Variational Autoencoder for Movie Recommendations

Hamad Alajeel
Department of ECE, UCSD
Email: halajeel@ucsd.edu

***Abstract*—The task of recommending digital products and services to users in the past decade has become a booming aspect of all industries which rely on the web for sales. For example, companies which provide streaming services for users who consume digital media aim to provide users with personalized experiences that caters towards their preferences. There exist many kinds of recommender systems, but the most popular approach is collaborative filtering, where similarities between user ratings reveal what kinds of items should be recommended to them. Many approaches have been used to implement this including: latent factor models, matrix factorization and neural networks. In this paper we will focus on improving a Variational Autoencoder, the mult-VAE, for the task of movie recommendations by performing an ablation study and experimenting with different strategies for controlling an important regularization hyperparameter, β.**

I. Introduction

Research on Collaborative filtering is based on the use of user/item interaction matrices to extract patterns and similarities between users or between items [6]. Traditional ways of collaborative filtering attempted to use matrix factorization techniques to extract informative latent features from these matrices. These models are called latent factor models [6]. Some of these matrix factorizations include SVD, and regularized SVD which are useful for finding latent factors, but are limited in their ability when the number of items is too high or if the user/item matrix is too sparse which causes overfitting [6]. Moreover, with the advent of deep learning in the past decade, researchers have taken an interest in using models that use non-linearities for their learning instead of relying solely on vector dot products [5]. These models have shown promising results, as not only are they able to reconstruct input data, but also able to generate novel examples with them. Moreover, the number of parameters grows linearly with the number of users and items in recommendation systems which use matrix factorization models which is a serious consideration to have when users and items are in higher orders of magnitude [4].

Therefore, in this paper we have focused on using a deep learning model for our recommendation system, namely we will be using the Mult-VAE model because it has shown promising results in generating movie recommendations based on Bayesian variational inference [1]. Moreover, this model assumes that the data points are sampled from a multinomial distribution, which has been shown to be a more accurate way to represent implicit feedback data, than logistic and Gaussian distributions [1]. The training data is also assumed to have originated from a latent variable z which in the case of movie recommendation data could indicate something like different users' preferences for genres of movies like horror, comedy, etc.

$$P(X_1 = x_1, X_2 = x_2, \dots, X_d = x_d) = \frac{n!}{x_1!\, x_2! \cdots x_d!}\, p_1^{x_1}\, p_2^{x_2} \cdots p_d^{x_d} \tag{1}$$

For a predefined set of movies, each user is given a vector of words where each feature in this vector is a movie, and if the user has positively interacted with that movie, then a 1 is assigned to that feature. Therefore, each user is given a binary vector with 1's indicating that the user has positively interacted with a specific movie. The VAE is composed of two sections: a decoder and an encoder. The encoder, encodes an input data sample into the hidden space, so it approximates the posterior of the distribution: $q_\phi(z_u \mid x_u)$. The decoder, decodes this encoded feature representation back into the original space, and thus reconstructs the original data. It, therefore, represents the conditional likelihood function which is what the VAE model optimizes during training: $p_\theta(x \mid z)$. During this process, the VAE introduces novel differences which represent movie recommendations for users who haven't viewed those movies by learning hidden patterns between users.

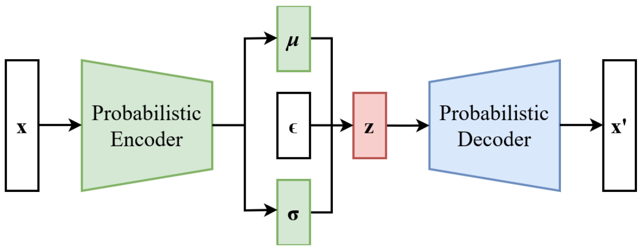

Fig. 1. Variational Autoencoder Architecture

The objective of a variational autoencoder is to maximize the log-likelihood of the data $\log p_\theta(x) = \log \int p_\theta(x \mid z)\, p_\theta(z)\, dz$. [2]. However, in most scenarios computing this integral is intractable, which also implies that the posterior distribution $p_\theta(z \mid x)$ is also intractable [2]. Therefore, algorithms such as EM cannot be used. In our case, the marginal log-likelihood is composed of the sum of the log likelihoods of each feature in our vector of words for each user which can be lower bounded in this manner:

$$\log p(x_u;\theta) \;\geq\; \mathbb{E}_{q_\phi(z_u|x_u)}\Big[\log p_\theta(x_u \mid z_u)\Big] \\ -\; \mathrm{KL}\Big(q_\phi(z_u \mid x_u) \,\Big\|\, p(z_u)\Big) \;\equiv\; \mathcal{L}(x_u;\theta,\phi). \tag{2}$$

