# Capturing Preferences of New Users in Generative Tasks with Minimal Interactions Collaborative Filtering Using Siamese Networks and Soft Clustering

**Subharag Sarkar, Manfred Huber**
The University of Texas at Arlington, Arlington, TX, USA
subharag.sarkar@mavs.uta.edu, huber@cse.uta.edu

## Abstract

Prediction of user preferences is a challenge, in particular when the objective is to learn them without requiring the user to provide a profile or a significant number of interactions. Many collaborative filtering algorithms exist but all of them require the availability of huge datasets of user information and expensive computations. In this paper, a novel architecture is introduced which aims to predict a new user's interests in the context of previous users' interactions with minimal feedback interactions. Here, a Siamese Network is used to generate an embedding space for data from existing users. This information is then used in a Gaussian Mixture Model to generate multiple soft clusters. Based on the embedding space, system responses to the user are generated using a Conditional Generative Adversarial Network which uses a vector drawn from the Gaussian Mixture in embedding space from the Siamese Network as the conditional input. The predictive model then interacts with the new user and based on their feedback adjusts the Gaussian Mixture to find the distribution with the highest probability of generating the user's preferred data. The approach is applied in the context of an image generation task where the goal is to learn to generate images that match the preferences of the user using only a minimal number of direct user interactions. Testing in this domain has shown promising results that exemplify the ability of the approach to capture the user's preferences while presenting only a minimal number of image examples.

## Introduction

With the popularity of the Internet and the advent of smartphones, there has been a great increase in Internet traffic. There are approximately 5.16 billion Internet users, out of which approximately 4.76 billion are social media users (Petrosyan 2023). Since big social media companies are handling such a huge population, there are pressures to understand, capture and cater to user preferences with the underlying objective of captivating the user and generating revenues. Over the years several Recommender Systems with state-of-the-art collaborative filtering algorithms have been proposed which use huge datasets and expensive computations to learn user preferences. However, they become difficult to use if there is very little information about the user.

The novel architecture in this paper is trying to address this problem by aiming to predict a user's preference with minimal interactions. For this, it uses the concept of collaborative filtering in an embedding space where, based on the information from the previous user, the architecture predicts the preference of the new user. A robust Siamese Network (Bromley et al. 1993) is used which takes user data from previous episodes and generates an embedding space for the dataset. After the embedding space is created, a Gaussian Mixture Model generates soft clusters in this space that help predict the preferences of the new user. A Conditional GAN (Mirza and Osindero 2014) is then used, which utilizes the embedding space vector from the soft clusters as the conditional input to generate new data. This data is provided to the new user for feedback which the architecture uses to adjust the clusters and thus the predicted preferences. To test the approach, an image generation domain has been used where the goal is to learn to generate images that match the user's preference based on minimal interactions.

In the remainder of the paper, we first discuss related works before introducing the underlying learning theories of the different machine-learning tools. Based on these concepts the framework is introduced and the details of the architecture are discussed. This is then followed by tests and results. Finally, conclusions and future work are presented.

## Related Work

Capturing user preferences is a difficult problem as acquiring information directly from the user can be prone to error and indirect methods have been computationally expensive and time-consuming. Still, capturing user preferences accurately helps understand and respond to the needs of the user and thus enterprises have collected large amounts of information on users and by using AI techniques a deeper understanding of user interaction has been learned. These Recommender Systems have automated the generation of recommendations based on the data analysis, but usually require huge amounts of information and computation to achieve good accuracy.

In one of the first commercial recommendation systems, called Tapestry (Goldberg et al. 1992), the term "collaborative filtering" was introduced. The system recommended documents from newsgroups to users. The aim was to use social collaboration to understand the user requirements and to save them from large volumes of preference questionnaires or document interactions. Collaborative filtering an-

alyzes all user data to find good user-item pair matches. In contrast, the earlier methodology of content filtering was based on information retrieval. After the early success of collaborative filtering in the Group lens system (Resnick et al. 1994), the problem was mapped to classification, where dimensionality reduction was used to improve the solution.

