Variants of AutoEncoder in RecSys

Viet-Bang Pham and Minh-Duc Le

*Corresponding author(s). E-mail(s):
Bang.PV222143M@sis.hust.edu.vn; Duc.LM222142M@sis.hust.edu.vn;

Abstract

Recommendation systems are facing challenges like data sparsity and cold-start issues. Recent solutions, particularly those employing autoencoders, show promise. Autoencoders efficiently learn data representations and are valuable for capturing patterns in user-item interactions, leading to personalized recommendations. This report explores autoencoder variants tailored for recommendations, covering designs, functions, and training strategies, with insights into their strengths and limitations.

1 Introduction

Recommendation systems have become integral for enhancing user experiences across various platforms like e-commerce, streaming services, and social media. Many recommendation models have been proposed during the last few years. However, they all have to face two main issues: data sparsity and cold-start issues.

To solve these problems, recent approaches have exploited side information about users or items. Among the various approaches to recommendation systems, autoencoder-based methods have become powerful tools for capturing complex patterns and latent features within user-item interaction data.

Autoencoders are a class of artificial neural networks. They are known for their ability to learn efficient representations of input data in two main steps. (1) Encoding data into a lower-dimensional latent space and (2) reconstructing the input from this representation. In recommendation systems, autoencoders are useful for learning meaningful embeddings of users and items, enabling accurate and personalized recommendations.

This report explores various variants of autoencoders tailored specifically for recommendation systems. First, we delve into different architectural designs, loss functions, and training strategies. Second, we provide a comprehensive overview of the state-of-the-art in utilizing autoencoders for recommendation tasks. Through a comparative implementation of these variants, we show their strengths, limitations, and potential applications. This report is referenced from the RecSys Series article of James Le on Medium.

2 Models

2.1 A Primer on Auto-encoder

First and foremost, we discuss the vanilla auto-encoder, a primer on auto-encoder. This is the basic form of auto-encoder which every version has to follow. As depicted in the diagram below, a standard autoencoder comprises an input layer, a hidden layer, and an output layer. The input data is fed into the input layer, which collaborates with the hidden layer to form an encoder. Subsequently, the hidden layer, in conjunction with the output layer, forms a decoder. Finally, the reconstructed output data emerges from the output layer.

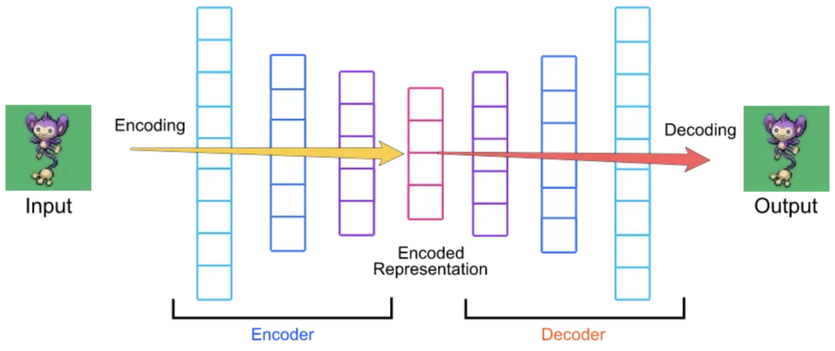


Fig. 1: Autoencoder Architecture

The encoder encodes the high-dimensional input data x into a lower-dimensional hidden representation h with a function f:

$$h = f(n) = s_f(Wx + b) \tag{1}$$

Where s_f is an activation function, W is the weight matrix, and b is the bias vector. The decoder decodes the hidden representation h back to a reconstruction x' by another function g:

$$x' = g(h) = s_g(W'h + b') \tag{2}$$

s_g is an activation function, W' is the weight matrix, and b' is the bias vector. s_f and s_g are non-linear, for example, Sigmoid, TanH, or ReLU. This is because they enable auto-encoders to learn more useful features than other unsupervised linear approaches, say Principal Component Analysis. We can train the model to minimize the reconstruction error between x and x' via either the squared error SE (for regression tasks) or the cross-entropy error CE (for classification tasks).

