JATIKOM: Jurnal Aplikasi dan Teori Ilmu Komputer | ISSN: 2615-577X (Online)
https://ejournal.upi.edu/index.php/JATIKOM

Mesin Rekomendasi Film Menggunakan Metode Deep Autoencoder

Film Recommendation Machine Using Deep Autoencoder Method

Muhamad Adie Maulana[1], Yaya Wihardi, S.Kom, M.Kom.[2], Erna Piantari, S.Kom., M.T.[3]

Prodi Studi Ilmu Komputer Departemen Pendidikan Ilmu Komputer Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam Universitas Pendidikan Indonesia
Jl. Dr. Setiabudhi No. 229 Bandung 40154 Jawa Barat – Indonesia

[1]muhamadadiem@student.upi, [2]yayawihardi@upi.edu,
[3]erna.piantari@upi.edu

Abstrak— **Layanan penyedia jasa berbasis internet seperti Netflix, Iflix, Amazon Prime, dan lainnya telah mengalami peningkatan total waktu tonton secara drastis dalam kurun waktu sepuluh tahun kebelakang. Pada tahun 2017, pengguna layanan Netflix secara kolektif telah menonton konten Netflix selama 140 juta jam per hari dan mendapatkan pendapatan sebesar 11 milyar dollar amerika [1]. Hampir 80% waktu yang ditonton di Netflix berasal dari mesin rekomendasi yang dibangun oleh Netflix. Hampir semua mesin rekomendasi menggunakan metode collaborative filtering, namun metode Restricted Boltz-mann machines untuk membangun collaborative filtering [2], menunjukkan hasil prediksi yang konsisten meskipun data pelatihan meningkat, sehingga masalah skabilitas dapat teratasi. Konsep yang diadopsi dari model deep learning adalah kemampuan metode ini untuk mengekstraksi fitur robust secara unsupervised melalui rekonstruksi input autoencoder. Berdasarkan hasil pengujian dengan data uji 6040 user, 3883 item (film), dan 1.000.209 rating menghasilkan nilai loss yang rendah yaitu 0.7322 dan nilai RMSE 0,7227 dengan menggunakan metode autoencoder yang telah dimodifikasi.**

Kata Kunci: *Collaborative-filtering, Autoencoder, Deep learning, Recommendation machine*

Abstract— **Internet-based service providers such as Netflix, Iflix, Amazon Prime, and others have experienced a drastic increase in total viewing time in the past ten years. In 2017, Netflix service users collectively watched 140 million hours of Netflix content per day and earned US$11 billion in revenue [1]. Nearly 80% of the time watched on Netflix comes from the recommendation engine built by Netflix. Almost all recommendation engines use the collaborative filtering method, but the Restricted Boltz-mann machines method for developing collaborative filtering [2], shows consistent prediction results even though the training data increases, so that scalability problems can be resolved. The concept adopted from the deep learning model is the ability of this method to extract robust features in an unsupervised manner through autoencoder input reconstruction. Based on the test results with test data of 6040 users, 3883 items (films), and 1.000.209 ratings, it resulted in a low loss value of 0.7322 and an RMSE value of 0.7227 using the modified autoencoder method.**

Keywords: *Collaborative-filtering, Autoencoder, Deep learning, Recommendation machine*

I. PENDAHULUAN

Layanan penyedia jasa berbasis internet seperti Netflix, Iflix, Amazon Prime, dan lainnya telah mengalami peningkatan total waktu tonton secara drastis dalam kurun waktu sepuluh tahun kebelakang. Pada tahun 2017, pengguna layanan Netflix secara kolektif telah menonton konten Netflix selama 140 juta jam per hari dan mendapatkan pendapatan sebesar 11 milyar dollar amerika [1]. Faktanya, hampir 80% waktu yang ditonton di Netflix berasal dari mesin rekomendasi yang dibangun oleh [1].