When Netflix, an online video streaming service released a large-scale dataset containing 100 million ratings from half a million users and announced an open competition for the best collaborative filtering algorithm, Matrix Factorization (Koren, Bell, and Volinsky 2009) based on linear algebra and statistical analysis emerged as a state of the art technique.

Later, a new recommendation framework (Wang et al. 2019) was proposed which exploits user-item graphs to propagate an embedding. This has led to modeling expressive high-order connections in the graph. Collaborative filtering was also used in IoT scenarios (Cui et al. 2020), where time correlation coefficient and K-means clustering has been used to group similar users for an accurate recommendation.

The works presented above are all based on the notion that big datasets are available that help train the Recommender System. In contrast, our approach is based on the notion of accurately predicting user preference with minimal interaction. Information from previous users is taken into account to develop an embedding space, but the amount of data used is comparably lower. This lessens the dependency on big datasets to train and make accurate predictions.

## Learning Model Background

This section introduces the learning models that will be used to build the proposed user preference learning approach.

### Siamese Networks

A Siamese neural network is an Artificial Neural Network architecture that uses two identical subnetworks to establish an embedding space that represents item similarity. To do this, the same weights are used to compute output vectors for two different inputs which can be compared to each other. This architecture was introduced in (Bromley et al. 1993) where it was used to compare signatures. After its introduction, the algorithm has been extensively used in face detection algorithms. (Koch, Zemel, Salakhutdinov, et al. 2015) incorporated the concept of Siamese networks and one-shot learning to create a system that learns from a single example of each class. (Sheng and Huber 2019) used the concept of Siamese networks on time series data which are trained in a weakly supervised fashion using only information about the similarity between data items to cluster behaviors based on sensor data. Utilizing that the network learns to output embedding vectors for the input data that reflect similarity in terms of Euclidean distance, this weakly supervised technique has been shown to yield comparable performance to fully supervised methods with reduced labeling overhead.

### Conditional Generative Adversarial Networks

Generative Adversarial Networks (GAN) (Goodfellow et al. 2014) have been designed as a framework that uses two competing neural networks in a zero-sum game. Here the generator network is indirectly trained to learn to generate output that aligns with the data distribution of the training set. The Discriminator on the other hand tries to predict whether the data provided is from the training dataset or from the generator. The constant struggle of the generator creating data and trying to pass them as original data and the discriminator finding out the real and false data helps the generator to create more realistic responses. However, the basic GAN framework uses simple noise to generate meaningful data and is not able to differentiate between different classes of data. For example, when given an animal dataset to train, the GAN can learn to create images of different animals. But it is not possible for the framework to generate images of one type of animal. Since we need to generate data based on user type, we need to use a GAN framework that can generate data based on some label that characterizes the type.

Conditional GAN (Mirza and Osindero 2014) is a machine learning framework, which uses the core principle of GAN but adds a condition vector in its input that helps generate class-based data. The conditional input represents the labels for the data which, when fed to the generator with the noise vector, trains it to generate data for that label. The general architecture of a Conditional GAN is shown in Figure 1

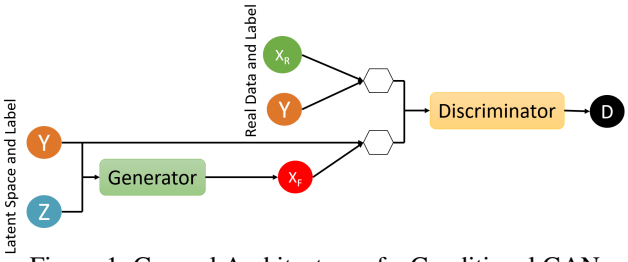



Figure 1: General Architecture of a Conditional GAN

### Gaussian Mixture Models

Clustering is an unsupervised learning problem aimed at grouping unlabeled data. Several hard clustering algorithms are used most commonly. Their main disadvantage is that they associate each point with only one cluster which will not yield good accuracy with datasets containing points that cater to multiple clusters as is the case with overlapping user preferences. By contrast, as a soft clustering technique Gaussian Mixture Models assume that the dataset contains multiple Gaussian distributions and that each data point can belong to each distribution with varied degrees of probability. Each distribution contains the following parameters:

- A mean $\mu$ that defines the center of the cluster
- A covariance $\Sigma$ that defines its width. In a multivariate scenario, it is equivalent to the dimensions of an ellipsoid.
- A mixing probability $\pi$ which defines how big or small each Gaussian function is.