The lower bound term includes the function $q_\phi(z_u \mid x_u)$ that approximates the true posterior distribution. This is one of two functions the VAE optimizes. In the end, maximizing this lower bound maximizes the log-likelihood of the data which would allow the VAE model to accurately capture the distribution of the data. To adapt variational autoeoncoders for different use cases, researchers have proposed introducing a parameter, $0 < \beta \leq 1$, in front of the KL term to control its influence in the training process [1], [4], [7]:

$$\mathcal{L}_\beta(x_u;\theta,\phi) \;\equiv\; \mathbb{E}_{q_\phi(z_u|x_u)}\big[\log p_\theta(x_u \mid z_u)\big] \\ -\beta \cdot \mathrm{KL}\big(q_\phi(z_u \mid x_u) \,\|\, p(z_u)\big). \tag{3}$$

Controlling beta has been an important topic of research for improving the performance of VAE's and adapting them for different tasks. This loss function can be viewed in a different manner to our original conceptualization. The expectation of the conditional log-likelihood can be viewed as a reconstruction term, while the KL term can be viewed as a regularization term which determines how far the approximate posterior distribution can deviate from the prior of the hidden variable z [1]. making β closer to one, ensures that the posterior does not deviate far from the prior, preventing it from fitting to the data and also enabling the model to generate new reconstructions of the input data which are not exact replicas, but which contain new information which follows patterns in the hidden variable z. Thus, β can be viewed as a hyperparameter used to control regularization of the posterior. In essense, it controls the trade-off between the generativity of the model, and its ability to fit to the data [1]. In the case of movie recommendations, keeping beta low for at least some duration of the training process ensures that the model attempts to recreate the input data it is processing with slightly novel differences following patterns between users in terms of their preferences for similar types of movies.

Traditionally, a simple heuristic was used to control β during training. This heuristic is to slowly anneal beta from 0 to 1 during training [7]. This was found to generate good results when training VAE's as it allows the VAE to learn to encode information before allowing the KL term to fully come into effect, forcing the model to stabilize [7]. For the case of movie recommendations, annealing β to a value of 0.2 was found to be better alternative [1]. Therefore, controlling β has not been a subject carefully delved into, and we will be focusing on applying two novel proposals for controlling β to investigate if this would improve our model's performance. Firstly, we will test controlling β using cyclical annealing [9]. This method was developed for NLP tasks because of a phenomenon called the "KL vanishing term" [9]. In NLP tasks, it is often the case that the KL term goes to 0 which prevents the model from learning more useful latent feature representations, thus dislodging the posterior from the prior by setting the KL term to 0 periodically allows it to learn more useful latent features using what it has learned from the previous cycle [9]. However, in NLP tasks VAE architectures are different because there are two pathways for the model. One with the current sample data being processed and another which includes a sequence of previous samples for learning contextual information [9]. This is different than traditional VAE's which only process one data sample at a time. Therefore, testing this novel method is tentative.

Another method for annealing which we have more confidence in comes from researchers focusig on improving VAE's for the same task we are working on, movie recommendations [4]. They have proposed setting β to be a function of the user input data in this manner:

$$\beta(x_u) = \gamma|x_u| = \gamma \sum_{i=1}^{N} x_{ui} \tag{4}$$

The authors who proposed this novel method did not compare the performance of their VAE model which had a different architecture when using this method as opposed to the original heuristic method of annealing. Therefore, we experiment with this annealing technique to see if it introduces any improvement to the mult-VAE model we our working on.

In summary we will attempt to improve the mult-vae model for the task of movie recommendations in two ways. Firstly, we will perform an ablation study on the model to examine the effects of tweaking different parameters of the model on its performance as well as different aspects of the training process. We will examine: changing the number of hidden layers, changing the activation functions in hidden layers, changing the dimension of the latent space, and changing the optimizer of the model. Secondly we will experiment with the two aforementioned techniques for controlling β to investigate if they have any positive effect on the performance or convergence of the model.

