Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2023106616
Vol. 10, No. 3, Juni 2023, hlm. 699-708 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

FILM RECOMMENDER SYSTEM MENGGUNAKAN METODE *NEURAL COLLABORATIVE FILTERING*

Ni'mah Khoiriyah Ayyiyah[1], Retno Kusumaningrum[*2], Rismiyati Rismiyati[3]

[1, 2,3]Universitas Diponegoro, Semarang
Email: [1]nimahkazizah@students.undip.ac.id, [2]retno@live.undip.ac.id, [3]rismiyati@live.undip.ac.id,
*Penulis Korespondensi

(Naskah masuk: 14 Oktober 2022, diterima untuk diterbitkan: 21 Juni 2023)

Abstrak

Pada saat ini media hiburan telah berkembang pesat dan tersedia secara digital. Hiburan khususnya dalam bentuk film semakin tersedia secara luas. Keinginan untuk menikmati hiburan dalam media digital mendorong pengguna internet lain untuk mengunjungi situs-situs yang menawarkan film tertentu, sehingga meningkatkan minat mereka terhadap *website* yang menawarkan hiburan digital. Tidak semua situs penyedia hiburan digital menyajikan item yang menjanjikan kepuasan pengguna. Sebuah item yang sama tidak tentu akan disukai oleh semua *user* dan terbatasnya informasi yang disediakan menjadi salah satu kendala bagi pengguna sehingga membutuhkan waktu untuk pengguna menemukan film yang sesuai. Oleh karena itu *recommender system* dibutuhkan dalam memberikan informasi berdasarkan kebutuhan pengguna. *Recommender system* akan membantu seorang *user* dalam mencari sebuah item yang berdasarkan ketertarikan masing-masing dengan memberikan prediksi beberapa item berdasarkan preferensi *user* yang berasal dari riwayat penilaian *user* terhadap item tersebut. *Recommender system* juga telah mengalami kemajuan dalam mengimplementasikan metode. *Deep learning* yang merupakan salah satu penemuan dalam metode *recommender system* dirancang untuk mengatasi beberapa kekurangan dari teknlogi lain dan memberikan revolusi arsitektur rekomendasi dalam meningkatkan kinerja dalam pemberian prediksi. Penelitian ini menggunakan pendekatan prediksi *Collaborative Filtering* dengan mengimplementasikan *deep learning* berdasarkan teknologi *Neural Collaborative Filtering* pada dataset MovieLens. Evaluasi model dilakukan menggunakan metrik skor regresi *Root Mean Square Error* (RMSE). Hasil pada pengujian model menunjukkan hasil terbaik dengan nilai rata-rata *loss value* sebesar 0,1356 pada fase train dan sebesar 0,8898 pada fase val, dengan *learning rate* dan *batch size* memperoleh kinerja terbaik ketika *learning rate* bernilai 0,001 dan *batch size* dengan nilai 1024.

Kata kunci: film *recommender system, deep learning, neural collaborative filtering.*

FILM RECOMMENDER SYSTEM USING NEURAL COLLABORATIVE FILTERING METHOD

Abstract

At this time entertainment media has become available digitally. Entertainment especially in the form of movies is increasingly widely available. The desire to enjoy entertainment in digital media encourages other internet users to visit sites that offer certain movies, thus increasing interest in websites that offer digital entertainment. Not all digital entertainment provider sites present items that promise user satisfaction. The same item will not necessarily liked by all users and the limited information is one of the obstacles for users so that it takes time for users to find the right film. Therefore, a recommendation system is needed in providing information based on user needs. The recommendation system will help users find items based on their respective interests by providing predictions. The recommender system will help a user find an item based on their respective interests by providing predictions of several items based on user preferences derived from the user's assessment history of the item. The recommendation system has also made progress in implementing the method. Deep learning which is one of the discoveries in the recommender system method is designed to overcome some of the shortcomings of other technologies and provide a recommendation architecture revolution in improving performance in delivery. This study using a Collaborative Filtering prediction approach by implementing deep learning based on Neural Collaborative Filtering technology on the MovieLens dataset. The evaluation of the model was carried out using the Root Mean Square Error regression score metric. The results on the model test show the best results with can average loss value of 0,1356 on the train label and 0,8898 on the val label, with the learning rate and batch size getting the best performance when the learning rate is 0,001 and the batch size is 1024.
Keywords: *film recommender system, deep learning, neural collaborative filtering*

1. PENDAHULUAN

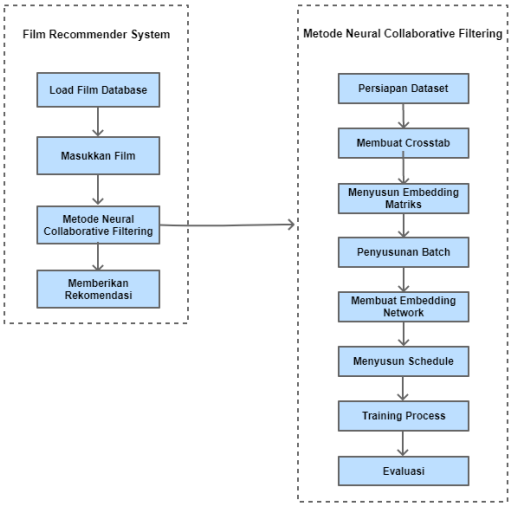
Perkembangan teknologi dan sistem informasi memiliki dampak terhadap kebutuhan informasi di era digital. Teknologi modern telah menyediakan berbagai bentuk kebutuhan informasi, salah satunya adalah media hiburan. Hiburan khususnya dalam bentuk film kini tersedia secara luas, film yang baru diproduksi biasanya menjadi tren di media sosial. Hal ini mendorong pengguna internet untuk mengikuti tren hiburan dengan mengunjungi *website* yang menawarkan film tertentu. Hal tersebut berdampak pada tingginya minat pengguna terhadap *website* yang menawarkan hiburan digital. Namun, tidak semua film yang menjadi tren disukai oleh semua pengguna, butuh waktu lama untuk menemukan film yang sesuai dengan minat pengguna karena banyaknya jumlah film dan terbatasnya informasi yang disediakan. Solusi dari permasalahan tersebut adalah dibutuhkannya suatu sistem rekomendasi untuk memberikan informasi berdasarkan kebutuhan pengguna, atau yang lebih dikenal dengan sistem pemberi rekomendasi.