The initial Gibbs sampling training (Rasmussen 1999) was later replaced with Expectation Maximization (Xuan, Zhang, and Chai 2001) as it provides more accurate clusters.

## Learning framework

**Dataset:** To develop and initially evaluate the framework, the Fashion-Mnist dataset (Zalandor 2017) has been used for

experimentation. This dataset, examples of which are shown in Figure 2, was chosen due to the following properties:

- Each image is 28x28 grayscale, which makes it easier to create an initial neural network for processing.
- The dataset contains images from 10 different fashion items where each item is visually different while sharing lower-level features, facilitating overlapping preferences.

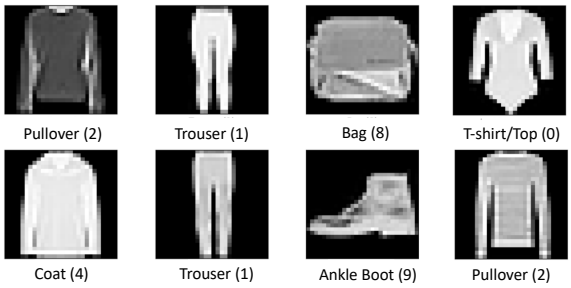



Figure 2: Examples from Fashion-Mnist Dataset

## Siamese Network

The primary objective of the Siamese network in this approach is to create an embedding space for the collaborative filter to work on. Figure 3 shows the internal structure of the Siamese Network. It includes the following three layers:

- **Input Layer -** Here image pairs of two types are sent as input. A Like-Like pair contains two images liked by the same user. They can be of similar or different classes. A Like-Dislike pair contains two images that can again be of the same class but have to be from two different users.
- **Embedding Layer -** In this layer, we use two similar Branch Networks which share the same weights for updating. The networks take each image from the Input Layer as input and then convert them into embedding vectors which are fed as input to the Decision Layer.
- **Decision Layer -** The first Layer which takes both outputs from the Embedding Layer is the Euclidean distance layer which computes the distance between the two embedding vectors. The images from the same user will generate embedding vectors close to each other whereas images from different users will have dissimilar embedding vectors. The concatenated output goes through a fully connected layer to predict either 0 (Same User) or 1 (Different User).

Based on the true and predicted answer for the pair, the network is trained. Binary Cross Entropy and Adam as the optimizer are used for the experiment. Since they predict similarity and dissimilarity, the branch networks are trained with the same weights. Once the Siamese Network is trained, the embedding vectors of the images from previous users generate a distribution in the latent embedding space which helps find clusters using Gaussian Mixture Models and serves as a condition vector to a Conditional Generative Adversarial Network that generates sample images.