Secara umum, mesin rekomendasi adalah algoritma yang ditujukan untuk menyarankan item yang relevan kepada pengguna (contohnya: film untuk ditonton, teks untuk dibaca, produk untuk dibeli atau apa pun tergantung pada industrinya) . Beberapa metode populer yang digunakan dalam membuat mesin rekomendasi yaitu content based filtering, collaborative filtering dan hybrid. content based filtering memanfaatkan interaksi antara konten item dengan profil pengguna [3], dimana yang termasuk konten item disini seperti genre, author, dan lain-lain. Collaborative filtering (CF) bekerja dengan membangun database preferensi konsumen terhadap suatu item. Seperti promosi dari mulut ke mulut, collaborative filtering memberikan prediksi rating dan personal rekomendasi berdasarkan yang disukai pengguna lain yang mempunyai selera yang sama [4]. Sementara hybrid merupakan penggabungan content based filtering dan collaborative filtering.

Tidak seperti metode collaborative filtering yang hanya mengandalkan interaksi item-user, metode content based menggunakan informasi tambahan tentang user atau item [5]. Jika kita mengambil contoh sistem rekomendasi film, informasi tambahan ini dapat berupa, usia, jenis kelamin, pekerjaan atau informasi pribadi lainnya dari user serta kategori, aktor utama, durasi atau karakteristik lainnya untuk film (item). Meskipun mempunyai kelebihan untuk memberi rekomendasi secara personal, namun salah satu kekurangannya yaitu metode ini hanya memberi rekomendasi berdasarkan preferensi penggunannya, yang mengakibatkan

mesin rekomendasi hanya memberi rekomendasi yang sangat terbatas [5].

Pendekatan awal untuk Collaborative Filtering, mengasumsikan bahwa pengguna yang memiliki kesamaan data dengan pengguna yang lain, pasti mempunyai kesamaan minat. Misalnya, pengguna yang mendengarkan musik jazz juga mendengarkan musik blues, maka pengguna lain yang mendengarkan music jazz yang serupa kemungkinan mendengarkan musik blues juga.

Beberapa algoritma telah dikembangkan untuk metode Collaborative Filtering, seperti algoritma memory based yang menggunakan teknik statistika untuk mencari kemiripan preferensi pengguna, namun teknik ini membutuhkan akses ke seluruh dataset setiap kali melakukan prediksi, sehingga membutuhkan banyak memory, yang berakibat akan ditemukan kerugian ketika menghadapi sparse dataset atau data yang jarang (bolong-bolong) dan masalah skalabilitas. Oleh karena itu algoritma model based diusulkan untuk menghadapi masalah ini. Ide awal algoritma ini adalah membangun model dari kumpulan rating pengguna, menggunakan algoritma deep learning dan mengambil pendekatan probabilistik untuk menghitung nilai yang diharapkan berdasarkan preferensi pengguna pada item lain.

Penggunaan deep learning telah melakukan terobosan dalam bidang image recognition [6]. Hal ini memicu minat para ilmuwan untuk membangun mesin rekomendasi menggunakan deep learning. Konsep yang diadopsi dari model deep learning adalah kemampuan metode ini untuk mengekstraksi fitur robust secara unsupervised melalui rekonstruksi input autoencoder.

Oleh karena itu, penelitian ini diharapkan bisa menyelesaikan masalah sparse data, skabilitas, dan meningkatkan akurasi dalam pemberian rekomendasi kepada user.

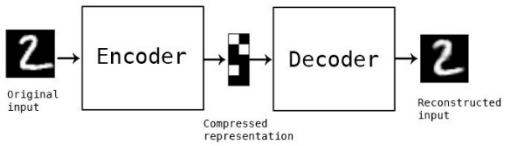
II. PENELITIAN TERKAIT

Penelitan sebelumnya yang dilakukan oleh Naiyan Wang, dan Dit-Yan Yeung [7] menggunakan metode autoencoder untuk mesin rekomendasi pencarian artikel. Autoencoder adalah tipe jaringan saraf tiruan yang digunakan untuk mempelajari data coding secara unsupervised learning atau pemelajaran tak terarah. Penggunaan Deep Learning untuk mesin rekomendasi pun telah dilakukan oleh Salakhutdinov, dkk [2]dengan menggunakan metode Restricted Boltz-mann machines untuk membangun collaborative filtering, menunjukkan hasil prediksi yang konsisten meskipun data pelatihan meningkat, sehingga masalah skabilitas dapat teratasi.