Recommender system atau sistem rekomendasi adalah sebuah teknik untuk membantu seorang *user* dalam mencari sebuah item berdasarkan ketertarikan masing-masing dalam memberikan efisiensi waktu. Sejak tahun 1990, sistem pemberi rekomendasi diterima secara luas dalam mengatasi kesulitan penyaringan data yang paling relevan dari sekumpulan informasi dalam jumlah besar dan kompleks (CC & Mohan, 2019). Kustomisasi teknologi pada sistem rekomendasi telah diadopsi secara luas oleh banyak perusahaan diberbagai bidang seperti Netflix, Amazon, Iflix, Youtube dan Spotify dalam mengembangkan bisnis mereka (Laksana, 2014).

Terdapat tiga pendekatan umum pada *recommender system* yang telah banyak digunakan dalam penelitian hingga saat ini, yaitu pendekatan *Content-based Filtering*, pendekatan *Collaborative Filtering*, dan pendekatan *Hybrid Recommendation System* (Zhang et al., 2018). *Content-based Filtering* adalah pendekatan yang menggunakan kesamaan konten item untuk memberikan rekomendasi kepada pengguna tanpa mengandalkan informasi sebelumnya *(rating)* (Zhang et al., 2018). *Collaborative Filtering* adalah pendekatan yang memberikan sebuah rekomendasi kepada pengguna berdasarkan preferensi pada profil (CC & Mohan, 2019). *Hybrid Recommender System* adalah pendekatan yang merupakan kombinasi dari dua pendekatan sebelumnya (Zhang et al., 2018).

Evolusi pengimplementasian metode *recommender system* berlanjut hingga saat ini. *Deep learning* yang merupakan salah satu penemuan dalam metode *recommender system* dapat mengatasi beberapa kekurangan dari teknlogi lain dan merevolusi arsitektur rekomendasi untuk meningkatkan kinerja pemberian rekomendasi (Zhang et al., 2018). Oleh karena itu, penelitian yang dilakukan menggunakan pendekatan prediksi *Collaborative Filtering* dengan mengimplementasikan *deep learning*, untuk menganalisis keakuratan prediksi yang dihasilkan berdasarkan teknologi *Neural Collaborative Filtering* pada dataset *MovieLens*.

2. METODE PENELITIAN



Gambar 1. Gambaran Umum Penelitian

Tahapan penelitian secara umum terbagi menjadi delapan tahapan, yaitu persiapan dataset, membuat *crosstab*, menyusun *embedding matrix*, menyusun *batch*, membentuk *embedding network*s, menyusun *schedule*, *training process*, dan evaluasi. Tahap persiapan dataset bertujuan untuk mempersiapkan dataset yang akan digunakan. Tahap membuat crosstab, bertujuan untuk membuat visualisasi dataset dengan mengubahnya ke dalam bentuk *crosstab* yang memperlihatkan hubungan antara *user* dan item. Tahap membentuk *embedding* matrix untuk mempersiapkan data ke dalam bentuk matrik data yang berisi hubungan vektor data berdimensi lebih rendah. Vektor data yang dihasilkan pada tahap sebelumnya kemudian dibagi menjadi beberapa bagian pada tahap menyusun *batch,* hal tersebut bertujuan untuk mengurangi ukuran data. Selanjutnya, tahap menyusun *embedding networks* untuk membangun sebuah jaringan *Neural Collaborative Filtering*. Tahap menyusun *schedule* dengan bantuan *Cyclical Learning Rate* (CLR) berfungsi untuk menerima parameter penjadwalan, bertujuan untuk membantu dalam menentukan *training epoch* berdasarkan hasil dari *learning rate*. Tahap *training* bertujuan untuk menghasilkan bobot dan performa model yang akan disimpan untuk tahap evaluasi. Tahap terakhir, yaitu evaluasi bertujuan untuk melihat rata-rata *error* terkecil perhitungan yang dihasilkan, parameter tersebut akan menunjukkan tingkat performa yang dihasilkan oleh metode yang telah digunakan.

2.1 Persiapan Dataset

Data yang digunakan untuk penelitian ini adalah data rating film dari situs MovieLens.org, yaitu sebuah laboratorium riset yang mempelajari mekanisme kajian dalam mesin rekomendasi yang dikembangkan oleh GroupLens. Dataset yang digunakan juga telah diimplementasikan untuk penelitian dalam bidang sistem rekomendasi lainnya.

Persiapan *dataset* dimulai dengan melakukan *input dataset rating* film dalam ekstensi atau format csv. ke dalam bentuk *dataframe*. Setelah dilakukan *input dataset* kemudian dilakukan konversi file .csv menggunakan fungsi *library pandas* untuk mengunggah data yang disimpan dalam variabel df dengan indeks dimulai dari 0. Struktur konversi file yang digunakan adalah *userId*, *movieId*, *rating* dan *timestamp*. Dataset yang digunakan memiliki atribut atau variabel yang meliputi konten item, *rating* item, *user* dan metadata sesuai dengan Tabel 1.

Tabel 1. Detail *Dataset MovieLens 100K*

No.	Nama Variabel	Jumlah Data
1.	ID Film	9066
2.	ID *User*	671
3.	*Rating* Film	100.004
4.	Timestamp	–

Data *MovieLens 100K* memiliki rincian data yang terdiri 671 pengguna, 9066 item (film), 100.004 jumlah *rating*, dan informasi konten item dalam bentuk *genre*. Kumpulan data berisi nilai dalam bentuk peringkat 1, 2, 3, 4, dan 5 dari 100.0004 *rating*. Beberapa contoh isi dari *dataset MovieLens 100K* dapat dilihat pada Tabel 2.

