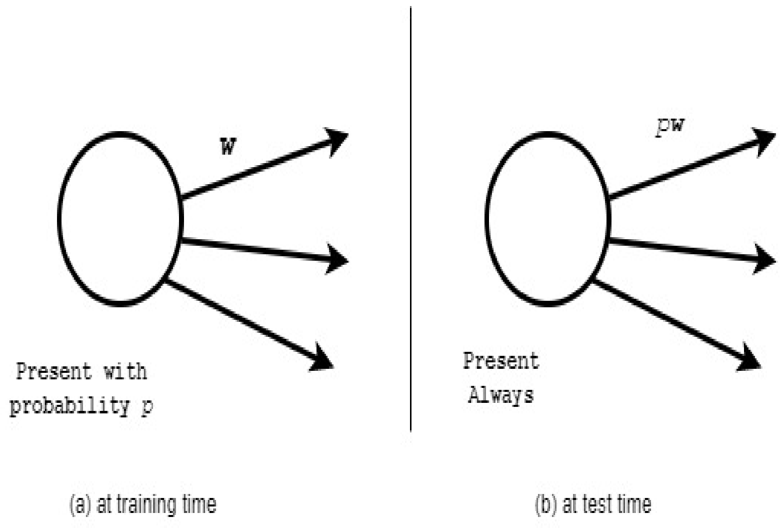

Figure 3.5: A unit at training time which is present with probability p with weights w connecting it to the next layer (left). At test time, the unit will be always present with weights that are multiplied by p.

Previous research indicates that, for deep learning models with a large number of hidden layers, dropout has demonstrated superior results when compared to L2 regularization [40]. Such findings, alongside the theoretical capacity of dropout to harness the benefits of added complexity without incurring significant downsides, have motivated our investigation into the potential advantages of incorporating dropout layers within the deep interaction model.

In consideration of the constraints delineated in Section 3.2.2, our investigation will be confined to the training and evaluation of the model encompassing four hidden layers, as this is the depth where we see the model starting to overfit. This pertains to both the baseline NCACF and the proposed variant of NCACF augmented with dropout. We also pertained results for the NCACF in its optimal case with 3 hidden layers, which can be used for comparison to answer **RQ2**. Opportunities for more extensive experimentation and broader exploration on more values for the interaction model's depth remain a promising avenue for future research.

3.2.9 Research Questions

Building on the previous section, our experiments aim at answering the following research questions, aforementioned in the Introduction Chapter:

RQ1 Will the incorporation of dropout into the deep interaction model equipped with four hidden layers lead to a discernible enhancement in the model's performance in the cold start setting compared to the performance of the baseline NCACF's performance without dropout with the same number of layers?

RQ2 Can the proposed model, in its 4-layer interaction model variant, surpass the performance benchmarks established by the baseline model (under optimal settings with three hidden layers)?

3.2.10 Implementation of the Improved Model Architecture

Our experimental methodology, devised to answer the research questions proposed earlier, can be categorized into three main components: model setup, training, and evaluation.

Model Setup

The primary distinction between the baseline NCACF model and the modified model is the incorporation of dropout layers within the deep interaction model. All other aspects, including the preprocessing and training procedures, were kept constant, thereby facilitating the isolation and assessment of the influence of dropout on the model's performance.

Although the literature suggests using a dropout rate of 0.5 [38], this dropout was fixed at a rate of 0.2, i.e, the probability of a neuron being deactivated is 20%, given that the data we were dealing with was inherently sparse and to avoid potential underfitting[39]. As an opportunity for future work, researchers could include a grid search to find the most suitable dropout value. Dropout was applied to hidden layers and not input, as research has shown that the optimal retention value for the input layer was 1 [38].

The training process did not deviate from the one elucidated in Section 3.2.5. After adding the dropout layers, we proceeded to train the model with the updated architecture.

Evaluation

In order to ascertain the effects of our modification, the updated NCACF was tested with a deep interaction model with four hidden layers. This allows for a direct comparison against the baseline NCACF, also equipped with four layers, and an optimal baseline NCACF model, which has been established to function best with three layers. This comparison strategy will enable us to determine whether the addition of dropout layers can improve the performance of an NCACF model with a greater number of hidden layers.