III. METODE AUTOENCODER

Autoencoder adalah model neural network yang memiliki input dan output yang sama [8]. Autoencoder mempelajari data input dan berusaha untuk melakukan rekonstruksi terhadap data input tersebut. [7]:

Autoencoder biasa digunakan untuk mengurangi dimensi dari features (Dimensionality Reduction). Jika kita mempunyai data yang mempunyai dimensi yang sangat tinggi (data dengan jumlah features yang sangat banyak) bisa jadi tiap features yang ada tersebar pada setiap dimensi dari data sehingga setiap data yang ada terlihat sangat berbeda. Untuk mengatasi masalah tersebut kita membutuhkan data yang sangat banyak atau mengurangi dimensi data tersebut. Beberapa metode yang bisa digunakan adalah PCA, t-SNE ataupun Autoencoder [8]



Gambar. 1 Skema Autoencoder

Autoencoder terdiri dari dua bagian utama yaitu encoder dan decoder. Encoder. Diantara encoder dan decoder, terdapat code layer atau bisa juga dibilang target layer atau hidden layer. Jumlah neuron pada code layer adalah jumlah dimensi yang kita harapkan untuk mengurangi dimensi dari data kita [8]/

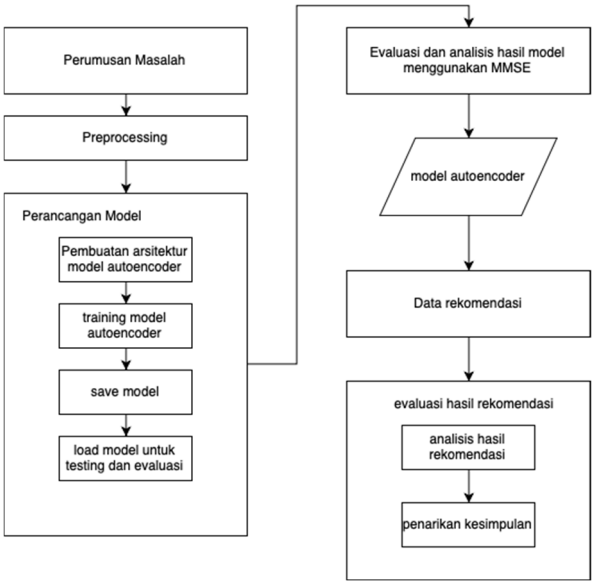
Autoencoder adalah neural network yang mampu merekonstruksi input. Ide dasar autoencoder tidak jauh dari konsep dekom- posisi/dimentionality reduction menggunakan singular value decomposition. Diberikan dataset X, kita ingin mensimulasikan pencarian matriks Xˆ yang merupakan sebuah low rank approximation dari matriks asli.

$$E(\theta) = \frac{1}{N} \sum_{i=j}^{Z} (X_{i[j]} - y_{i[j]})^2 \quad (1)$$

Kita memberi input matriks X pada autoencoder, kemudian ingin autoencoder tersebut menghasilkan matriks yang sama. Dengan kata lain, desired output sama dengan input. Apabila di- hubungkan dengan ANN , error function untuk melatih autoencoder diberikan pada persamaan diatas, dimana y adalah output dari jaringan dan Z adalah dimensi output, N adalah banyaknya in- stans dan xi adalah data ke-i (feature vector ke-i).

IV. METODOLOGI PENELITIAN

Pada sub bab ini, akan dipaparkan desain penelitian dari program yang dibuat dalam skripsi ini.



Gambar. 2 Desain Penelitian

Gambar. 2 merupakan desain penelitian yang dibuat oleh penulis, tahapan-tahapannya yaitu:

1) Perumusan Masalah: Merupakan tahap awal penelitian. Proses yang terjadi di tahap persiapan yaitu dimulai dari mengidentifikasi masalah yang akan dibahas, kemudian merumuskan masalah, lalu menentukan metode atau algoritma yang akan digunakan untuk menyelesaikan masalah tersebut, dan yang terakhir adalah menentukan model penelitian untuk membantu penyelesaian masalah..