Tabel 2. Contoh Data *Rating MovieLens*

	***User*Id**	**MovieId**	***Rating***	**Timestamp**
0.	1	31	2,5	1260759144
1.	1	1029	3,0	1260759179
2.	1	1061	3,0	1260759182
3.	1	1129	2,0	1260759185
4.	1	1172	4,0	1260759205
...	...	...	...	...
99999.	671	6268	2,5	1065579370
100000.	671	6269	4,0	1065149201
100001.	671	6365	4,0	1070940363
100002.	671	6385	2,5	1070979663
100003.	671	6565	3,5	1074784724

2.2 Membuat *Crosstab*

Tahap membuat *crosstab* bertujuan untuk memvisualisasikan *user* dan item dalam mendapatkan *rating* tertinggi *user* dan item, di mana baris mengidentifikasikan sejumlah *user* yang memberikan jumlah *rating* untuk *movies* tertinggi atau disebut dengan *Top User* dan kolom mengidentifikasikan sejumlah *movies* yang mendapatkan jumlah *rating* tertinggi atau disebut sebagai *Top Movies*. Pembuatan *crosstab* mendefinisikan matrik interaksi item *user* dari umpan balik *user* (He et al., 2017).

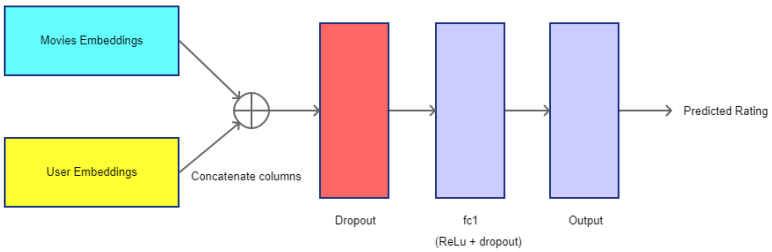
Proses yang dilakukan dalam membuat *crosstab* akan menampilkan 15 *rating* tertinggi dengan mengelompokkan sejumlah *userId* berdasarkan *rating*, kemudian dilakukan penyortiran dengan mengelompokan *userId* berdasarkan rating menjadi *Top User*. Kemudian dilakukan pengelompokan sejumlah *movieId* berdasarkan *rating*, lalu dilakukan penyortiran dengan mengelompokkan *movieId* berdasarkan rating menjadi *Top Movie*. Setelah didapatkan *Top User* dan *Top Movie* selanjutnya dilakukan penyaringan informasi *rating* dengan menghitung top data tertinggi dengan menghubungkan *Top User* dan *Top Movie* berdasarkan masing-masing *userId* dan *movieId*, penyatuan tersebut divisualisasikan menjadi bentuk visualisasi *crosstab*.

2.3 Menyusun *Embedding Matrix*

Tahapan yang dilakukan pada penyusunan *embedding matrix* bertujuan untuk mengubah bentuk data yang awalnya berupa *crosstab* menjadi bentuk matrik. *Embedding matrix* merepresentasikan bentuk data menjadi model yang mudah dipahami dalam melihat hubungan vektor data dan merupakan representasi dimensi yang lebih rendah.

Tahap awal yang dilakukan adalah mengecek apakah rating pada *userId* yang masuk dalam data *Top Rating* tidak kosong. Jika data *Top Rating* tidak kosong, maka dilakukan pengelompokkan jumlah *userId* dan *rating*. Id *user* yang masuk dalam kategori pengelompokan tersebut bersifat unik. Data *userId* yang tidak kosong tersebut akan menjadi sebuah *Unique User*, kemudian *unique user* tersebut akan dijadikan bentuk data *New User* berdasarkan *rating* dan *userId*. Hal yang sama juga dilakukan dengan data *movieId* yang menjadi sebuah *Unique Movies* kemudian dijadikan bentuk data *New Movies* berdasarkan *rating* dan *movieId*.

User dan *Movie* yang bersifat unik akan diinisialisasikan menjadi *New User* dan *New Movies* dan memiliki *index* yang berawal dari 0. Melalui batuan *pandas*, *dataframe* akan dibentuk *New User* dan *New Movies* yang membaca data berdasarkan *user_id* dan *movie_id* dan diinisialisasikan menjadi X. Sedangkan keseluruhan *rating* diinisialisasikan menjadi y. *Output* yang nantinya akan dikeluarkan adalah *user* dan *movie* dalam bentuk *embedding matrix*. Bentuk *dataset* X, yaitu sejumah *rating* dengan *user* dan *movie* dengan bentuk matrik, dan target *rating* dengan bentuk dataset y. Hasil yang diberikan adalah matrik *dataset* X yang ditunjukkan sesuai dengan Tabel 3 dan matrik *rating* yang ditunjukkan sesuai dengan Tabel 4.



Gambar 2. Skema *Embedding Networks*

Tabel 3. Contoh Matrik *Dataset* X

Index	*user*_id	movie_id
0	0	0
1	0	1
2	0	2
3	0	3
4	0	4
...	...	...
99999	670	7005
100000	670	4771
100001	670	1329
100002	670	1331
100003	670	2946

Tabel 4 Contoh Matrik Target y

Index	Rating
0	2,5
1	3,0
2	3,0
3	2,0
4	4,0
...	
99999	2,5
100000	4,0
100001	4,0
100002	2,5
100003	3,5

2.4 Penyusunan *Batch*

Tahap penyusunan *batch* bertujuan membagi *dataset* menjadi bagian kecil dengan melakukan *split dataset* (He et al., 2017). Pembagian dataset nantinya akan digunakan selama proses *training*. Proses dalam penyusunan *batch* dilakukan dengan membuat parameter untuk merepresentasikan objek yang akan dibuat, yaitu X atau *dataset* dalam bentuk *embedding* matrik, y atau target *rating*, jumlah data yang akan diambil pada setiap *batch* (*batch_size*) dengan ukuran 32, mengambil sejumlah data secara acak, dan mengubah bentuk *input* X dan y yang semula matrik menjadi sebuah *array*. Matrik X akan diacak sepanjang indeks pertamanya, *index* X atau *dataset* dalam bentuk *embedding* matrik adalah 0. Lalu dilakukan inisialisasi *self* terhadap X, y, *batch_size, shuffle,* dan jumlah ukuran X terhadap *batch,* langkah tersebut dilakukan dengan tujuan untuk membuat kelas yang menunjukkan iterasi pada *dataset* satu per satu, kemudian dilakukan perulangan.