$$SE(x, x') = ||x - x'|| \tag{3}$$

$$CE(x, x') = -\sum_{i=1}^{n}(x_i log x'_i + (1 - x_i) log(1 - x'_i)) \tag{4}$$

Finally, a regularization term can be added to the final reconstruction error of the auto-encoder:

$$RE_{AE}(x, x') = (\sum_{x} SE|CE(x, x')) + \lambda * Regularization \tag{5}$$

There are many variants of auto-encoders currently used in recommendation systems. In this report, we want to discuss the five most common models: AutoRec, DeepRec, Collaborative Denoising Auto-encoder (CDAE), Multinomial Variational Auto-encoder (MultiVAE), and RecVAE.

2.2 AutoRec

AutoRec was introduced from "Autoencoders Meet Collaborative Filtering" by Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie [1]. It is one of the earliest models that consider the collaborative filtering problem from an auto-encoder perspective.

In the paper, there are m users, n items, and R (a partially filled user-item interaction/rating matrix with dimension mxn). A partially filled vector r_u represents a user u and a partially filled vector r_i represents an item i. AutoRec directly takes these vectors as input data and obtains the reconstructed rating at the output layer. Depending on two types of inputs, there are two variants of AutoRec: item-based AutoRec (I-AutoRec) and user-based AutoRec (U-AutoRec).Given the input r_i, the reconstruction is:

$$h(r^{(i)}, \theta) = f(Wg(Vr^{(i)} + \mu) + b) \tag{6}$$

f and g are the activation functions, and the parameter Theta includes W, V, μ, and b. AutoRec uses only the vanilla auto-encoder structure. The objective function of the model is similar to the loss function of the auto-encoder:

$$argmin_{\theta} \sum_{i=1}^{N} ||r^{(i)} - h(r^{(i)}; \theta)||_o^2 + \lambda reg \tag{7}$$

This function can be optimized by resilient propagation (converges faster and produces comparable results) or L-BFGS (Limited-memory Broyden Fletcher Goldfarb Shanno algorithm).

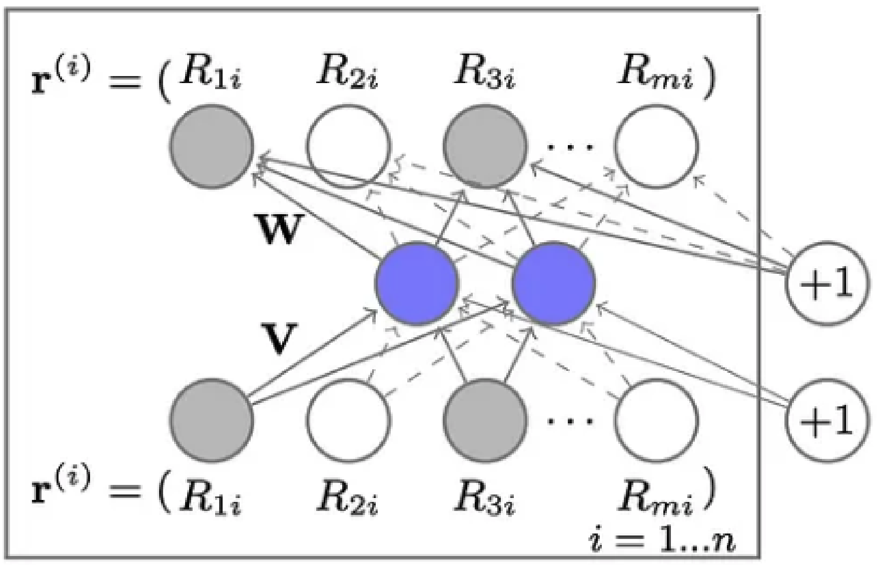

$\mathbf{r}^{(i)} = (R_{1i}\ R_{2i}\ R_{3i}\ \cdots\ R_{mi})$
W
V
+1
+1
$\mathbf{r}^{(i)} = (R_{1i}\ R_{2i}\ R_{3i}\ \cdots\ R_{mi})$
$i = 1...n$