Chapter 4

Results and Discussion

	Baseline NCACF		NCACF with dropout
	3 layers	4 layers	4 layers
mean NDCG	25.5	18.5	19.7
min NDCG	20.6	12.8	12.7
max NDCG	31	30.2	30

Table 4.1: NDCG test results showing a comparison between the different architectures

The results of testing the model on the dataset are shown in Table 4.1. The mean, minimum, and maximum Normalized Discounted Cumulative Gain (NDCG) values across all test splits for each model variant are presented. Regarding research question **RQ1**, it is pertinent to examine the results for the baseline NCACF with 4 layers, juxtaposed with the NCACF supplemented with dropout. An improvement in the mean NDCG value from 18.5 in the baseline to 19.7 in the modified model suggests that introducing dropout, coupled with L2 regularization, to a deep interaction model with four hidden layers improves its performance in cold-start scenarios. This echoes prior findings, indicating dropout's effectiveness in mitigating overfitting in models with an increased number of hidden layers, while L2 regularization enhances models with fewer layers [40][39]. This can also be seen in the Figure 4.1 where we see NDCG improve while the loss is decreasing as well.

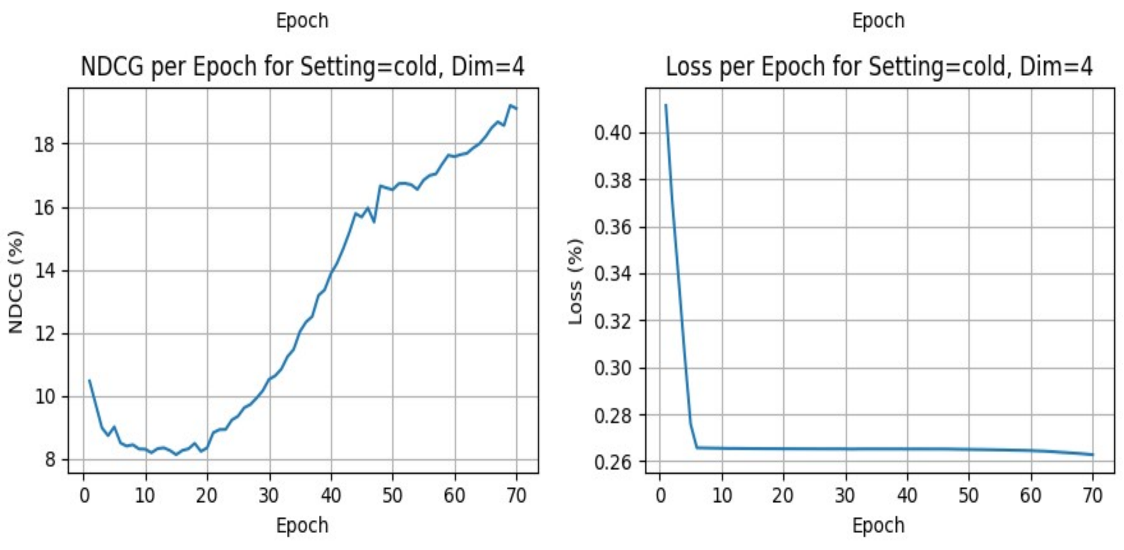

Figure 4.1: NDCG performance on NCACF with dropout (left), training loss results (right)
.

Nevertheless, addressing research question **RQ2**, it becomes evident that the improvements elicited by introducing dropout do not surpass the mean NDCG of 25.5 obtained by the optimal baseline NCACF variant with 3 hidden layers. This could be attributed to several factors, the most plausible of which is that models employing dropout generally require a longer convergence period to achieve optimal accuracy [40]. As established in prior research, dropout's convergence rate is more robust but marginally slower compared to other regularization techniques. This theory is validated by Figure 4.2 where the NDCG progression exhibits a unique trend. In Figure 4.2(a) illustrating the optimal baseline NCACF model with 3 hidden layers, there is a sudden surge in NDCG value around the 10th epoch, followed by a more steady increment. In contrast, Figure 4.2(c), representing the proposed NCACF, depicts a consistent and more robust increase in NDCG across epochs [40].