Nilai yang dihasilkan *batch* pada X adalah *torch.LongTensor* dan nilai yang dihasilkan *batch* pada y adalah *torch.FloatTensor*. Sedangkan kedua *batch* yang akan dibentuk berdasarkan *input* X, y, dan *self.batch_size* yang bernilai 4. *Batch* yang dihasilkan berupa tensor *x_batch* dan *y_batch.*

2.5 Membuat *Embedding Network*

Tahap membuat *Embedding Network* bertujuan untuk membangun jaringan model *Neural Collaborative Filtering* (He et al., 2017). Proses awal dalam membuat *embedding networks*, yaitu membentuk *embedding layer* dengan melakukan inisialisasi fungsi yang dibutuhkan dalam pembuatan *embedding networks* melalui bantuan modul. Terlebih dahulu perlu dibentuk *dense network* dengan *embedding layer*, dimana *embedding layer* berfungsi sebagai pembawa informasi dari satu *layer* ke *layer* lainnya. Perlu digunakan *generator* untuk menghasilkan rangkaian *hidden layer* yang terletak di antara *input* dan *output* untuk mengaktifkan *dropout*. Fungsi *generator* ini menangkap nilai *hidden* dan *dropout* berdasarkan nilai pola tersembunyi. Pembentukan *hidden layer* menggunakan fungsi *generator* dengan menghasilkan persamaan dan *dropout layer* yang bergantung pada nilai dari hasil *hidden layer* menjadi *single* modul.

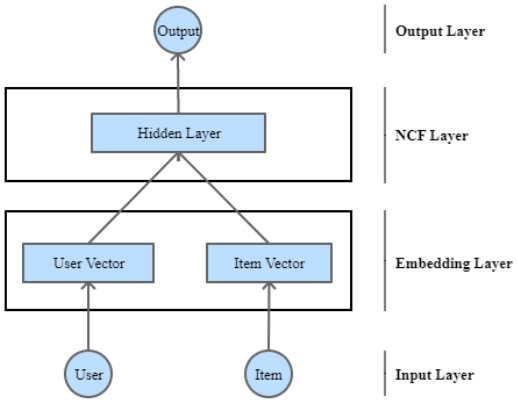
Proses selanjutnya adalah menempatkan *embeddings matrix* dan selanjutnya mengubah ID *integer* menjadi *array*, menyisipkan sekelompok *fully-connected layer* yang terdiri dari *hidden layer,* fungsi aktivasi, *output layer,* dan *loss function* dengan menggunakan MSE disertai *dropout* yang menghasilkan daftar *rating* yang diprediksi.

Pembuatan *neural networks* dapat membantu mengambil keputusan dalam melihat kemiripan antar *user* dan memprediksi *rating* tersebut dalam kumpulan data yang tersedia. Ilustrasi skema model *neural networks* yang terbentuk ditunjukkan sesuai pada Gambar 2.

Ilustrasi skema *embedding networks* menunjukkan setiap transaksi akan diekstraksi menjadi dua matrik atau bagian, yaitu vektor *user* dan vektor item. Vektor *user* dan vektor item kemudian direpresentasikan ke dalam *generalized matrix factorization* (GMF). Dua vektor *user* dan item digabungkan menjadi *multi layer perceptron* (MLP), lalu dibuat *embedding layer* dengan setiap *layer*nya yang menggunakan *dropout*, dimana setiap *embedding* menggunakan fungsi aktivasi ReLU.

Selanjutnya, menggabungkan *generalized matrix factorization* dengan *output* dari *neural networks layer* yang menghasilkan model *training* dengan regresi linier dan *loss function* yang digunakan adalah *mean square error* (MSE).

Proses selanjutnya adalah menerapkan *embedding matrix* yang sudah dikonversikan menjadi bentuk *array* kemudian didapatkan *layer* yang terhubung secara lengkap dengan *dropout,* dan didapatkan daftar prediksi *rating.* Aktifitas transfer membawa *input* yang diaktifasi dengan *Rectifier Linear Unit* (ReLU). Terdapat contoh model arsitektur *network* yang ditunjukkan pada Gambar 3.



Gambar 3. Contoh Arsitektur *Neural Networks*

Arsitektur *Neural Network* yang terbentuk berasal dari *input* sejumlah *user* dan item (*movie*). *Layer* pertama yang terbentuk adalah *input layer* yang berisi masukan *user* sejumlah 671 dan item (*movie*) sejumlah 9066. *Layer* kedua adalah *embedding layer* yang berisi *embedding user* yang berdimensi 671×150 dan *embedding movie* yang berdimensi 9066×150. Vektor *user* dan vektor item yang dihasilkan pada *layer* kedua akan dikonkatenasi menjadi *embedding matriks* dan akan menjadi *input* pada *layer* ketiga. *Layer* ketiga adalah *neural collaborative filtering* yang berisi *single hidden layer* dengan 300 *input* sampel, 100 *output* sampel, menggunakan *activation function* ReLU, dan *dropout* yang berisi parameter dengan nilai 0,5. *Layer* keempat adalah *output* yang berisi prediksi rating.