Fig. 2: Item-base AutoRec model

2.3 DeepRec

DeepRec is created by Oleisii Kuchaiev and Boris Ginsburg from NVIDIA, as seen in "Training Deep Autoencoders for Collaborative Filtering" [2]. The model has five important distinctions from AutoRec: (1)The network is much deeper. (2)The model uses "scaled exponential linear units" (SELUs). (3)The dropout rate is high. (4)The authors use iterative output re-feeding during training.

The figure above depicts a typical 4-layer autoencoder network. There are 2 layers e_1 and e_2 in the encoder, while the decoder has 2 layers d_1 and d_2. The representation z connects these layers. $f(W * x + b)$, where f is some non-linear activation function, represents the layers. The last layer of the decoder should be kept linear if the range of the activation function is smaller than that of the data. It is very important for activation function f in hidden layers to contain a non-zero negative part, and use SELU units in most of the experiments. Since it does not make sense to predict zeros in users' representation vector x, the authors optimize the Masked Mean Squared Error loss:

$$MMSE = \frac{m_i * (r_i - y_i)^2}{\sum_{i=0}^{n} m_i} \tag{8}$$

r_i is the actual rating, y_i is the reconstructed rating, and m_i is a mask function such that $m_i = 1$ if r_i is not 0 else $m_i = 0$.

During the forward pass and inference pass, the model processes a user, represented by a vector of ratings from the training set, denoted as x. It's important to note that x is highly sparse, while the output of the decoder $f(x)$ is dense, encompassing rating predictions for all items within the corpus. Consequently, to explicitly enforce

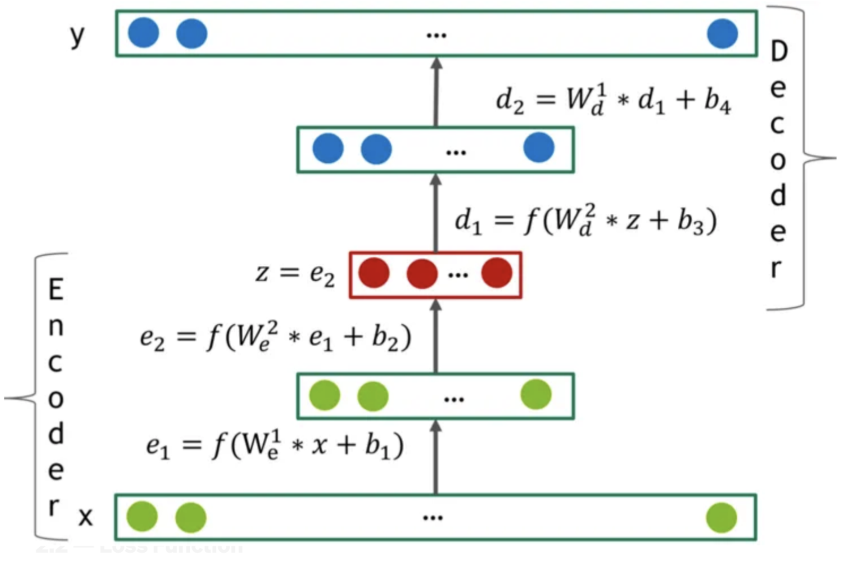

y
$d_2 = W_d^1 * d_1 + b_4$
$d_1 = f(W_d^2 * z + b_3)$
$z = e_2$
$e_2 = f(W_e^2 * e_1 + b_2)$
$e_1 = f(W_e^1 * x + b_1)$
x
Encoder
Decoder

Fig. 3: DeepRec Architecture

a fixed-point constraint and facilitate dense training updates, the authors introduce an iterative dense re-feeding step into each optimization iteration, outlined as follows:

In the initial forward pass, when provided with sparse input x, the model calculates the dense output $f(x)$ and evaluates the Mean Squared Error (MMSE) loss using equation 8. Subsequently, during the initial backward pass, the model computes gradients and adjusts weights accordingly.