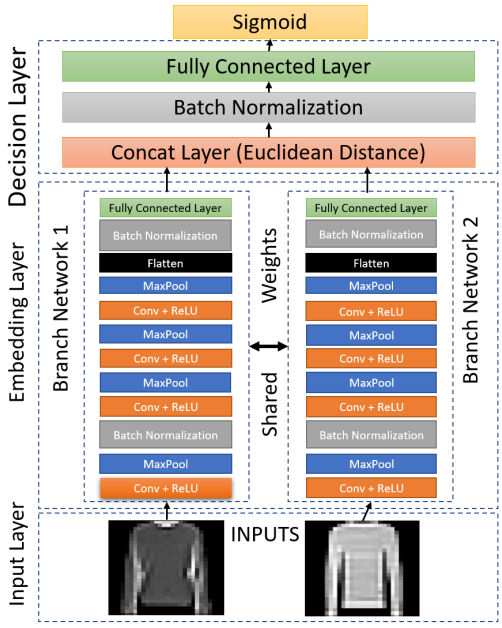



Figure 3: Architecture of Siamese Network

## Conditional Generative Adversarial Network

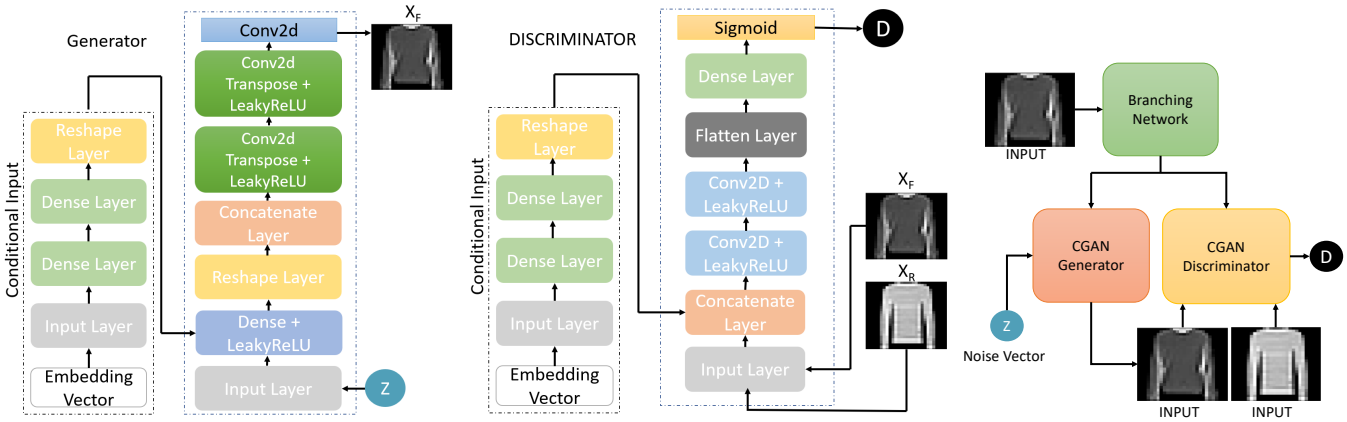
The main motivation for using the Conditional GAN is to provide the architecture the power to create its own data which will aid it to learn faster. Here the CGAN is trained with user data collected during experimentation. For the conditional input of the CGAN, the **Branching Network** of the Siamese network is used to generate embedding vectors for each image. The architecture is designed as follows:

- **Condition Generation -** We use the **Embedding Layer** of the Siamese Network, where all the user data is sent as input and an embedding vector is generated. This gives us the conditional input for the CGAN.
- **Generator Computation -** The conditional input is concatenated with a noise vector and sent to the generator. The generator uses this input to generate an image of the same dimension as in the original Fashion-Mnist dataset.
- **Discriminator Computation and Training -** The conditional input and the image (either from the source dataset or from the generator) are concatenated and sent to the discriminator. The discriminator outputs 1 or 0 depending on whether it predicts that the image is from the source or from the generator, and based on this the generator is trained to learn to create images that the discriminator can not distinguish from the originals.

Figure 4(a) and (b) show the Generator and Discriminator network structures, respectively, and Figure 4(c) shows their integration into the CGAN architecture.