2.6 Menyusun *Schedule*

Tahap menyusun *schedule* betujuan untuk mencari optima lokal atau optima global yang terbaik dengan bantuan *Cyclical Learning Rate* (CLR). *Cyclical Learning Rate* merupakan salah satu fitur pada *library fast.ai* dan menggunakan *learning rate cosine annealing* dengan teknik *restart* untuk membandingkan konfigurasi yang tersedia.

Proses menyusun *schedule* mengimplementasikan CLR dengan menggunakan *method* yang menghasilkan daftar *learning rate* berdasarkan *training epoch*. Semua *layer* akan memiliki *learning rate* yang sama dan menghasilkan *single value.* Proses awal yang dilakukan dalam menyusun *schedule*, yaitu melakukan inisialisasi terhadap parameter CLR yang menunjukkan fungsi yang dapat dipanggil kembali. Fungsi tersebut akan menerima *training epoch* dan *learning rate* untuk mengembalikan tingkat *learning rate* baru. Kemudian proses selanjutnya adalah membuat fungsi yang menerima parameter dalam penjadwalan dan menghasilkan fungsi baru. Proses ini untuk mengurangi iterasi *learning rate* setelah beberapa *epoch* sehingga *optimizer* yang bekerja akan mencari optima lokal, global atau terbaik dan memperbarui bobot serta bias yang dihasilkan agar *error* berkurang.

2.7 *Training Procces*

Tahap *training procces* bertujuan untuk melihat kemiripan pola di antara *rating user* untuk mengisi *gap* dalam membuat sebuah prediksi pada data *rating* yang hilang. Persentase data yang digunakan sebagai set *training* dan *testing*, yaitu 80% data digunakan sebagai set *training* dan 20% data digunakan sebagai set *testing*. Pada perintah *train_test_split* diisi dengan parameter *array* X dan y yang menunjukkan array yang dipakai adalah X dan y dengan nilai *test_size* sebesar 0,2. Penelitian ini menggunakan 100.004 data sehingga 80.003 data digunakan sebagai data *training* dan 20.001 data digunakan sebagai data *testing.* Pada tahapan ini terdapat beberapa fase yang akan dilewati, yaitu fase train dan fase val.

Proses awal yang dilakukan pada tahap *training*, yaitu membagi *dataset* menggunakan fungsi *train_test_split* menjadi data *testing* dan data *training*. *Embedding network* yang telah dibuat pada proses sebelumnya diberi beberapa *input,* yaitu jumlah *user* dan *movies*, jumlah *factors* sebesar 150, tiga *hidden layer* dengan node 500, dan *embedding dropouts* sebesar 0,05.

Proses berikutnya, yaitu mengatur *training loop* menggunakan *Cyclical Learning Rate* (CLR) untuk menyimpan bobot terbaik model yang dihasilkan. Selanjutnya menerapkan beberapa parameter dengan menggunakan *optimizer* Adam, menerapkan parameter pada *learning rate* sebesar 0,001 dan 0,01 menggunakan *MSE loss* sebagai matrik untuk mengukur kualitas *networks* atau *loss function*. Kemudian menggunakan *learning rate cosine annealing* dengan teknik *restart* untuk membandingkan konfigurasi yang tersedia, yaitu ketika terdapat penurunan kecepatan pada setiap *batch* selama 2 *epoch*, maka kecepatan akan kembali kenilai aslinya.

Proses berikutnya menjalankan dua fase pada tahap *training*, yaitu fase train dan val dalam *training loop*. Pertama disebut fase *train*, selama fase *train* bobot *network* akan diperbarui dan mengubah *learning rate*. Fase kedua disebut *val,* pada fase ini digunakan untuk menguji kinerja model. Saat bobot

pada nilai *loss* berkurang, maka parameter model akan disimpan.

Setiap *batch* pada proses *training* data akan diacak secara *random* menggunakan hasil tensor *x_batch* dan *y_batch*. Saat sistem akan menghitung gradien, maka gradien akan di-*reset* kembali dengan tujuan meng-*update* model. Proses perhitungan *gradien* diaktifkan pada fase *train* yang menjadi mekanisme untuk mengaktifkan komputasi *gradien*, tujuannya adalah untuk menghitung *loss* yang dilewati oleh *output* pada *network*.

Pada fase *val* bobot dan *rate* tidak akan diperbarui. Di saat proses *training* gradien akan dihitung secara otomatis dan kemudian akan dilakukan perhitungan pada gradien. Fase *val* akan dilakukan sebuah operasi *back propagation* dengan memanggil fungsi *backward()* pada *loss* yang dihitung berdasarkan *input* dan *output*. Bobot *loss* selama proses berjalan akan dihasilkan dari jumlah nilai *loss* selama proses dan *loss.item*. Saat menghitung *loss batch* maka akan menghitung rata-rata *loss* dari *epoch* dengan cara memplot *loss* untuk setiap *epoch* dan menambahkan hasilnya ke *loss value* di setiap *epoch*. Rata-rata *loss* pada *batch* akan memberi perkiraan *epoch loss* yang berisi hilangnya seluruh *mini-batch* selama *training*.

Loss value didapatkan dengan menjumlahkannya dan menghitung rata-rata setelah *epoch* selesai. *Training loss* ini akan digunakan untuk melihat seberapa baik kinerja model yang dihasilkan pada *training dataset*. Setiap *epoch training* akan mengevaluasi performa model yang dilatih. Pada fase *val* tidak diperlukan parameter untuk diaktifkan dan parameter gradien tidak akan diperbarui bagian di tensor mana pun. Karena di fase *val*, *optimizer* akan bekerja untuk mengetahui parameter model mana saja yang akan di *update*. Fase ini juga akan menyimpan bobot model terbaik. Jika *epoch loss* bernilai lebih kecil dari *best_loss* maka *output* kenaikan *loss* akan dikeluarkan sesuai dengan *epoch* terakhir. Proses *training* berakhir setelah tidak ada peningkatan setelah 10 *training epoch* berturut-turut dan proses *looping* akan berhenti. Hasilnya bobot terbaik akan disimpan selama *training,* dan menggunakan data *testing* untuk validasi data dalam melihat performa model akhir.