2) Praprocessing: Pada tahap ini data akan masuk ke tahap praproses data, file .dat akan diubah menjadi file .csv menggunakan library panda agar dapat diproses. Karena indeks dimulai dari 1 maka semua userid dan movie id akan dikurangi satu agar indeks nya dimulai dari 0, hal ini tidak akan mempengaruhi training karena semua id berurutan

3) Perancangan Model: Pada tahap ini penulis mempersiapkan untuk membangun sistem perangkat lunak dengan arsitektur model yang telah didesain. Pertama adalah praproses data yang didapatkan dari situs Movielens, lalu membagi data train dan data test. Setalah itu data akan diubah menjadi matriks agar dapat di prediksi oleh model yang dibuat pada tahap selanjutnya. Lalu pada tahap terakhir model akan melakukan proses training dan didapatkan hasil training berupa model.

4) Evaluasi dan Analisis Hasil Model: Ditahap ini penulis mengevaluasi dan analisis hasil yang sudah didapatkan berdasarkan tahap pengujian atau training dalam bentuk grafik garis. Analisa hasil model berupa perbandingan hasil prediksi pada dataset yang digunakan pada model yang dibuat menggunakan RMSE.

5) Data Model Prediksi: Data testing akan berupada hasil rekomendasi yang memuat Userid, Moviesid, dan Rating. Akan ditampilkan 10 nilai tertinggi yang didapatkan hasil testing.

6) Evaluasi Hasil Rekomendasi: Pada tahap ini dilakukan analisis terhadap hasil rekomendasi yang dibuat dan pernarikan kesimpulan dari hasil penelitian tersebut

V. HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang hasil penelitian dan pembahasan dari masing-masing hasil penelitiannya. Pengembangan perangkat lunak akan dijelaskan ke dalam beberapa sub sub bab berdasarkan metode *waterfall.*

A. Pengumpulan Data

Data set diambil dari Movielens.org sebuah lembaga riset yang memfokuskan kajian tentang mesin rekomendasi (https://movielens.org/). Data set yang dipilih adalah Movielens, data ini terdiri dari 6040 user, 3883 item (film), 1.000.209 rating, serta info konten item berupa genre. Nilai rating yang terdapat pada data set adalah 1, 2, 3, 4, dan 5. Dari 1.000.000 rating. Untuk mempermudah proses komputasi, Tiap user memiliki minimal 20 rating item yang dapat digunakan sebagai data history untuk menghitung prediksi dan menghasilkan rekomendasi. Sebagian besar pola user memberi rating tersebar dari 1 hingga 5 sehingga menggambarkan preferensi user terhadap item.

B. Perancangan Model

Pada tahap ini model akan dirancang, dari tahapan memasukan dataset ke Jupyter Notebook sampai hasil training beres menghasilkan model.

C. Load Dataset

Data set yang diambil adalah berupa matrix user-item yang terbentuk dari aktivitas user yang telah memberikan rating terhadap film. Aktivitas itulah yang disimpan dan dikelola oleh MovieLens seperti pada gambar 3.

Ada 2 file yang akan diproses di tahap praproses data. Yang pertama yaitu file Movie.dat, dan Ratings.dat. Movie.dat berisi 3883 file film yang terdiri dari MovieID, judul film, dan genre. File Ratings.dat berisi 1.000.209 data rating yang berisi UserID, MoviesID, dan Rating. Setiap user telah memberi rating minimal 20 rating untuk 20 judul film.

```
1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
```

Gambar. 3 File movies.dat yang berisi atribut movieid, judul film, dan genre.

```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
1::595::5::978824268
```

Gambar. 4 File rating.dat yang berisi userid, movieid, rating, dan timestamp.

D. Pra Proses Data

Pada tahap ini data yang sudah diload akan masuk ke tahap praproses data, file .dat akan diubah menjadi file .csv menggunakan library panda agar dapat diproses. Karena indeks dimulai dari 1 maka semua userid dan movie id akan dikurangi satu agar indeks nya dimulai dari 0, hal ini tidak akan mempengaruhi training karena semua id berurutan.