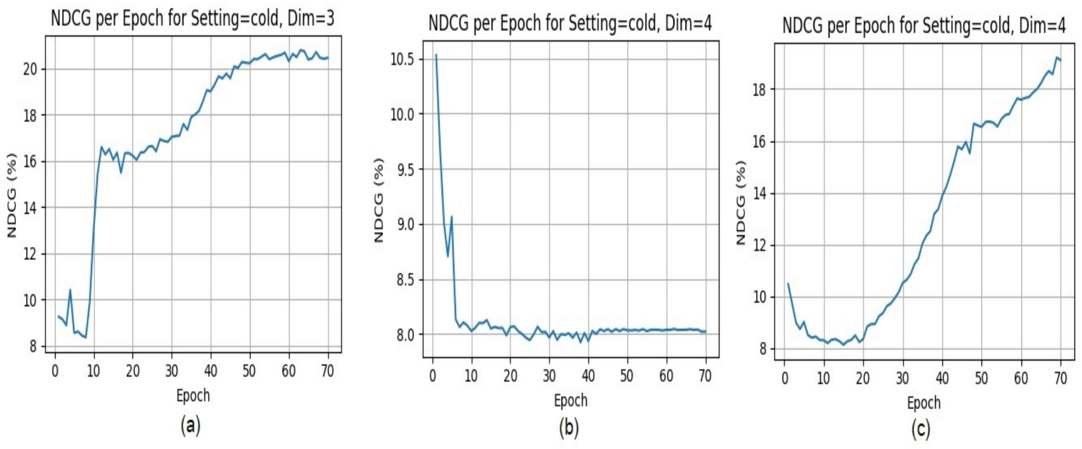

Figure 4.2: Comparison of the advancement of NDCG for different variants across training epochs, (a) is for the optimal NCACF baseline with 3 hidden layers, (b) is for the baseline NCACF with 4 hidden layers, (c) is for the proposed NCACF updated with dropout value 0.2 to all hidden layers
.

The results of our analysis suggest that the slower convergence rate of the NCACF model with dropout may be one of the primary factors affecting its performance. This hypothesis is substantiated by literature in the field, which reports that dropout, while being an effective regularization technique, can lead to a slower convergence rate [41]. Consequently, the optimal accuracy of a model employing dropout may only be achieved after a greater number of training epochs than what has been employed in this study.

In light of this, potential refinements of this research could include extending the training and testing phase across a greater number of epochs. Such a procedure could provide the model with additional opportunity to more fully explore the solution space and potentially converge to a more optimal solution. It is worth noting that Hinton et al. [41] suggest that dropout training could necessitate approximately twice the number of epochs to converge to optimal accuracy when compared to training procedures that do not employ dropout.

This poses an additional consideration for researchers who plan to extend this work or develop similar models. The time and computational resources required for training may significantly increase, particularly in scenarios where the dataset is large or complex. Hence, the choice of employing dropout should be made judiciously, taking into account the trade-off between potential improvements in model performance and the additional computational cost.

Chapter 5

Conclusion and Future Work

In this research work, the impact of dropout integration in the Neural Content-Aware Collaborative Filtering (NCACF) model has been explored , particularly in a cold start setting. This investigation was prompted by the challenge of overfitting that arises as the number of hidden layers in the interaction model increases, a phenomenon that persists despite the original NCACF authors' mitigation attempts employing L2 regularization and k-fold cross-validation.

Through the lens of our research questions, we have endeavored to evaluate the effect of dropout on the performance of the NCACF model with a deep interaction model of four hidden layers, and to compare it with the baseline model under similar conditions (**RQ1**). Our findings demonstrate an appreciable improvement in test results for the modified NCACF model when dropout is applied during training, signaling a successful mitigation of overfitting due to an increased number of hidden layers.

Nonetheless, the adjusted model fails to outperform the baseline NCACF model in its optimal configuration with three hidden layers (**RQ2**). This might be attributed to the longer time required for models incorporating dropout to converge to optimal accuracy, an assertion supported by the trend visible in our results and previous literature [40][41].