Moving to the second forward pass, the model treats f(x) as a new data point, leading to the computation of $f(f(x))$. Both $f(x)$ and $f(f(x))$ become dense at this stage. Notably, the MMSE loss now incorporates all m as non-zero values. Finally, during the second backward pass, the model once again computes gradients and updates weights based on these calculations.

2.4 Collaborative Denoising Auto-encoder

"Collaborative Denoising Autoencoders for Top-N Recommender Systems" authored by Yao Wu, Christopher DuBois, Alice Zheng, and Martin Ester [3], introduces a neural network architecture characterized by a single hidden layer. Diverging from approaches like AutoRec and DeepRec, CDAE offers distinctive features:

Firstly, the input of CDAE diverges from traditional user-item ratings; instead, it relies on partially observed implicit feedback r, representing a user's item preferences. Specifically, if a user expresses a preference for a given item, the corresponding entry in r is marked as 1; otherwise, it remains 0.

Secondly, while previous models such as AutoRec and DeepRec primarily focus on rating prediction tasks, CDAE shifts its emphasis towards ranking prediction, often

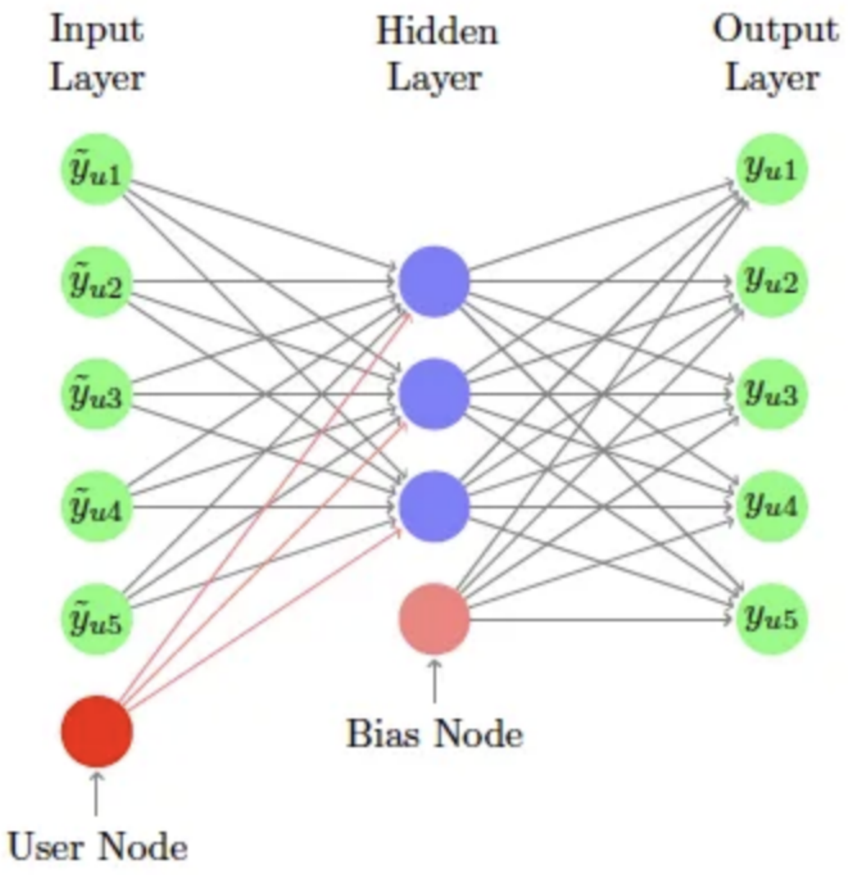


Fig. 4: CDAE Architecture

referred to as Top-N preference recommendations. This strategic pivot underscores CDAE's utility in scenarios where the goal is to identify and prioritize a subset of items that are most likely to appeal to a given user, rather than solely predicting individual ratings.