## Gaussian Mixture Model for New User Preferences

The Branching Network of the Siamese Network takes all the images from previous users and generates embedding vectors which create the embedding space. These unlabeled vectors in the embedding space will be used to bootstrap




(a) Generator (b) Discriminator (c) CGAN Architecture

Figure 4: Architecture of Conditional GAN. Generator (left), Discriminator (middle), and Complete CGAN (right)

and structure the user preference for new users. When the model works on predicting the preference of a new user, the new user might prefer images from different clusters (previous user preferences) or sub-cluster combinations. The goal is to capture these by building a Gaussian Mixture Model for the new user's likes and dislikes, capturing them in the form of soft clusters in the embedding space. This implicitly assumes that the structure of the embedding space captures basic commonalities related to human tastes. Note that we are not assuming that resulting user preferences fall into a type that is already present among previous users (as is done generally in collaborative filtering) but only that the embedding space captures basic underlying data properties in light of human preferences. The training algorithm works in a semi-supervised fashion, treating user feedback as preference labels while assigning no labels to other users' data. The presence of liked and disliked examples leads to a small modification where each Gaussian also receives a cluster label, assigning it to either the liked or disliked class for the user. The complete algorithm operates in the following way:

- **Initialization -** For initialization K-means Clustering is a good choice. The value of the cluster number $k$ is defined by the elbow method. These clusters provide a starting point for the Gaussian Mixture model to train itself. The mixing probability $\pi$ is initialized according to the data distribution, and as initially no data for the new user is available, membership in the liked or disliked distribution for each Gaussian is initialized randomly.
- **Expectation-Maximization -** The EM algorithm finds the maximum likelihood of a vector among all the clusters. It is divided into two steps:

  **E-Step -** In the E-Step the posterior probabilities (Expectation value) of each data point are computed with the given $\pi$, $\mu$, and $\Sigma$. For data points that received feedback from the new user, posterior probabilities for all Gaussians that have the opposite liked or disliked label as the feedback indicates are set to 0.

  **M-Step-** In the M-Step, $\pi$, $\mu$, and $\Sigma$ are updated by maximizing the log-likelihood with respect to the parameters. Once data points with user feedback are available, cluster labels for all Gaussians are updated according to the likelihood of the cluster generating liked or disliked data points from the set that the user provided feedback on.

The E-M step continues until the algorithm converges. The architecture is shown in Figure 5

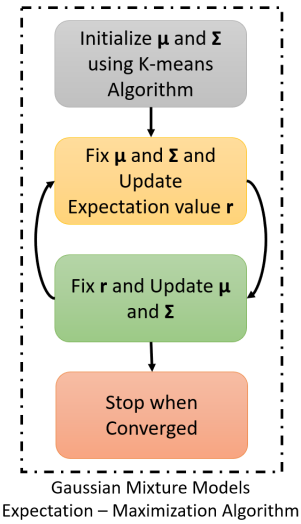



Figure 5: Architecture of Gaussian Mixture Model

## Prediction Model

The Prediction Model interacts directly with new users and based on their feedback predicts the images a user might like. The following steps are used for the model to work.

- Initially there is no user information, so a cluster from the embedding space is chosen randomly and the corresponding $\mu$ and $\Sigma$ are used to pick a vector from that cluster.
- This vector is then used as the conditional input for the CGAN. The generated image is shown to the user and the feedback is recorded as "Liked" or "Disliked". The embedding vector is also added to the dataset.

- The EM algorithm is called to update the Gaussian Mixture Model based on the latest dataset and user feedback. During the EM step, the new images for the new user are updated differently. If the user has liked or disliked the image, its Expectation value for all the predefined clusters which the model thinks has the opposite label is set to 0. This process moves the clusters every time, customizing the prediction for the new user.
- After the EM step, the predicted labels are updated. For each cluster, we find the probability density values of the liked images (li) and disliked images (di), and then use the formula $(li - di)/(li + di)$. If the value is greater than 0, then the cluster is liked, otherwise, it is disliked.
- The model runs for at most 100 iterations. Initially, it explores 90% of the time with a decay of 20% every 20 iterations. This initially promotes exploration and slowly moves towards exploitation. At each iteration, Precision, Recall, and Accuracy are evaluated and training is stopped if a threshold is crossed. For the experiment, the threshold for (Precision, Recall, Accuracy) is (80%,50%,50%). We use a higher Precision value than Recall since we want the model to learn to present liked images early but are not as interested in covering all interests of the user.