It is vital to note that while the incorporation of dropout has shown promise in enhancing the performance of the NCACF model in cold start settings with regards to overfitting, it also introduces additional considerations that bear practical significance. The model's training duration and associated computational resources are among these key factors, pointing towards an implicit trade-off.

These findings, therefore, underscore the importance of thoughtfully considering the balance between potential performance improvements and computational cost in future research. Future model iterations should aim to utilize the advantages of deeper layers while addressing the challenges of overfitting and computational efficiency. This thesis forms a solid foundation for such future investigations, offering valuable insights into the application and effects of dropout within the NCACF framework.

Building upon the findings and insights of this thesis, future research could explore the application of the updated NCACF model in a warm start setting given that it could show improvement on the accuracy of the model.

Moreover, additional explorations might involve experimenting with different dropout rates. Current research literature suggests that there isn't a 'one-size-fits-all' dropout rate, effective across all datasets and model architectures [39][38][40]. As such, it might prove beneficial to identify the optimal dropout rate for specific configurations, further honing the model's performance.

These prospective lines of inquiry align with the overarching objective of continually enhancing the NCACF model, supporting its robustness and accuracy across varying settings. The framework and observations provided in this thesis thus serve not only as a valuable contribution to the existing knowledge pool, but also as a springboard for further advancements in this field.

Bibliography

[1] Markus Schedl et al. "Current challenges and visions in music recommender systems research". In: *International Journal of Multimedia Information Retrieval* 7.2 (Apr. 2018), pp. 95–116. DOI: 10.1007/s13735-018-0154-2. URL: https://doi.org/10.1007\%2Fs13735-018-0154-2.

[2] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative Filtering for Implicit Feedback Datasets". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 263–272. DOI: 10.1109/ICDM.2008.22.

[3] Kazuyoshi Yoshii et al. "Hybrid Collaborative and Content-based Music Recommendation Using Probabilistic Model with Latent User Preferences." In: Jan. 2006, pp. 296–301.

[4] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. "Deep content-based music recommendation". In: *Advances in neural information processing systems* 26 (2013).

[5] Xinxi Wang and Ye Wang. "Improving Content-Based and Hybrid Music Recommendation Using Deep Learning". In: MM '14. New York, NY, USA: Association for Computing Machinery, 2014, 627–636. ISBN: 9781450330633. DOI: 10.1145/2647868.2654940. URL: https://doi.org/10.1145/2647868.2654940.

[6] Paul Magron and Cédric Févotte. "Neural content-aware collaborative filtering for cold-start music recommendation". In: (2022). arXiv: 2102.12369 [cs.IR].

[7] Muhammad Uzair and Noreen Jamil. "Effects of Hidden Layers on the Efficiency of Neural networks". In: *2020 IEEE 23rd International Multitopic Conference (INMIC)*. 2020, pp. 1–6. DOI: 10.1109/INMIC50486.2020.9318195.

[8] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. “Matrix factorization model in collaborative filtering algorithms: A survey”. In: *Procedia Computer Science* 49 (2015), pp. 136–146.

[9] Xiaoyuan Su and Taghi M Khoshgoftaar. “A survey of collaborative filtering techniques”. In: *Advances in artificial intelligence* 2009 (2009).

[10] Benjamin Marlin and Richard Zemel. “The multiple multiplicative factor model for collaborative filtering”. In: Jan. 2004. DOI: 10.1145/1015330.1015437.

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. “Collaborative Filtering for Implicit Feedback Datasets”. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 263–272. DOI: 10.1109/ICDM.2008.22.

[12] Ruslan Salakhutdinov and Andriy Mnih. “Bayesian probabilistic matrix factorization using Markov chain Monte Carlo”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 880–887.

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems”. In: *Computer* 42.8 (2009), pp. 30–37. DOI: 10.1109/MC.2009.263.

[14] Xiangnan He et al. “Fast matrix factorization for online recommendation with implicit feedback”. In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2016, pp. 549–558.