L2 Regularization	Optimizer	Learning Rate	Epoch	Batch Size	Activations	Dropout
0.001	Adam	0.0001	1000	256	SELU+SELU	0.8

```
ratings = pd.read_csv(rating.dat),
ratings.to_csv(RATINGS_CSV_FILE,

columns=['user_emb_id','movie_emb_id',   'rating',
'timestamp'])
```

Pseudo code untuk merubah file .dat menjadi .csv

Pada baris pertama code, file .dat akan dimasukan terlebih dahulu, lalu menambahkan kolom-kolom header karena pada file .dat tidak terdapat header.

user_emb_id	movie_emb_id	rating	timestamp
4032	452	4	965513126
4291	3696	4	965274731
3312	355	4	968914033
3789	777	4	966040424
5462	3460	4	959900645

Gambar. 5 Dataframe rating.csv

Sebelum membuat arsitektur model, terlebih dahulu dataframe akan dirubah kedalam bentuk matriks agar dapat diaplikasikan kedalam model. Matriks berupa MxN dimana R(i, j) adalah rating yang diberikan oleh user i terhadap item j.

$$\begin{matrix} 4 & 4 & 5 & \dots & 3 & 1 \\ 2 & 5 & 2 & \dots & 2 & 4 \\ 2 & 5 & 4 & \dots & 1 & 2 \end{matrix} \quad (2)$$

```
INPUT:
Dataframe. Coloumn=['userid', 'movieid',
'rating']
Jumlah baris: int jumlah user
Jumlah kolom: int jumlah movie

OUTPUT:
            Matriks: 2D numpy array
```

Pseudo code untuk merubah bentuk dataframe menjadi matriks

E. Arsitektur Model

Setelah data diubah menjadi matriks, arsitektur model deep autoencoder dibuat. deepautoencoder adalah model neural network yang memiliki input dan output yang sama (Liou, Cheng, Liou, & Liou, 2014). Autoencoder mempelajari data input dan berusaha untuk melakukan rekonstruksi terhadap data input tersebut. (Wang & Yeung, 2013). Input yang digunakan dalam kasus ini yaitu matriks user dan item dari dataset yang telah melalui tahap praproses sebelumnya. Lalu input akan masuk ke dalam layer encoder untuk dienstraksi data nya, setelah itu masuk ke proses decoder. Lalu output yang keluar yaitu hasil rekonstruksi input.

Layer (type)	Output Shape	Param #
UserRating (InputLayer)	[(None, 3952)]	0
EncLayer0 (Dense)	(None, 256)	1011968
LatentSpace (Dense)	(None, 512)	131584
dropout_1 (Dropout)	(None, 512)	0
DecLayer0 (Dense)	(None, 256)	131328
UserScorePred (Dense)	(None, 3952)	1015664

Gambar. 6 Arsitektur Autoencoder

F. Training Model

Setelah arsitektur dibuat maka data akan dilakukan training menggunakan data train. Pengaturan yang digunakan akan ditampilkan pada table 1.

.

TABEL I
UKURAN HURUF UNTUK MAKALAH

Data train akan dilatih selama 1000 kali pengulangan atau epoch, dengan setting yang sudah ditetapkan di table 4.1.

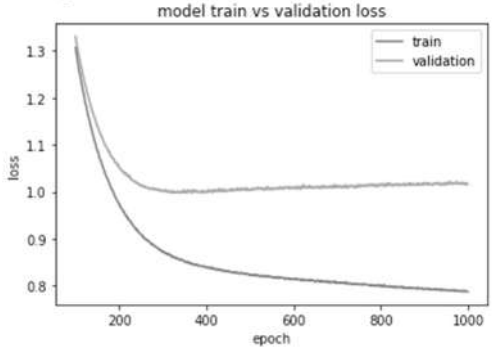
Diberikan input sampel x (baris user-item matriks) umpan yang menghitung output yang dibuat, dengan kata lain input dan output mempunyai jumlah neuron yang sama. Seperti yang sudah dijelaskan di sub-bab sebelumnya, ¬hidden layer menggunakan fungsi aktivasi SELU.