2.8. Evaluasi

Tahap evaluasi bertujuan untuk mengetahui seberapa besar akurasi yang dihasilkan oleh metode yang digunakan dalam penelitian dan sebagai landasan terhadap kelayakan hasil rekomendasi. Penelitian ini menggunakan RMSE sebagai evaluasi terhadap hasil prediksi dataset yang menerapkan metode *Neural Collaborative Filtering.* Setelah didapatkan hasil model performa yang berasal dari tahap *training* kemudian dilakukan estimasi *error* prediksi *rating* menggunakan rumus RMSE dengan menggunakan data *testing*. Metrik skor regresi *Root Mean Square Error* (RMSE) akan menghitung rata-rata dari kuadrat perbedaan yang berasal dari hasil *rating* aktual dan prediksi. Semakin besar tingkat kesalahan yang dihasilkan akan berdampak pada nilai RMSE yang semakin besar, sehingga menunjukkan tingkat kesalahan yang tinggi. Rumus untuk setiap metrik skor seperti ditampilkan dalam Persamaan 1:

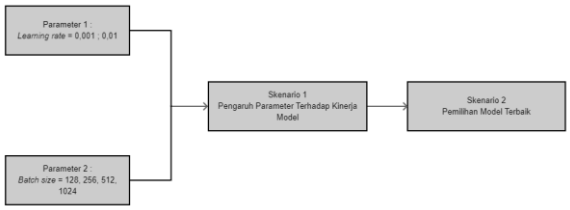
$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n}} \quad (1)$$

Keterangan:
n : Jumlah item yang dirating pengguna
y : Rating prediksi
x : Rating sebenarnya

3. HASIL DAN PEMBAHASAN

3.1. Skenario Pengujian

Skenario pengujian membahas data pengujian dan skenario pengujian terhadap tingkat akurasi menggunakan *Neural Collaborative Filtering* (NCF) pada dataset *MovieLens* 100K. Gambaran umum skenario pengujian ditunjukkan pada Gambar 4.



Gambar 4. Skenario Pengujian

Nilai parameter yang diujikan dengan nilai *learning rate* adalah 0,001 dan 0,01, sedangkan nilai *batch size* adalah 128, 256, 512 dan 1024. Kedua parameter akan dikombinasikan untuk mendapatkan hasil terbaik. Kinerja dari setiap kombinasi dilakukan pada tahap *training* untuk mendapatkan nilai akurasi. Hasil perhitungan akurasi akan dibandingkan untuk mendapatkan hasil terbaik.

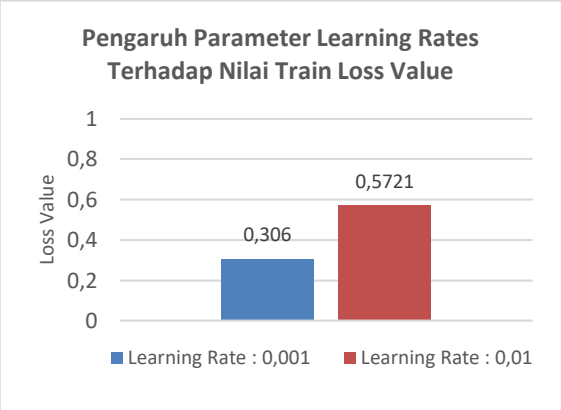
3.2. Hasil dan Analisa Skenario 1

Skenario 1 melakukan analisa berdasarkan kombinasi parameter *learning rate* dan *batch size.* Pengaruh dari kombinasi ini akan dihitung untuk mendapatkan bobot nilai. Pada Tabel 6 menunjukkan kombinasi parameter yang digunakan. Pada Gambar 5 menunjukkan hasil *loss value* pada parameter *learning rate*, sedangkan pada Gambar 6 menunjukkan hasil *loss value* pada parameter *batch sizes.*

Tabel 6. Hasil *Loss Value* Kombinasi Parameter yang Digunakan

No	Kombinasi	Parameter		Fase	
		Learning Rate	*Batch Size*	*Train*	*Val*
1.	Kombinasi-1	0,001	128	0,5639	0,8425

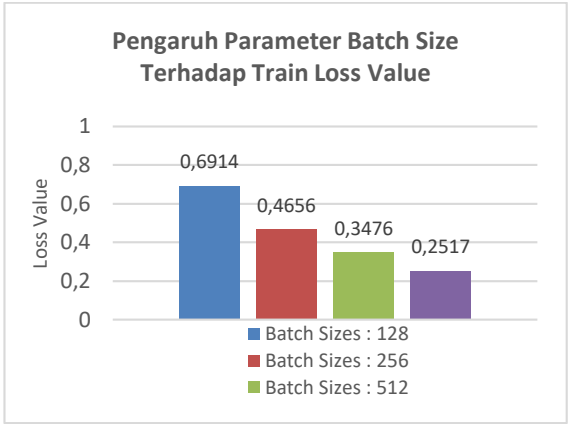
No	Kombinasi	Parameter		Fase	
		Learning Rate	*Batch Size*	*Train*	*Val*
2.	Kombinasi-2	0,001	256	0,2902	0,9205
3.	Kombinasi-3	0,001	512	0,2116	0,9342
4.	Kombinasi-4	0,001	1024	0,1582	0,9019
5.	Kombinasi-5	0,01	128	0,8189	0,9035
6.	Kombinasi-6	0,01	256	0,6409	0,9138
7.	Kombinasi-7	0,01	512	0,4835	0,9600
8.	Kombinasi-8	0,01	1024	0,3452	0,9660



Gambar 5. Pengaruh *Learning Rate* Terhadap *Loss Value*

Parameter *learning rate* menunjukkan bagaiman perubahan bobot. Jika *learning rate* rendah akan berakibat pada proses *training* yang lebih lama dalam pencarian nilai *loss function*. Sedangkan nilai *learning rate* yang tinggi berakibat pada perubahan bobot yang terlalu besar. Pengaruh parameter *learning rate* terhadap bobot *loss value* dapat terlihat pada Gambar 5 dari grafik tersebut terlihat bahwa *learning rate* dengan nilai 0,001 menghasilkan rata-rata *loss value* lebih rendah daripada *learning rate* dengan nilai 0,01 yang memiliki *loss* lebih besar. Nilai *learning rate* 0,001 melakukan perubahan terhadap variabel secara kecil, hal tersebut lebih baik dibandingkan dengan nilai *learning rate* 0,01 sehingga memberi peluang pada model untuk mempelajari kumpulan bobot menjadi lebih optimal.