[15] Olivier Gouvert, Thomas Oberlin, and Cédric Févotte. “Matrix co-factorization for cold-start recommendation”. In: *19th International Society for Music Information Retrieval Conference (ISMIR 2018)*. 2018, pp. 1–7.

[16] Dawen Liang, Minshu Zhan, and Daniel PW Ellis. “Content-Aware Collaborative Music Recommendation Using Pre-trained Neural Networks.” In: *ISMIR*. 2015, pp. 295–301.

[17] Taejun Kim, Jongpil Lee, and Juhan Nam. “Sample-level CNN Architectures for Music Auto-tagging Using Raw Waveforms”. In: (Oct. 2017).

[18] Jordi Pons et al. *End-to-end learning for music audio tagging at scale*. 2018. arXiv: 1711.02520 [cs.SD].

[19] Shuai Zhang et al. "Deep learning based recommender system: A survey and new perspectives". In: *ACM computing surveys (CSUR)* 52.1 (2019), pp. 1–38.

[20] Hong-Jian Xue et al. "Deep matrix factorization models for recommender systems." In: *IJCAI*. Vol. 17. Melbourne, Australia. 2017, pp. 3203–3209.

[21] Xiangnan He et al. *Neural Collaborative Filtering*. 2017. arXiv: `1708.05031 [cs.IR]`.

[22] Wanyu Chen et al. "Joint neural collaborative filtering for recommender systems". In: *ACM Transactions on Information Systems (TOIS)* 37.4 (2019), pp. 1–30.

[23] Haidong Li et al. "Research on Overfitting of Deep Learning". In: *2019 15th International Conference on Computational Intelligence and Security (CIS)*. 2019, pp. 78–81. DOI: `10.1109/CIS.2019.00025`.

[24] Jianxun Lian et al. "CCCFNet: A Content-Boosted Collaborative Filtering Neural Network for Cross Domain Recommender Systems". In: Apr. 2017, pp. 817–818. DOI: `10.1145/3041021.3054207`.

[25] Dawen Liang et al. *Modeling User Exposure in Recommendation*. 2016. arXiv: `1510.07025 [stat.ML]`.

[26] Viet-Anh Tran et al. "Improving Collaborative Metric Learning with Efficient Negative Sampling". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, July 2019. DOI: `10.1145/3331184.3331337`. URL: `https://doi.org/10.1145\%2F3331184.3331337`.

[27] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative Filtering for Implicit Feedback Datasets". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 263–272. DOI: `10.1109/ICDM.2008.22`.

[28] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[29] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]`.

[30] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[31] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[32] Thierry Bertin-Mahieux et al. “The million song dataset”. In: (2011).

[33] Arthur Flexer. “Statistical evaluation of music information retrieval experiments”. In: *Journal of New Music Research* 35.2 (2006), pp. 113–120. DOI: `10.1080/09298210600834946`. URL: `https://doi.org/10.1080/09298210600834946`.

[34] Yann A. LeCun et al. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Springer Berlin Heidelberg, 2012, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: `10.1007/978-3-642-35289-8_3`. URL: `https://doi.org/10.1007/978-3-642-35289-8_3`.

[35] Yining Wang et al. *A Theoretical Analysis of NDCG Type Ranking Measures*. 2013. arXiv: `1304.6480 [cs.LG]`.

[36] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. “Collaborative Deep Learning for Recommender Systems”. In: KDD ’15. New York, NY, USA: Association for Computing Machinery, 2015, 1235–1244. ISBN: 9781450336642. DOI: `10.1145/2783258.2783273`. URL: `https://doi.org/10.1145/2783258.2783273`.

[37] Huo Huan et al. “Collaborative filtering recommendation model based on convolutional denoising auto encoder”. In: *Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*. 2017, pp. 64–71.

[38] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.

[39] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.

[40] Ekachai Phaisangittisagul. “An Analysis of the Regularization Between L2 and Dropout in Single Hidden Layer Neural Network”. In: *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*. 2016, pp. 174–179. DOI: `10.1109/ISMS.2016.14`.

[41] Geoffrey Hinton et al. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. arXiv: `1207.0580 [cs.NE]`.