The Model is shown in Figure 6

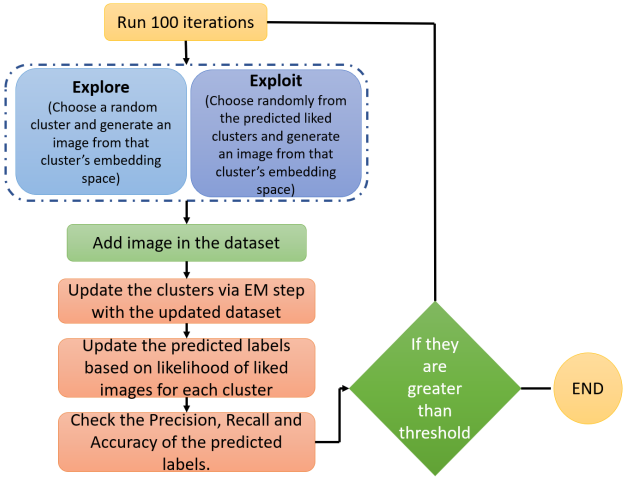



Figure 6: Prediction Model Architecture

## Experiments and Results

In this section, each of the components as well as the complete approach is tested and evaluated.

### Siamese Network

The Siamese Network was trained with user-consistent and user inconsistent pairs. The network ran for 250 epochs and achieved a testing accuracy of 97% and validation accuracy of 95%. The training loss is 0.11 and the validation loss is 0.21. The training curves are shown in Figure 7.

These results show that the Siamese network is able to learn consistent embedding of the distinguishing characteristics of existing user data with high precision.

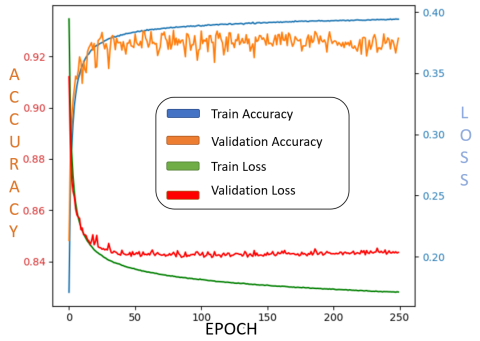



Figure 7: Accuracy of Siamese Network

### Gaussian Mixture Model

Using the embedding space and the existing user data without labels, a Gaussian Mixture Model with 18 clusters (chosen by the elbow method) is trained and then customized using new user feedback. The clusters are shown in Figure 8, where (a) displays the clusters before training with the new user's feedback, and (b) shows the clusters after training, where red clusters are Disliked and green clusters are Liked.

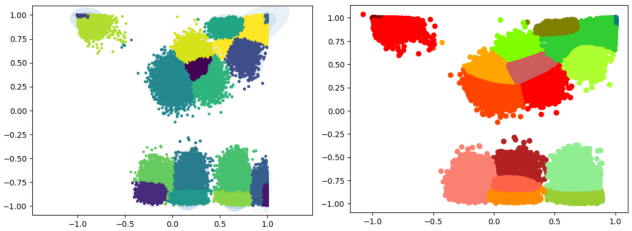



Figure 8: Soft Clusters in Embedding Space Before (a) and After (b) User Interaction

This figure shows that the system is able to form distinct clusters that align with the users, indicating that it can identify relevant characteristics within the embedding space.