II. Methods

A. Ablation Test Design

We experiment with changing each aspect of the model or training process in isolation to examine their effects on performance:

- **Activation Function**: Compare ReLU vs. Tanh under identical architectures to assess non-linearity impacts on latent space formation and reconstruction fidelity.
- **Layer Depth**: Examine the effects on performance of using two hidden layers instead of one: from [input dimensions,600,200] to [input dimensions,600,400,200] for both the decoder and encoder layers
- **Hidden Layer Size**: Change the hidden layer size from 200 to 300
- **KL Regularization**: Evaluate two methods to control the β hyperparameter against the traditional heuristic of annealing to 1:
  - cyclical annealling under this schedule:
  - using beta that is a function of the number of interactions of each user
- **Optimization**: Contrast Gradient Descent with the Adam optimizer to determine:
  - Momentum-based vs. adaptive learning rate impacts
  - Training dynamics under different gradient scaling regimes

We evaluate the performance of the model under each one of the aforementioned changes while keeping all the model's default hyperparameters constant along with the training pipeline.

B. Dataset and Preprocessing

We use the MovieLens-20M dataset comprising 138,493 unique users (each contributing at least 20 ratings) and 26,744 movies (each receiving at least 5 ratings), with a total of 20,000,263 explicit user-movie interactions on a 1-5 star rating scale. Following the preprocessing protocol [1], we first binarize the ratings by thresholding scores at 4.0 (i.e., ratings ≥ 4 are treated as positive feedback), then filter users with fewer than 5 positive interactions to ensure meaningful interaction signals. The processed dataset is partitioned into training (80% of each user's earliest interactions), validation (10%), and test sets (10% most recent interactions), preserving temporal consistency in recommendation evaluation. This rigorous pre-processing yields a final corpus of 15.3 million valid interactions across 112,318 users and 24,612 movies, ensuring data quality while maintaining coverage of long-tail items.

C. Evaluation Metrics

We employ two principal metrics to assess recommendation performance:

1. Normalized Discounted Cumulative Gain (NDCG@100): Quantifies ranking quality through position-sensitive discounting, where relevant items appearing higher in the recommendation list receive greater weight. For a user u, let $\text{rel}_i \in \{0, 1\}$ indicate the relevance label of the item at position i, and IDCG_k represent the maximum achievable DCG when all relevant items are perfectly ranked. The metric is defined as:

$$\text{NDCG@}k(u) = \frac{1}{\text{IDCG}_k} \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)} \tag{5}$$

where:

- The numerator $2^{\text{rel}_i} - 1$ creates exponential gain for relevant items
- The denominator $\log_2(i+1)$ imposes position-based discounting
- $\text{IDCG}_k = \sum_{i=1}^{\min(k,|I_u|)} \frac{1}{\log_2(i+1)}$ serves as normalization constant

2. Recall@20/Recall@50): Measures the proportion of relevant items captured in the top-k recommendations. Given I_u as the set of ground-truth relevant items for user u, and $\text{Top}_k(u)$ as the recommended items:

$$\text{Recall@}k(u) = \frac{|I_u \cap \text{Top}_k(u)|}{\min(k, |I_u|)} \tag{6}$$

Key characteristics:

- **Recall@20**: Emphasizes precision in short-list recommendations
- **Recall@50**: Evaluates coverage in longer recommendation lists
- Denominator uses $\min(k, |I_u|)$ to cap maximum achievable recall at 1

Both metrics are computed per-user and averaged across all test users. NDCG@100 evaluates ranked list quality through graded position sensitivity, while Recall@k measures binary relevance coverage at different list lengths.

D. Architectural Analysis

1) Layer Architecture: The architectural modification from a 1-layer to 2-layer hierarchy in both encoder and decoder networks introduces distinct theoretical tradeoffs in representation learning dynamics:

Original Architecture ($[d_{\text{in}}, 600, d_{\text{latent}} = 200]$): The shallow bottleneck structure is anticipated to enforce aggressive dimensionality reduction, potentially filtering high-frequency noise at the cost of oversmoothing complex preference patterns. The single 600→200 compression layer may act as a strong informational bottleneck, theoretically promoting disentangled latent representations through forced feature merging. However, the limited nonlinear transformations could restrict the model's capacity to capture hierarchical user preference structures, particularly for users with diverse interaction histories.

