**Model Architecture**

In this assignment, we implemented a Deep AutoEncoder to predict missing ratings in a user-item matrix and generate recommendations for users. The AutoEncoder is structured as follows:

**1. Encoder**

* The encoder serves to learn a compressed, lower-dimensional representation of the input data (user ratings). By capturing patterns in user preferences, it reduces the dimensionality of the input while preserving essential information for accurate rating predictions.
* **Layers**:
  + The input layer has 9,559 neurons, corresponding to the total number of movies in the dataset.
  + The first hidden layer consists of 512 neurons, with a ReLU activation function applied after the linear transformation. ReLU is used here to introduce non-linearity and allow the network to model complex patterns in the data.
  + The second hidden layer also has 512 neurons and uses the ReLU activation function, helping the model progressively capture user and movie relationships.
  + The final encoder layer has 1,024 neurons. This is the lowest-dimensional representation and serves as a bottleneck. The bottleneck layer helps the model generalize better by forcing it to capture only the most critical features, removing noise and redundant information.

**2. Decoder**

* The decoder reconstructs the original input matrix from the compressed representation. Its purpose is to attempt to reproduce the original user-item matrix, filling in missing values in the process.
* **Layers**:
  + The first layer in the decoder mirrors the encoder’s bottleneck layer, with 1,024 neurons and a ReLU activation function.
  + The next layer has 512 neurons, followed by a ReLU activation, similar to the encoder’s second layer.
  + The following layer also has 512 neurons with a ReLU activation, completing the symmetric structure of the AutoEncoder.
  + The final output layer has 9,559 neurons, corresponding to the original input size. This output layer represents the reconstructed user-movie rating matrix, where the filled-in values represent predicted ratings for movies that users haven’t rated.
* **Activation Function**: ReLU activation is used in all hidden layers to introduce non-linearity, allowing the model to learn complex user preferences and interactions between users and movies.

**3. Loss Function – Masked Mean Square Error (MMSE)**

* Given the sparsity of the user-movie rating matrix, it’s important to focus on actual ratings rather than zeros (which indicate missing values). The Masked Mean Square Error (MMSE) loss function calculates the loss only on non-zero entries, ignoring missing ratings.
* **Calculation**: For each prediction, MMSE compares the output of the AutoEncoder with the actual ratings only at the positions where a user has rated a movie. This selective calculation prevents the loss from being affected by zeros in the matrix, focusing solely on meaningful rating data.
* **Purpose**: MMSE is particularly effective for recommender systems, where datasets are often sparse, and only a fraction of user-item pairs have ratings. By concentrating on known ratings, the model better learns relationships and patterns among users and items, leading to more accurate predictions.

**Results and Analysis**

**1. Training and Validation Loss**

* Over the course of training, we monitored both the training and validation losses. The losses indicate how well the model learns user-item interactions and its generalization ability on unseen data.
* **Observation**: As expected, the training loss gradually decreases, showing that the model is learning to reconstruct user ratings accurately. The validation loss should ideally follow a similar pattern; however, if it begins to increase, it might indicate overfitting, where the model memorizes training data instead of generalizing patterns.

**2. Evaluation and Recommendations**

* Once trained, the AutoEncoder can generate recommendations for users. Given a user’s existing ratings, the AutoEncoder outputs predicted ratings for all items.
* **Recommendation Process**:
  + For each user, the output vector from the AutoEncoder is analyzed.
  + The model’s predictions for items the user hasn’t rated are sorted in descending order, and items with the highest predicted ratings are recommended.
* This approach ensures that recommendations are personalized based on the latent patterns the AutoEncoder has learned about user preferences and item similarities.

**3. Analysis of Model Performance**

* **Effectiveness of MMSE**: The MMSE loss has proven useful in this sparse dataset setting, as it ensures that only known ratings impact the learning process. This allows the model to concentrate on meaningful rating data, leading to better accuracy in the reconstructed ratings.
* **Limitations and Future Improvements**:
  + **Sparse Data**: Although MMSE helps manage sparsity, the AutoEncoder could benefit from more advanced techniques, such as incorporating user and item embeddings, which can further improve the model’s ability to capture complex relationships in sparse data.
  + **Cold Start Problem**: The AutoEncoder may struggle with recommendations for new users or items with few ratings (known as the cold start problem). In the future, hybrid models combining collaborative filtering with content-based methods could address this issue by leveraging additional user or item attributes.

**4. Visual Analysis**

* **Loss Curves**: The training and validation loss curves provide insight into the model’s learning dynamics. A gradually declining validation loss, closely following the training loss, suggests that the model generalizes well.
* **Sample Recommendations**: To demonstrate the model’s effectiveness, sample recommendations were reviewed for a few users. These recommendations align with known user preferences and provide confidence that the model has learned meaningful patterns in the data.