These distinctions underscore the unique role and capabilities of Collaborative Denoising Autoencoders within the landscape of recommender systems, emphasizing their suitability for addressing specific recommendation objectives and user interaction patterns.

The provided figure illustrates a typical structure of CDAE, comprising three layers: the input layer, the hidden layer, and the output layer. In the input layer, there are a total of $I + 1$ nodes. The initial I nodes represent user preferences, with each node corresponding to a specific item. The final node, depicted in red in the figure, is user-specific, indicating that different users possess distinct nodes and associated weights. Here, Yu represents the I-dimensional feedback vector of user u across all i items. This vector is sparse and binary, with non-zero values denoted as $y_u i = 1$ if item i has been rated by user u, and $y_u i = 0$ otherwise.

Moving to the hidden layer, there are K nodes, with K¡¡I, along with an additional node, depicted in pink, to capture bias effects. The blue nodes are fully connected to the nodes in the input layer.

Transitioning to the output layer, there are I nodes, representing the reconstructed output of the input y_u. These nodes are fully connected to the nodes in the hidden layer.

In the reconstruction process of CDAE, the corrupted input r_{corr} is sampled from a conditional Gaussian distribution $P(r_{corr}|r)$. The reconstruction of r_{corr} is formulated accordingly:

$$h(r_{corr}) = f(W_2 g(W_1 r_{corr} + V_u + b_1) + b_2) \tag{9}$$

In the formula, W_1 represents the weight matrix associated with the encoder, governing the transformation from the input layer to the hidden layer. Similarly, W_2 denotes the weight matrix corresponding to the decoder, regulating the transition from the hidden layer to the output layer. Additionally, V_u signifies the weight matrix for the user-specific red node, capturing individual user characteristics. Alongside, both b1 and b2 are bias vectors contributing to the model's overall bias effects. The parameters of CDAE are learned by minimizing the average reconstruction error as follows:

$$\underset{W_1, W_2, V_u, b_1, b_2}{min} \frac{1}{M} \sum_{u=1} ME_{P(r_{corr}|r)}[l(r_{corr}, h_{corr})] + \lambda Regularization \tag{10}$$

The loss function $l(r_{corr}, h(r_{corr}))$ in the provided equation can be formulated using either square loss or logistic loss. However, CDAE employs the square L2 norm to govern the model's complexity. Additionally, two key strategies are implemented for effective parameter learning: (1) Stochastic Gradient Descent (SGD): This optimization technique is utilized to iteratively update the model's parameters, enabling efficient learning by computing gradients from a subset of the training data. (2) AdaGrad: CDAE employs AdaGrad to dynamically adjust the training step size throughout the learning process. This adaptive optimization method automatically scales the learning rate based on the historical gradient information, facilitating faster convergence and improved performance.

Furthermore, the authors introduce a negative sampling technique to alleviate the computational burden while maintaining recommendation quality. This technique involves extracting a small subset of items that the user has not interacted with, significantly reducing time complexity without sacrificing ranking accuracy. During inference, CDAE leverages a user's existing preference set (uncorrupted) as input and proceeds to recommend items based on the largest prediction values obtained from the output layer. This process ensures personalized and efficient recommendations tailored to each user's preferences.

2.5 Multinomial Variational Auto-encoder

One of the seminal contributions to this discourse is the paper titled "Variational Autoencoders for Collaborative Filtering" authored by Dawen Liang, Rahul Krishnan, Matthew Hoffman, and Tony Jebara from Netflix [4]. This work introduces a variant of Variational Autoencoders (VAE) tailored specifically for recommendation systems dealing with implicit data. Notably, the authors propose a principled Bayesian inference framework to estimate model parameters, showcasing superior performance compared to conventional likelihood functions commonly employed in this domain.