Modified Architecture ($[d_{\text{in}}, 600, d_{\text{latent}} = 300]$ and $[d_{\text{in}}, 600, 400, d_{\text{latent}} = 200]$): The expanded 600→300, 600→400→200 hierarchy introduces graduated compression, where the larger unit intermediate layer is hypothesized to act as a "feature buffer" - preserving nuanced interaction patterns before final latent projection. The additional nonlinear transformation step allows progressive abstraction of user preferences, potentially better modeling both broad genre affinities and specific niche interests.

This architectural evolution tests whether the expanded depth (1) improves preference modeling through hierarchical feature learning versus (2) introduces optimization challenges from increased non-convexity. The dimensionality reduction cascade (600→300, 600→400→200) maintains computational tractability while theoretically enabling more granular control over latent space organization compared to the original single-step compression.

2) β Controlling Methods: Firstly we will test the effects of cyclically annealing beta under this scedule:

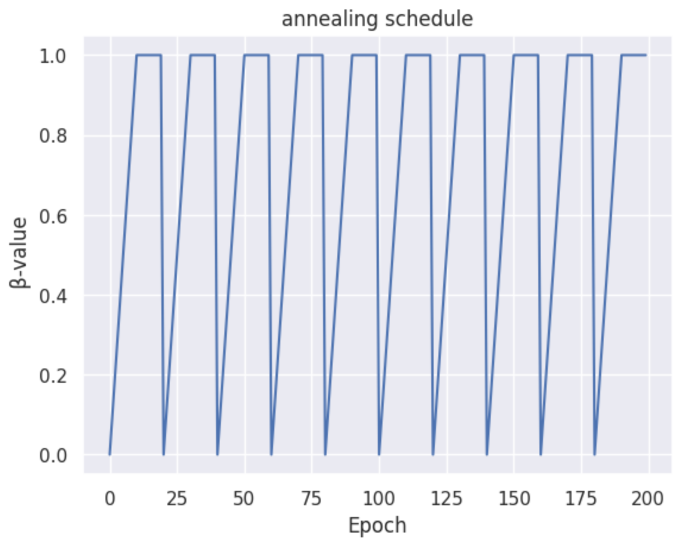

Fig. 2. Cyclical Annealing Schedule for β

Secondly, we will test making beta a function of the number of user interactions with $\gamma = 0.005$ which has been cross validated, as tweaking γ can drastically change our model's performance.

3) Activation Function Analysis: In our variational autoencoder architecture, where the same activation function is applied symmetrically to both encoder and decoder networks, we establish theoretical expectations for ReLU and Tanh through their inherent mathematical properties:

ReLU (Rectified Linear Unit):

$$f(x) = \max(0, x) \tag{7}$$

The piecewise linear nature is anticipated to create sparse latent representations through hard gating of negative activations, potentially enhancing the encoder's ability to discard irrelevant features during dimensionality reduction. For the decoder, this sparsity may promote selective feature reconstruction by suppressing noise propagation. However, the non-differentiability at zero and complete deactivation of negative values could theoretically lead to fragile gradient flow between the encoder-decoder pair, particularly in our deepened architecture with three hidden layers.

Tanh (Hyperbolic Tangent):

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{8}$$

The smooth saturation boundaries (output $\in$ [-1,1]) are expected to enforce implicit regularization on both encoder outputs and decoder reconstructions, potentially stabilizing latent space organization. However, the bounded output range may artificially constrain the decoder's reconstruction capacity, as movie preference patterns often exhibit unbounded multinomial distributions. The zero-centered nature could either facilitate faster encoder convergence by aligning with Gaussian latent priors or induce destructive interference in decoder feature combinations.

This controlled comparison tests whether ReLU's sparsity-driven feature selection outweighs its gradient fragility risks versus Tanh's stable gradient propagation versus its output magnitude constraints, under identical architectural conditions across both network halves.

4) Optimization Strategy: The choice of optimization algorithm critically impacts the variational autoencoder's ability to navigate the complex loss landscape shaped by the reconstruction-KL divergence tradeoff. We analyze two fundamentally distinct approaches:

Adam Optimizer:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{9}$$

The adaptive moment estimation mechanism is theoretically advantageous for handling sparse gradients in our deep encoder-decoder architecture. The per-parameter learning rate adaptation (via $\hat{m}_t$ and $\hat{v}_t$) is expected to automatically balance gradient magnitudes between the reconstruction-dominated decoder path and KL-regularized encoder path. However, the exponential moving averages may introduce momentum mismatch between first and second moments, potentially destabilizing latent space organization during early training phases. The inherent bias-correction steps could either mitigate initialization sensitivity or amplify noise in gradient estimates for our 2-layer network.