TABEL II
HASIL TRAINING DENGAN 1000 EPOCH

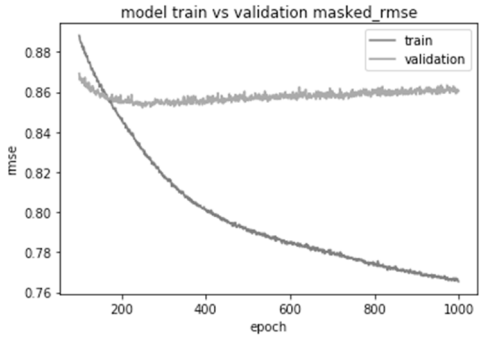
Epoch	loss	RMSE	Validation loss	Validation rmse
1	16.3628	2.8623	14.0006	2.8196
50	1.7846	0.9551	1.7367	0.8990
100	1.2893	0.8873	1.3167	0.8703
150	1.0691	0.8624	1.1215	0.8582
200	0.9569	0.8445	1.0420	0.8574
250	0.8974	0.8301	1.0065	0.8548

300	0.8667	0.8193	0.9982	0.8572
350	0.8476	0.8099	0.9959	0.8570
400	0.8361	0.8032	0.9950	0.8549
450	0.8288	0.7980	0.9978	0.8568
500	0.8224	0.7928	1.0022	0.8581
550	0.8176	0.7895	1.0019	0.8566
600	0.8139	0.7866	1.0051	0.8579
650	0.8098	0.7839	1.0082	0.8599
700	0.8065	0.7818	1.0072	0.8585
750	0.8034	0.7796	1.0113	0.8605
800	0.8011	0.7779	1.0093	0.8588
850	0.7990	0.7768	1.0110	0.8600
900	0.7959	0.7749	1.0146	0.8625
950	0.7924	0.7724	1.0137	0.8618
1000	0.7907	0.7715	1.0114	0.8599

Pada table 2 terlihat hasil loss dan rmse yang didapatkan dari pengu-langan atau epoch 1000 kali. Hasil loss yang didapatkan dari 1000 epoch adalah 0.7907 dengan validasi 1.0114 , dan hasil RMSE yang didapatkan adalah 0.7715 dengan validasi 0.8599. Hasil training akan divisualisasikan dalam bentuk grafik.



Gambar. 7 Grafik loss hasil training



Gambar. 8 Grafik rmse hasil training

Dari gambar 7 dan 8 grafik menunjukan gap yang besar antara training dan validasi, fenomena ini menunjukan bahwa model hasil training akan mudah overfit. Penulis melakukan eksperimen lain untuk mengurangi gap antara hasil train dan validasi dengan cara menambahkan Gaussian Noise pada input:

TABEL III
HASIL TRAINING 1000 EPOCH DENGAN MENAMBAHKAN GAUSSIAN NOISE

Epoch	loss	RMSE	Validation loss	Validation rmse
1	16.5052	2.8569	14.6402	2.8347
50	1.9365	0.9661	1.8718	0.9051
100	1.3299	0.8755	1.3712	0.8652
150	1.0546	0.8404	1.1407	0.8552
200	0.9177	0.8117	1.0533	0.8529
250	0.8458	0.7866	1.0301	0.8561
300	0.8082	0.7677	1.0285	0.8589
350	0.7882	0.7558	1.0347	0.8622
400	0.7764	0.7480	1.0377	0.8625
450	0.7686	0.7429	1.0415	0.8640
500	0.7615	0.7379	1.0426	0.8652
550	0.7587	0.7363	1.0443	0.8658
600	0.7535	0.7330	1.0436	0.8655
650	0.7502	0.7314	1.0459	0.8662
700	0.7493	0.7312	1.0462	0.8680
750	0.7438	0.7279	1.0435	0.8663
800	0.7422	0.7268	1.0450	0.8673
850	0.7388	0.7256	1.0458	0.8678
900	0.7367	0.7246	1.0447	0.8679
950	0.7342	0.7234	1.0447	0.8679
1000	0.7322	0.7227	1.0432	0.8686

Hasil yang didapatkan dari training ini yaitu loss sebesar 0.7322 dengan validasi 1.0432 dan rmse sebesar 0.7227 dengan validasi 0.8686. Meskipun tidak berbeda jauh dengan eksperimen sebelumnya namun hasil akurasi cukup baik, hal ini dibuktikan dengan nilai loss dan rmse yang lebih rendah.