In the paper, the authors employ U to index all users and I to index all items within the system. They represent the user-by-item interaction matrix as X, with dimensions UI. Each lowercase x_u denotes a bag-of-words vector, indicating the number of interactions (e.g., clicks) for each item from user u. Given the implicit nature of the feedback, this matrix is binarized, resulting in entries with only 0s and 1s, simplifying the representation for computational efficiency and interoperability.

$$z_x \ N(0, I_K); \pi(z_u) \ softmax f_\theta(z_u); x_u \ Mult(N_t, \pi(z_u)) \tag{11}$$

The generative process of the model, as depicted in the equation, unfolds as follows: (1) For each user u, the model samples a K-dimensional latent representation z_u from a standard Gaussian prior distribution. (2) Subsequently, z_u is transformed through a non-linear function f_θ to generate a probability distribution over I items, denoted as $\pi(z_u)$. (3) The function f_θ is implemented as a multi-layer perceptron with parameters θ and a softmax activation function, allowing for complex mappings from latent space to item probabilities. (4) Given the total number of interactions (e.g., clicks) from user u, the bag-of-words vector x_u is sampled from a multinomial distribution with probabilities determined by $\pi(z_u)$.

The log-likelihood for user u, conditioned on the latent representation u, is then computed as follows:

$$logP_\theta(x_u|z_u) = \sum_i x_{ui} log \pi_i(z_u) \tag{12}$$

The authors contend that the multinomial distribution is well-suited for addressing the collaborative filtering problem outlined in the equation. This choice of distribution reflects a deliberate design aimed at appropriately incentivizing the model to allocate probability mass to non-zero entries in x_u, representing user-item interactions.

However, given that $\pi_i(z_u)$ must sum to 1, the items within the distribution are compelled to compete for a limited budget of probability mass. Consequently, the model is inclined to assign more probability mass to items deemed more likely to be interacted with by users. This strategic allocation of probability mass is crucial for enhancing the model's performance, particularly in the context of top-N ranking evaluation metrics commonly used in recommendation systems.

By prioritizing the likelihood of interactions for items with higher probabilities, the model aims to excel in recommending the most relevant items to users, thereby elevating the overall effectiveness of the recommendation process. This emphasis on probability mass allocation underscores the nuanced balance between incentivizing diverse item exploration and emphasizing recommendations aligned with user preferences, ultimately contributing to the model's efficacy in generating top-N recommendations.

2.6 RecVAE

"RecVAE: a New Variational Autoencoder for Top-N Recommendations with Implicit Feedback" by Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh [5] is a development from VAE and Mult VAE methods based on implicit feedback. (1) New Encoder architecture. (2) New composite prior distribution. Using normal distribution and latent code distribution from the previous model. (3) A new way to set the weighted beta of Kullback leibler terms. Use a user-specific beta. (4) A new way to learn. Alternating between updating the encoder and updating the decoder.

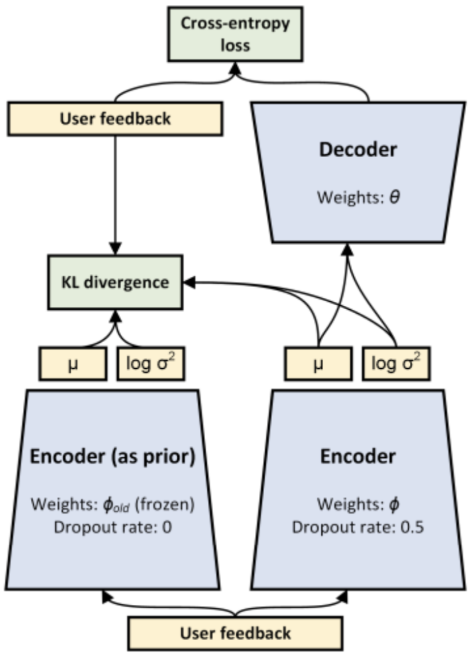


Fig. 5: CDAE Architecture

In the encoder architecture, input data moves from ELBO to a denoising variational autoencoder.