Gradient Descent:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L} \tag{10}$$

The static learning rate imposes uniform update scaling across all parameters, theoretically enforcing coordinated learning between encoder and decoder networks. This uniformity may help maintain equilibrium between latent space regularization (KL term) and reconstruction fidelity. However, the fixed step size risks either slow convergence in flat loss regions or overshooting in high-curvature areas, particularly problematic near latent space bifurcation points where the reconstruction-KL balance shifts abruptly.

This comparison tests whether Adam's parameter-wise adaptation better handles the VAE's inherent loss landscape complexity versus Gradient Descent's rigid but stable update consistency. The critical tension lies in balancing the encoder's need for precise latent space regularization against the decoder's requirement for flexible reconstruction capability, mediated through identical learning dynamics across both networks.

III. Experiment

In this section, we describe the experiments conducted to evaluate the performance of the model under different hyperparameter settings. Our main focus was on analyzing the impact of varying the hidden layer, activation function, optimizer, and controlling the β term in the loss function. We run each test in 200 epochs.

A. *Original model*

We replicated the experiment of the original model and obtained the results shown in the figure below. This is used for comparison with the other result after changing different parameters.

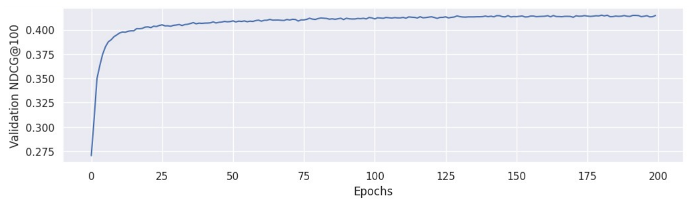

Fig. 3. Validation ranking metrics for original model.

TABLE I
TEST RESULTS FOR THE ORIGINAL MODEL.

Metric	Value
Test NDCG@100	0.41827
Test Recall@20	0.38831
Test Recall@50	0.52439

B. *Hidden Layer*

We experimented with the following configurations:

- **Hidden Layer Size:** We changed size of hidden layer, from [input dim, 600, 200] to [input dim, 600, 300].

Fig. 4. Validation ranking metrics for changing hidden layer size.

TABLE II
TEST RESULTS FOR CHANGING HIDDEN LAYER SIZE.

Metric	Value
Test NDCG@100	0.41608
Test Recall@20	0.38736
Test Recall@50	0.52168

We observed that changing the hidden layer size led to a decline in model performance. As shown in the test results, increasing the hidden layer size can introduce excessive parameters, making training more challenging and resulting in lower NDCG@100 and Recall scores. Therefore, selecting an appropriate hidden layer size is crucial to balance model expressiveness and generalization for optimal performance.

- **Number of Hidden Layers:** We varied the number of hidden layers, from [input dim, 600, 200] to [input dim, 600, 400, 200].

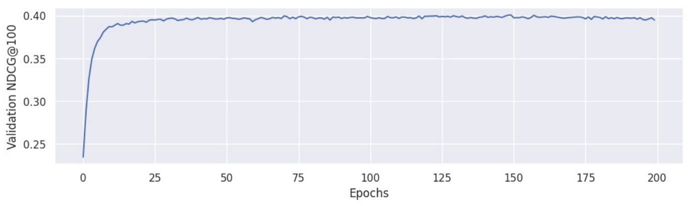

Fig. 5. Validation ranking metrics for changing number of hidden layers.

TABLE III
TEST RESULTS FOR CHANGING NUMBER OF HIDDEN LAYERS.

Metric	Value
Test NDCG@100	0.40345
Test Recall@20	0.37775
Test Recall@50	0.51692

We observed that increasing the number of hidden layers in the MLP led to a decline in model performance. Although deeper networks generally improve representational capacity, excessive layers can introduce challenges such as overfitting, vanishing gradients, and increased training instability. As seen in the test results, the NDCG@100 and Recall scores decreased, suggesting that the additional layers did not contribute to better feature extraction but rather hindered effective learning. This indicates that an overly deep MLP may not always be beneficial and that an optimal depth should be chosen to balance complexity and generalization.

C. *Cyclical Annealing for β in Loss Function*

We experimented with the following configurations:

- **Cyclical Annealing with 10 cycles and 0.5 ratio:** We set β with 10 cycles and 0.5 ratio for 200 epochs.