Batch size yang bernilai kecil akan berdampak pada waktu komputasi yang lebih lama. Sedangngkan menggunakan *batch size* yang lebih besar akan mengurangi kemampuan model dalam memberikan hasil yang kurang sesuai. Pengaruh parameter *batch size* terhadap bobot *loss value* dapat terlihat pada Gambar 6., grafik tersebut terlihat *batch size* dengan nilai 1024 menghasilkan rata-rata nilai *loss value* yang lebih baik dari pada *batch size* dengan nilai 128, 256, dan 512.



Gambar 6. Pengaruh Parameter *Batch Size* Terhadap *Loss Value*

Semakin besar nilai *batch size* berdasarkan gambar maka nilai *loss* yang didapatkan semakin kecil. Walaupun nilai *batch size* tersebut lebih besar dari yang lainnya, dapat ditunjukkan hasil pada *batch size* 1024 yang paling optimal karena nilai *loss* yang paling mendekati 0, hal tersebut memungkinkan percepatan komputasi untuk mempelajari model dalam mencari nilai *loss value* yang lebih rendah karena menunjukkan jumlah sampel data yang disebarkan ke *neural networks* lebih besar daripada yang lainnya dan memberikan.

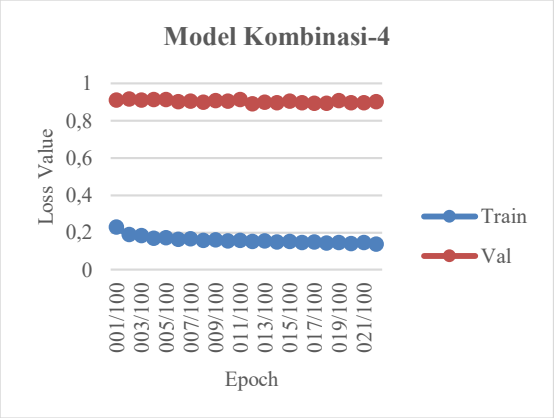
3.3. Hasil dan Analisa Skenario 2

Skenario 2 melakukan pemilihan model terbaik dari model yang telah diperoleh dari skenario 1 pada setiap pasang label. Parameter *learning rate* dengan nilai 0,001 memperoleh rata-rata *loss value* terendah. Parameter *batch size* dengan nilai 1024 memperoleh rata-rata *loss value* terendah. Dari kedua parameter tersebut dapat disimpulkan bahwa kombinasi parameter terbaik terdapat pada Kombinasi-4 dengan *learning rate* bernilai 0,001 dan *batch size* bernilai 1024. Tabel 7 menunjukkan *loss value* pada model Kombinasi-4 pada setiap model.

Tabel 7. *Loss Value* Model Pada Kombinasi Kombinasi-4

No	*Epoch*	*Train*	*Val*
1.	001/100	0,2271	0,9101
2.	002/100	0,1861	0,9144
3.	003/100	0,1824	0,9082
4.	004/100	0,1678	0,9126
5.	005/100	0,1701	0,9114
6.	006/100	0,1612	0,9002
7.	007/100	0,1637	0,9025
8.	008/100	0,1551	0,8989
9.	009/100	0,1583	0,9069
10.	010/100	0,1531	0,9046
11.	011/100	0,1571	0,9110
12.	012/100	0,1494	0,8898
13.	013/100	0,1535	0,8981
14.	014/100	0,1465	0,8954
15.	015/100	0,1505	0,9037
16.	016/100	0,1440	0,8942
17.	017/100	0,1476	0,8929
18.	018/100	0,1419	0,8927
19.	019/100	0,1453	0,9062
20.	020/100	0,1393	0,8942

No	*Epoch*	*Train*	*Val*
21.	021/100	0,1437	0,8948
22.	022/100	0,1356	0,8996



Gambar 7. Grafik Model Kombinasi-4

Berdasarkan Tabel 7, hasil bobot *loss value* pada *train* menunjukkan penurunan *loss* dan memperoleh hasil terbaik pada *epoch* ke 022/100 dengan *train loss* sebesar 0,1356 dan *val loss* sebesar 0,8896. Melalui penelitian yang telah dilakukan parameter yang memberikan hasil paling optimal adalah *learning rate* dengan nilai 0,001 dan *batch size* dengan nilai 1024. Berdasarkan hasil yang ditunjukkan berhubungan dengan nilai *learning rate* yang kecil sehingga perubahan bobot secara kecil memberikan kesempatan pada model untuk mempelajari kumpulan bobot agar menjadi lebih optimal. dan nilai *batch size* yang digunakan menyebarkan jumlah sampel data ke *neural network* yang lebih besar sehingga nilai *loss function* lebih rendah. Kemudian di dapatkan hasil RMSE dengan nilai 0,91467.

4. KESIMPULAN

Metode *Neural Collaborative Filtering* dapat diterapkan pada *Film Recommender System* melalui kombinasi pendekatan *Collaborative Filtering* dan metode *deep learning*, yaitu *neural netwoks.* Parameter *learning rate* dan *batch size* mempengaruhi kinerja model memperoleh kinerja terbaik ketika *learning rate* bernilai 0,001 dan *batch size* dengan nilai 1024 dengan hasil *train loss value* sebesar 0,1356 dan *val loss value* sebesar 0,8898. Model prediksi sistem rekomendasi menunjukkan hasil nilai RMSE sebesar 0,91467.