$$L_{Mult-VAE} = AND_{q_\phi(z_{in}|x_{in})}[logp_i(x_{in}|z_{in}) - \beta KL(q_\phi(z_{in}|x_{in})||p_i(z_{in})] \tag{13}$$

$$L_{Mult-VAE} = AND_{q_\phi(z_{in}|\tilde{x}_{in})}AND_{p(\tilde{x}_{in}|x_{in})}[logp_i(x_{in}|z_{in}) - \beta KL(q_\phi(z_{in}|x_{in})||p_i(z_{in})] \tag{14}$$

Similar to MultVAE, the model uses the noise distribution $p(\tilde{x}|x)$ defined as element-wise multiplication of the vector x by a vector of Bernoulli random variables parameterized by their mean u noise. The authors keep the structure of both likelihood and approximate posterior unchanged.

$$p_\theta(x_u|z_u) = Mult(x|n_u, \pi(z_u)) \tag{15}$$
$$\pi(z_u) = softmax(f_\theta(z_u)) \tag{16}$$
$$f_\theta(z_u) = Wz_u + b \tag{17}$$
$$q_\phi(z_u|x_u) = N(z_u|\varphi_\phi(x_u)) \tag{18}$$

The architecture for inference networks takes the idea of densely connected layers from CNNs, swish activation function, and layer normalization.

$$swish(x) = xsigmoid(\beta x) = \frac{x}{1 + e^{-\beta x}} \tag{19}$$

The decoder network is simply a linear layer with a softmax activation function

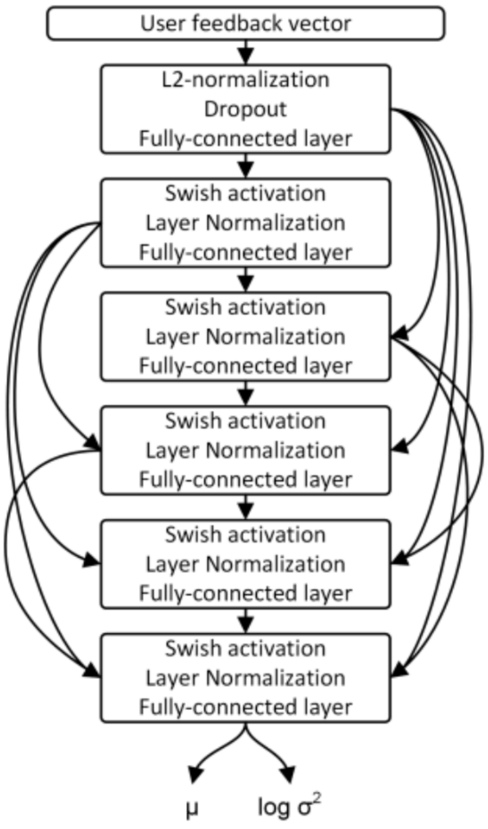
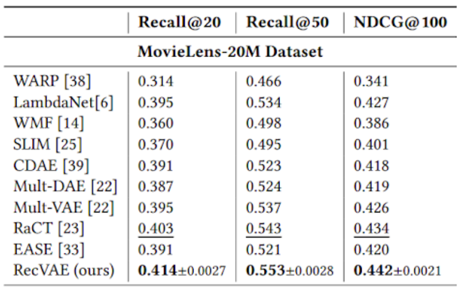


Fig. 6: RecVAE Architecture

3 Implementation

In this section, we proceed to implement all models for the recommendation problem using the Movielens dataset. The goal is to predict the ratings that a user will give to a movie, with ratings ranging from 1 to 5. Here, we will utilize k-fold cross-validation and metrics including RMSE, Precision, and Recall to evaluate the models. Information about the dataset can be found here. The implementation can be found at the following link: AutoRec, DeepRec, CDAE, MultVAE, RecVAE, and MultDAE.

