OPTIMASI MODEL SISTEM REKOMENDASI FILM DENGAN *NEURAL NETWORK* (STUDI KASUS: PLATFORM LETTERBOXD)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Strata-1
Program Studi Teknik Industri - Fakultas Teknologi Industri
Universitas Islam Indonesia

Nama	: Safiella Citra Aishwvarya
No. Mahasiswa	: 18522346

PROGRAM STUDI TEKNIK INDUSTRI
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2022

PERNYATAAN KEASLIAN

Demi Allah, saya akui karya ini adalah hasil kerja saya sendiri kecuali nukilan dan ringkasan yang setiap satunya telah saya jelaskan sumbernya. Jika di kemudian hari ternyata terbukti pengakuan saya ini tidak benar dan melanggar peraturan yang sah dalam karya tulis dan hak kekayaan intelektual, maka saya bersedia ijazah yang telah saya terima untuk ditarik kembali oleh Universitas Islam Indonesia.

Yogyakarta, 28 November 2022

(Safiella Citra Aishwvarya)
18522346

SURAT BUKTI PENELITIAN

UNIVERSITAS ISLAM INDONESIA

FAKULTAS TEKNOLOGI INDUSTRI
Gedung KH. Mas Mansur
Kampus Terpadu Universitas Islam Indonesia
Jl. Kaliurang km 14,5 Yogyakarta 55584
T. (0274) 898444 ext. 4100, 4101
F. (0274) 895007
E. fti@uii.ac.id
W. fti.uii.ac.id

SURAT KETERANGAN PENELITIAN

Nomor : 246/Ka.Lab.Datmin/70/Lab.Datmin/I/2023

Assalamu'alaikum Warahmatullahi Wabarakatuh

Kami yang bertanda tangan dibawah ini, menerangkan bahwa mahasiswa dengan keterangan sebagai berikut :

Nama	:	Safiella Citra Aishwvarya
No. Mhs	:	18522346
Dosen Pembimbing	:	Ir. Andrie Pasca Hendradewa, S.T., M.T., IPM.

Telah selesai melaksanakan penelitian yang berjudul " Optimasi Model Sistem Rekomendasi Film dengan Neural Network (Studi Kasus: Platform Letterboxd)" di Laboratorium Data Mining, Program Studi Teknik Industri, Fakultas Teknologi Industri, Universitas Islam Indonesia tercatat mulai tanggal 1 September sampai dengan tanggal 30 September 2022

Demikian surat keterangan kami keluarkan, agar dapat dipergunakan sebagaimana mestinya.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 18 Januari 2023

Kepala Laboratorium
Data Mining

Annisa Uswatun Khasanah, ST., M.B.A., M.Sc.

LEMBAR PENGESAHAN PEMBIMBING

OPTIMASI MODEL SISTEM REKOMENDASI FILM DENGAN *NEURAL NETWORK*
(STUDI KASUS: PLATFORM LETTERBOXD)

TUGAS AKHIR

Disusun Oleh:

Nama : Safiella Citra Aishwvarya
No. Mahasiswa : 18522346

Yogyakarta, 28 November 2022
Dosen Pembimbing

(Andrie Pasca Hendradewa, S.T., M.T., IPM)

LEMBAR PENGESAHAN DOSEN PENGUJI

OPTIMASI MODEL SISTEM REKOMENDASI FILM DENGAN *NEURAL NETWORK*
(STUDI KASUS: PLATFORM LETTERBOXD)

TUGAS AKHIR

Disusun Oleh:

Nama : Safiella Citra Aishwvarya
No. Mahasiswa : 18522346

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata-1 Teknik Industri Fakultas Tekonologi Industri Universitas Islam Indonesia

Yogyakarta, 4 Februari 2023

Tim Penguji

Andrie Pasca Hendradewa, S.T., M.T., IPM
Ketua

Yuli Agusti Rochman, S.T., M.Eng.
Anggota I

Ir. Muchamad Sugarindra, S.T., M.T.I., IPM
Anggota II

Mengetahui
Ketua Program Studi Teknik Industri
Fakultas Teknologi Industri
Universitas Islam Indonesia

Ir. Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D., IPM

HALAMAN PERSEMBAHAN

Penelitian ini saya persembahkan untuk agama dan bangsa dengan harapan membawa manfaat, kedua orang tua dan adik saya, serta teman-teman yang telah menemani saya selama menjalani masa akademik di Teknik Industri Universitas Islam Indonesia.