### Prediction Model

Once the Gaussian Mixture is pre-trained, the prediction and feedback process is started with a set of 50 new users with randomly initialized preferences defined as subsets of the Fashion-Mnist categories. To compare the efficiency of the proposed architecture, another experiment is set up with the same number of clusters but where the Gaussian Mixtures do not utilize previous users' data but only the new user's data points and feedback. This corresponds to the situation where no collaborative filtering is used but preferences are learned in a supervised way using the pre-trained embedding space. Figure 9 shows the number of iterations the models required in each condition to reach the prediction thresholds for each of the new users and how many Liked and Disliked images each user has seen during the training process. The use of the collaborative filtering approach by pre-trained Gaussian Mixtures (shown in the right figure) significantly reduces the need for user feedback compared to the baseline (shown on

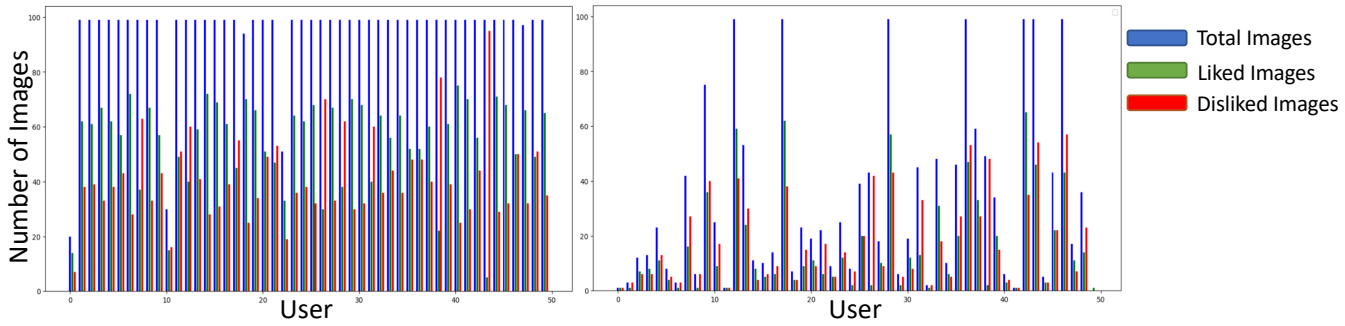



Figure 9: Iterations Required to Train User Model for Each of the 50 Users Without Previous User Gaussian Mixture Model (left) and With Previous User Gaussian Mixture Model for Collaborative Filtering (right).

the left) and thus increases the efficiency of the architecture. Using this, the system is able to learn preference distributions relatively fast, with many users having to rate less than 20 images. However, the results also show that there still is significant variance among users, indicating the potential for improvement in the way data points are generated from the Mixture distribution. Focusing more on liked clusters should here increase precision at the cost of recall with fewer disliked images, while focusing longer on exploration should yield higher recall but more disliked images. The model with no pre-trained Gaussian Mixture (shown in the right figure), required significantly more interactions with only 3 instances where fewer than 40 images were shown. For the rest of the users, the model completed its run of displaying all 100 images and never reached the desired thresholds.

To check how efficient the proposed user preference learning approach is in predicting user preferences, we generated 100 images from each liked and disliked cluster according to the trained user-specific Gaussian Mixture model's predicted preference for the user. These images were then tested against the actual preference of the user and precision and recall values were computed. Precision shows how well the model can generate items matching each user's preference while Recall measures how comprehensively the user's preferences are captured. Figure 10 shows the precision and recall for each of the 50 new users for the proposed model with the pre-trained Gaussian Mixture from previous users' data.

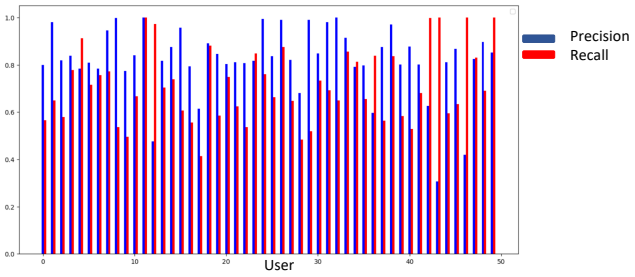



Figure 10: Precision and Recall of Prediction Model

The model achieved an average precision of 81.9% and an average recall of 71.3% across all 50 users. By contrast, the model without the pre-trained mixture only achieved an average precision of 36.6% and an average recall of 37.2%.

Figure 11 similarly shows the model accuracy value for each user using the pre-trained Mixtures.