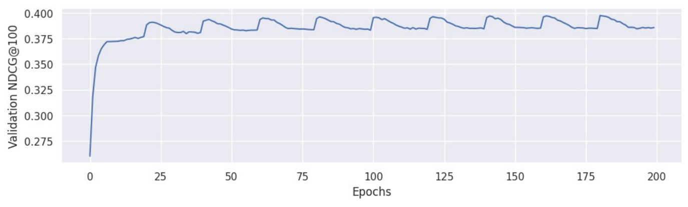

Fig. 6. Validation ranking metrics for changing β with cyclical annealing.

TABLE IV
TEST RESULTS FOR CHANGING β WITH CYCLICAL ANNEALING.

Metric	Value
Test NDCG@100	0.40222
Test Recall@20	0.37263
Test Recall@50	0.51050

In our experiment, we observed that the use of cyclical annealing led to a decline in model performance. Although cyclical annealing aims to mitigate the KL vanishing issue by dynamically adjusting the β term during training, it can also introduce instability in optimization. Frequent fluctuations in β can cause the model to struggle to maintain a stable trade-off between latent space regularization and reconstruction quality.

Moreover, since our model has only one path for reconstruction, the potential benefits of cyclical annealing may not be fully utilized. In models with multiple paths, cyclical annealing can help different components of the model balance between learning a meaningful latent space and improving the quality of the reconstruction. However, with a single path, abrupt changes in the β term can disrupt stable training, leading to suboptimal latent representations. This instability could explain the observed decline in NDCG@100 and Recall scores, suggesting that cyclical annealing may not be well-suited for models with a single reconstruction path.

- **Cyclical Annealing with tuning all 1s in the cycle to 0.2:** We set β with 10 cycles, 0.5 ratio and tuning all 1s in the cycle to 0.2 for 200 epochs.

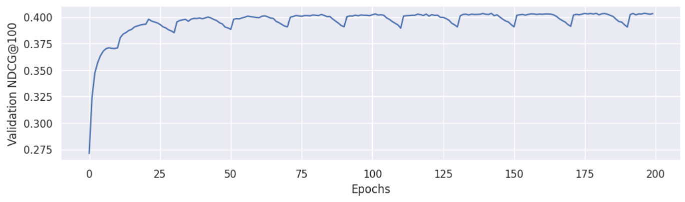

Fig. 7. Validation ranking metrics for changing β with cyclical annealing and tuning all 1s in the cycle to 0.2.

TABLE V
TEST RESULTS FOR CHANGING β WITH CYCLICAL ANNEALING AND TUNING ALL 1S IN THE CYCLE TO 0.2.

Metric	Value
Test NDCG@100	0.40694
Test Recall@20	0.37841
Test Recall@50	0.51647

We experimented with setting β to 10 cycles with a 0.5 ratio and further fine-tuned the cycle by upper bounding β to 0.2 over 200 epochs. Compared to the original setup, where β was set to 10 cycles with a 0.5 ratio without further modifications, this adjustment led to an improvement in performance. The reduction of 1s to 0.2 helped mitigate excessive regularization, allowing the model to better balance latent space learning and reconstruction quality. This suggests that overly high β values in the cycle may have constrained the latent representation too much, limiting the model's ability to capture useful information. By introducing a lower peak value of 0.2, the model was able to retain more flexibility in learning informative latent features, ultimately leading to better NDCG@100 and Recall scores relative to upper bounding β by 1. This still yielded slightly worse performance than the original model which slowly annealed β to 0.2.

D. β as a Function of User Interactions

The results obtained from this experiment are as follows:

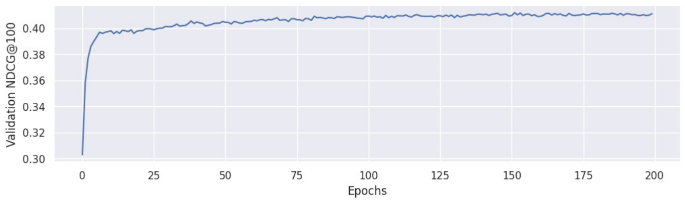

Fig. 8. Validation ranking metrics for β as a function of user interactions.

TABLE VI
TEST RESULTS USING $\gamma = 0.005$.