MOTO

"*Maka sesungguhnya beserta kesulitan ada kemudahan,*
sesungguhnya beserta kesulitan itu ada kemudahan."
(QS. Al-Insyirah: 5-6)

"*You flare, you flicker, you fade,*
and in the end, all your tomorrows become yesterdays."
—Afterlight

"*My biggest fear is being buried six feet deep when I'm nowhere near being the Muslim I want to become. May Allah show his mercy upon us, and grant us light in the grave.*"

KATA PENGANTAR

Assalamu'alaikum warahmatullahi wabarakatuh

Segala puji dan syukur saya haturkan kepada Allah Swt. atas berkat dan hidayah-Nya sehingga saya dapat menyelesaikan Tugas Akhir berjudul "**Optimasi Model Sistem Rekomendasi Film dengan *Neural Network* (Studi Kasus: Platform Letterboxd)**" ini. Tidak lupa salawat dan salam senantiasa saya panjatkan kepada nabi besar kita Nabi Muhammad saw. beserta keluarga, sahabat, serta para pengikutnya yang telah berjuang dan membimbing kita keluar dari kegelapan menuju jalan terang benderang untuk menggapai rida Allah Swt.

Tugas Akhir merupakan salah satu persyaratan yang harus dipenuhi untuk menyelesaikan program studi S-1 dan memperoleh gelar Sarjana Strata-1 pada Jurusan Teknik Industri Fakultas Teknologi Industri Universitas Islam Indonesia. Dalam mengerjakan Tugas Akhir ini, saya banyak mendapatkan bantuan dan dukungan dari berbagai pihak. Dengan segala kerendahan hati, saya ingin mengucapkan terima kasih kepada:

1. Bapak Prof. Dr. Ir. Hari Purnomo, M.T., IPU., ASEAN.Eng. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D., IPM. selaku Ketua Program Studi Teknik Industri Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Andrie Pascra Hendradewa, S.T., M.T., IPM. selaku dosen pembimbing Tugas Akhir yang telah memberikan bimbingan, arahan, serta waktunya dalam penyusunan laporan Tugas Akhir.
4. Kedua orang tua yang telah mendoakan, memberikan dukungan materi, motivasi, dan nasihat selama proses pengerjaan Tugas Akhir.
5. Adik saya Fierlansyah Krishna Desfarro yang telah menyemangati saya selama pengerjaan Tugas Akhir.
6. Rekan-rekan seperjuangan Tugas Akhir Rahmi Meliyandini, Gita Febriani, Evanayeda Anindita, Rizqia Putri Kinasih, Alifia Salsabila Putri, dan Delpi Rahmadani yang saling berbagi bantuan dan suka duka dalam menyelesaikan Tugas Akhir.
7. Laboratorium Data Mining (DATMIN) Teknik Industri Universitas Islam Indonesia tempat penulis mengerjakan Tugas Akhir.

Meskipun laporan Tugas Akhir ini sudah berusaha disusun sebaik mungkin, laporan ini masih tetap memiliki kekurangan di beberapa bagian. Maka dari itu, besar harapan saya agar ke depannya dapat menjadi lebih baik lagi melalui saran dan kritik yang membangun. Semoga laporan Tugas Akhir ini dapat bermanfaat bagi pembaca. *Aamiin*.

Mohon maaf jika ada salah kata dalam laporan ini. Terima kasih atas perhatian pembaca. *Wassalamu'alaikum warahmatullahi wabarakatuh*.

Yogyakarta, November 2022

Safiella Citra Aishwvarya

ABSTRAK

Perkembangan internet yang pesat diiringi banyaknya penggunaan internet membuat pengguna menghadapi masalah kelebihan informasi. Sistem rekomendasi pun hadir sebagai alat penyaringan informasi yang menyajikan item kepada pengguna berdasarkan preferensi dan perilaku pengguna. Salah satu penerapan sistem rekomendasi dalam dunia hiburan adalah sistem rekomendasi pada film untuk ditonton. Letterboxd merupakan sebuah platform kecil yang masih berkembang untuk komunitas yang sangat gemar terhadap film. Sebagai platform media sosial yang berfokus pada film, Letterboxd belum memberikan fasilitas rekomendasi kepada pengguna berdasarkan preferensi pengguna. Sampai saat ini, penelitian tentang penggunaan *optimizer* yang optimal untuk digunakan dalam sistem rekomendasi juga belum terpapar dalam literatur terkait. Untuk itu, dilakukan penelitian yang bertujuan untuk membangun algoritma model dan menemukan *optimizer* terbaik dalam model sistem rekomendasi dengan metode *item-based collaborative filtering* berbasis *neural network* (jaringan saraf). Penelitian ini membangun algoritma model sistem rekomendasi yang terdiri dari tiga *dense layer* utama dengan jumlah neuron yang diberikan pada masing-masing *layer* adalah 400, 200, dan 100 neuron. Penelitian ini membagi sesi *training* ke dalam 5 skenario berdasarkan *optimizer* yang digunakan. Hasil penelitian menunjukkan bahwa model dengan *optimizer* SGD menunjukkan model dengan kinerja terbaik dengan nilai *loss* prediksi 2,1742.