G. Testing Model

Setelah semua hasil training selesai maka model akan di save dalam format file json sedangkan hasil weights akan disimpan ke dalam format hdf5.

```
def save_model(name, model):

  # # Mengubah model ke format json

  model_json = model.to_json()

  with open("{}.json".format(name), "w") as
json_file:

      json_file.write(model_json)

  # mengubah weights ke format hdf5

  model.save_weights("{}.h5".format(name))

  print("Model berhasil disimpan!")
```

Pseudo code untuk menyimpan hasil training

1. Load model
2. Mengisi rating awal
3. Inisialisasi dan mengubah menjadi bentuk matriks
4. Melakukan prediksi dengan model
5. Output prediksi

Gambar. 9 Data keluaran hasil testing

Langkah pertama yaitu load model yang mempunyai hasil rmse dan loss paling rendah dari hasil training sebelumnya. Lalu setelah model telah di masukan, maka rating awal akan diisi terlebih dahulu. Hal ini dilakukan untuk memberi rekomendasi awal yang sesuai dengan history pengguna. Langkah selanjutnya yaitu inisialisasi dataframe menjadi bentuk matriks agar masukan dapat diprediksi oleh model yang telah dibuat dari hasil training. Setelah semua langkah tersebut dilakukan maka akan keluar keluaran seperti pada gambar 10.

	movieid	rating	title
2761	2761	4.491282	Iron Giant, The (1999)
2202	2202	4.338063	Lifeboat (1944)
1147	1147	4.332850	When We Were Kings (1996)
1222	1222	4.326926	Full Metal Jacket (1987)
3469	3469	4.321336	Inherit the Wind (1960)
1241	1241	4.312613	Braindead (1992)
122	122	4.303579	Boomerang (1992)
3818	3818	4.286380	Pot O' Gold (1941)
1268	1268	4.271552	Pump Up the Volume (1990)
1302	1302	4.267405	Field of Dreams (1989)

Gambar. 10 Hasil keluaran dari testing

VI. KESIMPULAN DAN SARAN

Berikut kesimpulan yang didapatkan berdasarkan penelitian yang telah dilakukan:

- Dari penelitian yang telah dilakukan, dapat disimpulkan bahwa sistem rekomendasi film dapat dibangun dengan menggunakan metode Autoencoder dan dapat menghasilkan tingkat akurasi yang baik. Hal ini dibuktikan dengan nilai loss dan RMSE yang cukup rendah.
- Berdasarkan hasil pengujian dengan data uji 6040 user, 3883 item (film), dan 1.000.209 rating menghasilkan nilai loss yang rendah yaitu 0.7322 dan nilai RMSE 0,7227.Mengefisiensikan kode program teknik LSB modifikasi agar lebih cepat dalam proses komputasi.

Adapun saran yang ingin penulis berikan setelah melakukan penelitian ini:

- Pada penelitian ini penulis menggunakan metode Autoencoder untuk rekomendasi, untuk penelitian selanjutnya dapat dikembangkan lagi dengan menggunakan metode dan bidang lainnya agar dapat menghasilkan rekomendasi yang lebih baik lagi.
- Untuk penelitian selanjutnya evaluasi rekomendasi bisa dicoba dengan cara lain selain dengan evaluasi metrik agar dapat melihat tingkat akurasi dari sisi lain, misalnya dengan feedback dari user

REFERENSI

[1] C. A. Gomez-uribe and N. Hunt, "The Netflix Recommender System : Algorithms , Business Value ," vol. 6, no. 4, 2015.

[2] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," 2008.

[3] F. Ricci, L. Rokach, and B. Shapira, "Introduction to Recommender Systems Handbook," in *Recommender Systems Handbook*, 2011.

[4] M. M. Recker and A. Walker, "Supporting ' Word-of-Mouth ' Social Networks through Collaborative Information Filtering," *J. Interact. Learn. Res.*, 2003.

[5] B. Rocca, "Introduction to recommender systems," *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.

[6] Y. Lecun, Y. Bengio, and G. Hinton, "Nature Deep Review," *Nature*, 2015.

[7] N. Wang and D.-Y. Yeung, *Learning a Deep Compact Image Representation for Visual Tracking*. 2013.

[8] C. Y. Liou, W. C. Cheng, J. W. Liou, and D. R. Liou, "Autoencoder for words," *Neurocomputing*, 2014.