Metric	Value
Test NDCG@100	0.41745
Test Recall@20	0.38790
Test Recall@50	0.52463

Out of all of our experiments, scaling β in proportion to the amount of user interactions has shown the most promising results. Firstly, it attains an NDCG@100 value which is almost equivalent to the original model. Secondly, it outperforms the original model in terms of the recall@50 metric. This means that it was able to retrieve movies from the original data sample more accurately while almost maintaining the same ranking for each movie. Additionally, by inspecting the plot, convergence is way quicker than the original heuristic method. Thus, this has shown to be a promising new addition to our VAE model which makes it more competitive than the original just using the traditional heuristic.

E. Activation Functions

We investigated the role of different activation functions:

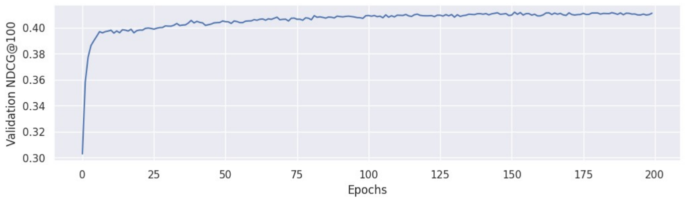

Fig. 9. Validation ranking metrics for changing activation function to ReLU.

TABLE VII
TEST RESULTS CHANGING ACTIVATION FUNCTION TO RELU.

Metric	Value
Test NDCG@100	0.41485
Test Recall@20	0.38642
Test Recall@50	0.52442

We experimented with changing the activation function from tanh to ReLU, and the results showed a trade-off in performance. Specifically, NDCG@100 decreased, while Recall@50 improved. This suggests that ReLU, being a non-saturating activation function, may have helped the model learn more diverse representations, leading to better recall at a higher cutoff. However, the drop in NDCG@100 indicates that the ranking quality of top recommendations might have been negatively affected, possibly due to ReLU's tendency to introduce sparsity in activations. This trade-off suggests that while ReLU improves overall item coverage, tanh might still be preferable when optimizing for ranking precision in top results.

F. Optimizer

We investigated the role of different Optimizers:

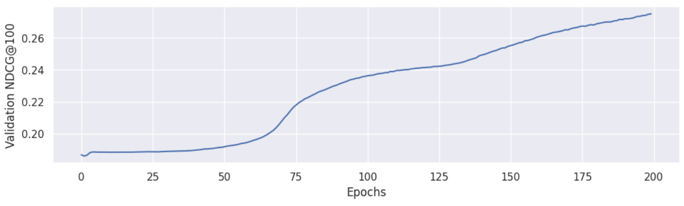

Fig. 10. Validation ranking metrics for changing Optimizer to Gradient Descent.

TABLE VIII
TEST RESULTS FOR CHANGING OPTIMIZER TO GRADIENT DESCENT.

Metric	**Value**
Test NDCG@100	0.27933
Test Recall@20	0.24704
Test Recall@50	0.35517

We observed that changing the optimizer from Adam to Gradient Descent led to a decline in performance. This drop is likely due to the fact that rank data contains many zeros, making optimization more challenging for Gradient Descent. Unlike Adam, which adaptively adjusts learning rates for different parameters and handles sparse gradients more effectively, Gradient Descent uses a fixed learning rate, which may struggle to update parameters efficiently when dealing with many zero-valued gradients. As a result, the model may have converged more slowly or gotten stuck in poor local minima, leading to lower NDCG and Recall scores. This suggests that optimizers with adaptive learning rates, such as Adam, are better suited for handling sparse ranking data.

G. The Final Result

Here is the table that compare all the results.

TABLE IX
COMPARISON OF TEST RESULTS ACROSS DIFFERENT EXPERIMENTAL SETTINGS.

Experiment	**Test NDCG@100**	**Test Recall@20**	**Test Recall@50**
Original Model	0.41827	0.38831	0.52439
Changed Hidden Layer Size	0.40694	0.37841	0.51647
Increased Number of Hidden Layers	0.40345	0.37775	0.51692
Cyclical Annealing (10 cycles, 0.5 ratio, capped to 1)	0.41608	0.38736	0.52168
Cyclical Annealing (10 cycles, 0.5 ratio, capped to 0.2)	0.41485	0.38642	0.52442
β as a function of user interactions	0.41745	0.38790	0.52463
Activation Function: ReLU	0.40222	0.37263	0.51050
Optimizer: Gradient Descent	0.27933	0.24704	0.35517