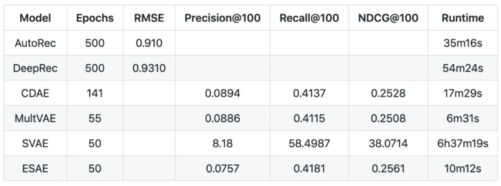
Paper [5] public result:



	Recall@20	Recall@50	NDCG@100
	MovieLens-20M Dataset		
WARP [38]	0.314	0.466	0.341
LambdaNet[6]	0.395	0.534	0.427
WMF [14]	0.360	0.498	0.386
SLIM [25]	0.370	0.495	0.401
CDAE [39]	0.391	0.523	0.418
Mult-DAE [22]	0.387	0.524	0.419
Mult-VAE [22]	0.395	0.537	0.426
RaCT [23]	0.403	0.543	0.434
EASE [33]	0.391	0.521	0.420
RecVAE (ours)	**0.414**±0.0027	**0.553**±0.0028	**0.442**±0.0021

Fig. 7: Paper public result

James Le report result:



Model	Epochs	RMSE	Precision@100	Recall@100	NDCG@100	Runtime
AutoRec	500	0.910				35m16s
DeepRec	500	0.9310				54m24s
CDAE	141		0.0894	0.4137	0.2528	17m29s
MultVAE	55		0.0886	0.4115	0.2508	6m31s
SVAE	50		8.18	58.4987	38.0714	6h37m19s
ESAE	50		0.0757	0.4181	0.2561	10m12s

Fig. 8: Medium report result

After implementation, we obtained the following results:

Table 1: Implementation Result

Model	Epochs	RMSE	Recall@50	Recall@100	NDCG@100
MultDAE	200		0.2574	0.3610	0.2038
MultVAE	400			0.3920	0.4300
RecVAE	50		0.5496	0.6746	0.4435
AutoRec	500	0.9139			
DeepRec	500	0.8615			

Upon implementation, the observed outcomes closely align with the findings presented in the academic literature, notably with the RecVAE model demonstrating superior performance among the trio of models: MultiDAE, MultVAE, and RecVAE. Noteworthy is the exceptional performance of the RecVAE model, surpassing the reported results in the published paper.

In the context of James Le's report, empirical results diverge from the documented findings. Specifically, DeepRec exhibits enhanced performance relative to AutoRec.

4 Conclusion

In this comprehensive report, we have meticulously examined state-of-the-art Autoencoder architectures and their application within collaborative filtering. Our analysis encompasses a thorough review of five seminal papers employing Autoencoders in the recommendation framework, namely: (1) AutoRec, (2) DeepRec, (3) Collaborative Denoising Autoencoder, (4) Multinomial Variational Autoencoder, and (5) Sequential Variational Autoencoder. These models have been rigorously tested and evaluated on movie ratings and ranking recommendation tasks utilizing the Movielens dataset, yielding encouraging outcomes. Looking forward, our future endeavors will involve further exploration of additional models, conducting comprehensive evaluations, and potentially integrating them into our forthcoming academic research, such as our master's thesis, or professional projects within enterprise settings.

References

[1] Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B.: Graph neural networks in recommender systems: a survey. ACM Computing Surveys **55**(5), 1–37 (2022)

[2] Kuchaiev, O., Ginsburg, B.: Training deep autoencoders for collaborative filtering. arXiv preprint arXiv:1708.01715 (2017)

[3] Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp. 153–162 (2016)

[4] Liang, D., Krishnan, R.G., Hoffman, M.D., Jebara, T.: Variational autoencoders for collaborative filtering. In: Proceedings of the 2018 World Wide Web Conference, pp. 689–698 (2018)

[5] Shenbin, I., Alekseev, A., Tutubalina, E., Malykh, V., Nikolenko, S.I.: Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 528–536 (2020)