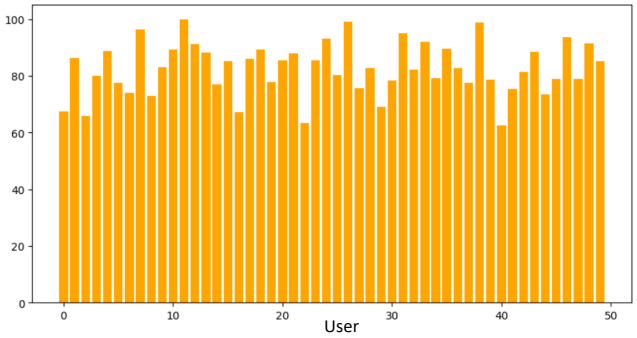



Figure 11: Accuracy of Predictive Model

The average accuracy here is 82.67%, which contrasts with only 60.18% without the collaborative pre-training. These results again demonstrate the ability of the proposed architecture to learn effective user preferences with limited interactions and illustrate the importance of the approach's ability to utilize previous users' data.

## Conclusion and Future Work

This paper presents a novel framework that uses the concept of collaborative filtering in conjunction with a Siamese Network, Gaussian Mixture Models, and CGAN to create a predictive model that generates an accurate preference prediction for new users with limited amounts of user-specific data. For this, it establishes a user preference distribution as a Gaussian Mixture Model on an embedding space learned by a Siamese Network using other users' data. Results in an image generation domain show that the system is capable to embed important general preference characteristics, successfully using these to build a user preference distribution of unknown preferences from limited feedback and generate training data efficiently using a Conditional GAN network. While the results presented here are promising, the used exploration and exploitation policy is very simple and could be improved upon in terms of reducing false-positive exposures using other exploration strategies. We can also use exploration strategies that are more targeted to a specific task to further improve the experience of users with the system.

## References


Bromley, Jane et al. (1993). "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems* 6.

Cui, Zhihua et al. (2020). "Personalized Recommendation System Based on Collaborative Filtering for IoT Scenarios". In: *IEEE Transactions on Services Computing* 13.4, pp. 685–695. DOI: 10.1109/TSC.2020.2964552.

Goldberg, David et al. (1992). "Using collaborative filtering to weave an information tapestry". In: *Communications of the ACM* 35.12, pp. 61–70.

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661.

Koch, Gregory, Richard Zemel, Ruslan Salakhutdinov, et al. (2015). "Siamese neural networks for one-shot image recognition". In: *ICML deep learning workshop*. Vol. 2. Lille.

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix factorization techniques for recommender systems". In: *Computer* 42.8, pp. 30–37.

Mirza, Mehdi and Simon Osindero (2014). *Conditional Generative Adversarial Nets*. DOI: 10.48550/ARXIV.1411.1784. URL: https://arxiv.org/abs/1411.1784.

Petrosyan, Ani (2023). *Worldwide digital population 2023*. URL: https://tinyurl.com/4fht5679.

Rasmussen, Carl (1999). "The infinite Gaussian mixture model". In: *Advances in neural information processing systems* 12.

Resnick, Paul et al. (1994). "Grouplens: An open architecture for collaborative filtering of netnews". In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186.

Sheng, Taoran and Manfred Huber (2019). "Siamese networks for weakly supervised human activity recognition". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, pp. 4069–4075.

Wang, Xiang et al. (2019). "Neural Graph Collaborative Filtering". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: Association for Computing Machinery, pp. 165–174. ISBN: 9781450361729. DOI: 10.1145/3331184.3331267. URL: https://doi.org/10.1145/3331184.3331267.

Xuan, Guorong, Wei Zhang, and Peiqi Chai (2001). "EM algorithms of Gaussian mixture model and hidden Markov model". In: *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*. Vol. 1, 145–148 vol.1. DOI: 10.1109/ICIP.2001.958974.

Zalandor (2017). *Fashion-Mnist Dataset*. URL: https://github.com/zalandoresearch/fashion-mnist.