IV. CONCLUSION

In this study, we explored various modifications to our model and evaluated their impact on performance using the **NDCG@100**, **Recall@20**, and **Recall@50** metrics. The key findings from our experiments are as follows:

- **Hidden Layer Size:** Changing the hidden layer size led to performance degradation, due to the model becoming harder to optimize. This highlights the importance of tuning hidden layer dimensions to balance complexity and representation capacity.
- **Number of Hidden Layers:** Increasing the number of hidden layers decreased performance, as both **NDCG@100** and **Recall** scores dropped. This indicates that an overly deep network introduce unnecessary complexity, leading to overfitting or unstable convergence.
- **Cyclical Annealing:** Tuning the cyclical annealing schedule by reducing all 1s in the cycle to 0.2 resulted in better performance compared to the standard 10-cycle, 0.5 ratio setting. This suggests that excessive regularization in the annealing process may have constrained the model's latent space learning, while a reduced β peak allowed for better reconstruction.
- β **as a function of user interactions:** This has shown to be the most promising direction for improving the VAE for movie recommendations. Out of all the studies completed on this model, formulating β as function of the number of user interactions had the best NDCG@100 value out of all the other methods in comparison to the original model. Moreover, it achieved better recall@50 than the original model, meaning it was better able to reconstruct the input data sample. This user specific method of controlling β ensures that users who have not had many movie interactions, thus who are outliers or anomalies, do not have a disproportionate effect on the training of this model. Thus, we conclude that using this method of controlling β is more beneficial for the task of movie recommendations.
- **Activation Function Change (Tanh $\rightarrow$ ReLU):** Switching from Tanh to ReLU resulted in lower NDCG@100 but higher Recall@50. This suggests that ReLU, while useful for generalizing across a broader range of items, may negatively impact ranking precision at the top positions due to its sparse activation behavior.
- **Optimizer Change (Adam $\rightarrow$ Gradient Descent):** Replacing Adam with Gradient Descent significantly lowered performance in NDCG@100 and Recall scores. This is because ranking data contains many zeros, and Gradient Descent, which does not adaptively adjust learning rates like Adam, struggles with sparse gradients.

Our findings indicate that maintaining only one hidden layer, and using adaptive optimizers like Adam are crucial for achieving good performance. Additionally, while ReLU may improve recall, its impact on ranking precision should be considered when optimizing for NDCG. Our most promising finding was using beta which was a function of the amount of user interactions. It achieved the best results for recall@50 out of all of the other experiments, and it achieved almost comparable NDCG@100 to the original model. This means that using this method for controlling β is a positive improvement to the mult-VAE and possibly other variational autoencoders which have similar tasks to this one. Future work could explore altering the architecture of this VAE more systematically and experimenting with different techniques for training the decoder and encoder MLP's.

References

[1] Liang, Dawen, et al. "Variational Autoencoders for Collaborative Filtering." Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18, 2018, arxiv.org/pdf/1802.05814.pdf, https://doi.org/10.1145/3178876.3186150.

[2] Kingma, Diederik P, and Max Welling. "Auto-Encoding Variational Bayes." ArXiv (Cornell University), 20 Dec. 2013, https://doi.org/10.48550/arxiv.1312.6114.

[3] Rezende, Danilo Jimenez, et al. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." ArXiv:1401.4082 [Cs, Stat], 30 May 2014, arxiv.org/abs/1401.4082.

[4] Shenbin, Ilya, et al. "RecVAE: A new variational Autoencoder for top-N recommendations with implicit feedback." Proceedings of the 13th International Conference on Web Search and Data Mining, 20 Jan. 2020, pp. 528–536, https://doi.org/10.1145/3336191.3371831.

[5] He, Xiangnan, et al. "Neural Collaborative Filtering." ArXiv:1708.05031 [Cs], 25 Aug. 2017, arxiv.org/abs/1708.05031. Accessed 28 May 2021.

[6] P. Parhi, A. Pal and M. Aggarwal, "A survey of methods of collaborative filtering techniques," 2017 International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2017, pp. 1-7, doi: 10.1109/ICISC.2017.8068603.

[7] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning. 10–21.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2017. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[9] Fu, Hao, et al. "Cyclical annealing schedule: A simple approach to mitigating." Proceedings of the 2019 Conference of the North, 2019, https://doi.org/10.18653/v1/n19-1021.