DAFTAR PUSTAKA

AL AMIN, A. 2021. Mereduksi Error Prediksi Pada Sistem Rekomendasi Menggunakan Pendekatan Collaborative Filtering Berbasis Model Matrix Factorization. EXPLORE, 11(2), 8-14. https://doi.org/10.35200/EXPLORE.V11I2.434.

BOBADILLA, J., ALONSO, S., & HERNANDO, A. 2020. Deep learning architecture for collaborative filtering recommender systems. *Applied Sciences*, *10*(7), 2441. https://doi.org/10.3390/app10072441.

CHAVARE, S. R., AWATI, C. J., & SHIRGAVE, S. K. 2021. Smart recommender system using deep learning. In 2021 6th International Conferense on Inventive Computation Technologies. (ICICT) (pp. 590-594). IEEE. https://doi.org/10.1109/ICICT50816.2021.9358580.

DWICAHYA, IMAM. 2018. Perbandingan Sistem Rekomendasi Film Metode User-based dan Item-based Collaborative Filtering. Universitas Sanata Dharma.

CAKRANINGRAT, R. 2011. Sistem pendukung Keputusan untuk UMKM. [ebook]. UBX Press. Tersedia melalui: Perpustakaan Universitas BX <http://perpustakaan.ubx.ac.id> [Diakses 1 Juli 2021]

GARANAYAK, M. *dkk*. 2019. Recommender system using item based collaborative filtering (CF) and K-means. *International Journal of Knowledge-based and Intelligent Engineering Systems*, *23*(2), 93-101. https://doi.org/ 10.3233/KES-190402.

GIRSANG, A. S., & WIBOWO, A. 2021. Neural collaborative for music recommendation system. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1071, No. 1, p. 012021). IOP Publishing. https://iopscience.iop.org/article/10.1088/1757-899X/1071/1/012021

HE, X. *dkk*. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173-182). https://doi.org/10.1145/3038912.3052569.

LAKSANA, E. A. 2014. Collaborative Filtering dan Aplikasinya. *Jurnal Ilmiah Teknologi Infomasi Terapan*, *1*(1). https://doi.org/10.33197/jitter.vol1.iss1.2014.44

LIU, Y. *dkk*. 2018. A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering. *Big Data Mining and Analytics*, *1*(3), 211-221. https://doi.org/10.26599/BDMA.2018.9020019

MASRURI, F., & MAHMUDY, W. F. 2007. Personalisasi web e-commerce menggunakan recommender system dengan metode item-based collaborative filtering. *Jurnal Ilmiah Kursor*, *3*(1).

Grouplens.org. 1998. MovieLens 100K dataset. Tersedia di:

<https://grouplens.org/datasets/movielens/100k/> [Diakses 3 September 2021].

CC, N., & MOHAN, A. 2019. A social recommender system using deep architecture and network embedding. *Applied Intelligence*, *49*(5), 1937-1953. https://doi.org/10.1007/s10489-018-1359-z

RIZKY, M. I., ASROR, I., & MURTI, Y. R. 2020. Sistem Rekomendasi Program Studi Untuk Siswa Sma Sederajat Menggunakan Metode Hybrid Recommendation Dengan Content Based Filtering Dan Collaborative Filtering. *eProceedings of Engineering*, *7*(1).

ROCHMAWATI, N. *dkk*. 2021. Analisa Learning Rate dan Batch Size pada Klasifikasi Covid Menggunakan Deep Learning dengan Optimizer Adam. JIEET (Journal of Information Engineering and Educational Technology), 5(2), 44-48. https://doi.org/10.26740/jieet.v5n2.p44-48

PUTRA, A. I., & SANTIKA, R. R. 2020. Implementasi Machine Learning dalam Penentuan Rekomendasi Musik dengan Metode Content-Based Filtering. *Edumatic: Jurnal Pendidikan Informatika*, *4*(1), 121-130. https://doi.org/10.29408/edumatic.v4i1.2162

PRATAMA, Y. A. *dkk*. 2013. Digital Cakery Dengan Algoritma Collaborative Filtering, 14(1), 79–88.

SAHOO, A. K. *dkk*. 2019. DeepReco: deep learning based health recommender system using collaborative filtering. *Computation*, *7*(2), 25. https://doi.org/10.3390/computation7020025

SARWAR, B. *dkk*. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285-295). https://doi.org/10.1145/371920.372071

SHAKIROVA, E. 2017. Collaborative filtering for music recommender system. In *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)* (pp. 548-550). IEEE. https://doi.org/10.1109/EIConRus.2017.7910613

SMITH, L. N. 2017. Cyclical learning rates for training neural networks. In 2017 IEEE winter conference on applications of computer vision (WACV) (pp. 464-472). IEEE. https://doi.org/10.1109/WACV.2017.58

YOSHUA, I., & BUNYAMIN, H. 2021. Pengimplementasian Sistem Rekomendasi Musik Dengan Metode Collaborative Filtering. *Jurnal STRATEGI-Jurnal Maranatha*, *3*(1), 1-16.

ZARZOUR, H., AL-SHARIF, Z. A., & JARARWEH, Y. 2019. RecDNNing: a recommender system using deep neural network with user and item embeddings. In *2019 10th International Conference on Information and Communication Systems (ICICS)* (pp. 99-103). IEEE. https://doi.org/10.1109/IACS.2019.8809156

ZHANG, L. *dkk*. 2018. A recommendation model based on deep neural network. *IEEE Access*, *6*, 9454-9463. https://doi.org/10.1109/ACCESS.2018.2789866

ZHANG, S. *dkk*. 2019. Deep learning based recommender system: ` A survey and new perspectives. *ACM Computing Surveys (CSUR)*, *52*(1), 1-38. https://doi.org/10.1145/3285029.

Halaman ini sengaja dikosongkan