Kata kunci: *deep learning*, kinerja optimal, *neural network*, *optimizer*, sistem rekomendasi

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar belakang

Di era modern saat ini, teknologi internet berkembang sangat cepat diiringi produksi data yang terus meningkat. Laporan yang dikeluarkan oleh Statista.com memaparkan bahwa jumlah data yang dihasilkan dari penggunaan internet tercatat sebesar 79 *zettabyte* (setara dengan 1 miliar *terabyte* atau 1 triliun *gigabyte* dan diperkirakan akan terus meningkat hingga mencapai 181 *zettabyte* pada tahun 2025 (Statista, 2021). Peningkatan ini membuat pengguna menghadapi masalah kelebihan informasi berupa banyaknya jumlah dan variasi item untuk dipilih. Para peneliti pun menyadari akan adanya kebutuhan untuk memiliki sistem yang dapat menyeleksi item sesuai kebutuhan pengguna dan merekomendasikannya kepada pengguna dalam waktu sesingkat mungkin sehingga sistem rekomendasi pun diperkenalkan (Sharma & Singh 2016; Nozari & Koohi 2021).

Di bidang hiburan, terutama film, sistem rekomendasi telah mencapai kesuksesan besar dan diterapkan pada berbagai *platform* yang berkaitan dengan film seperti MovieLens.com, TiVo, bahkan *platform* besar penyedia jasa *streaming film* seperti Netflix. Letterboxd adalah sebuah *platform* media sosial kecil yang masih berkembang untuk komunitas yang sangat gemar terhadap film, memungkinkan penggunanya untuk saling berinteraksi dan berdiskusi mengenai film yang mereka ulas, membuatnya dikatakan sebagai sebuah situs media sosial sederhana yang diposisikan sebagai surga bagi komunitas kritikus film. Meski begitu, *platform* ini tidak pernah mengalami lonjakan keanggotaan atau bertambahnya pengguna sejak perilisannya. Kebanyakan orang menemukannya sendiri atau direkomendasikan dari seorang teman, dan pertumbuhannya yang menjadi 2,5 juta pengguna (1 juta di antaranya aktif) selama delapan tahun terakhir berjalan lambat (Tobias, 2020). Saat mengumpulkan survei dari pengguna Letterboxd, pemilik akun bernama Cineflect memperoleh respons dari pengguna yang salah satunya adalah pengguna menginginkan adanya fitur rekomendasi film yang memiliki algoritma yang baik sesuai dengan preferensi pengguna. Sampai saat ini, Letterboxd memang tidak

memiliki fitur sistem rekomendasi dengan algoritma yang didasarkan pada preferensi pengguna (Cineflect, 2022).

Maka dari itu, dilakukan penelitian yang bertujuan untuk membuat model sistem rekomendasi film yang didasarkan pada preferensi pengguna. Ehrlich & Ma (2017) dalam penelitiannya membandingkan tiga metode sistem rekomendasi menggunakan data Letterboxd, yaitu Naïve Bayes, *Matrix Factorization*, dan *K-Nearest Neighbor* (KNN) dan mendapatkan nilai RMSE terendah pada metode KNN sebesar 1,29137. Sementara itu, penelitian yang dilakukan oleh Pandey et. al. (2019) dan Aljunid & D (2020) menunjukkan bahwa metode *Neural Network* mengungguli metode KNN dan *Matrix Factorization* sebelumnya. Dengan demikian, penelitian ini akan membuat model sistem rekomendasi dengan metode *Neural Network*.

Di satu sisi, *optimizer* pada sebuah model *neural network* merupakan salah satu parameter penting untuk meningkatkan akurasi prediksi model (Desai, 2020). Sampai saat ini, penelitian tentang penggunaan *optimizer* terbaik untuk digunakan dalam sistem rekomendasi belum terpapar dalam literatur yang ada. Pada penelitian yang dilakukan oleh Pandey et al. (2019) dan Aljunid & D (2020), peneliti menggunakan *optimizer* Adam dalam membangun sistem rekomendasi. Sementara itu, pada penelitian yang dilakukan oleh He et al. (2017), *optimizer* yang digunakan adalah *Stochastic Gradient Descent* (SGD), dan pada penelitian yang dilakukan oleh Naumov et al. (2019), peneliti menambahkan *optimizer* Adagrad selain *optimizer* SGD. Maka dari itu, penelitian ini akan mencari *optimizer* terbaik untuk digunakan dalam algoritma sistem rekomendasi yang dibangun. Peneliti berharap hasil penelitian ini dapat digunakan untuk mengetahui *optimizer* yang paling optimal untuk digunakan pada model sistem rekomendasi Letterboxd.

1.2. Rumusan masalah

Rumusan masalah pada penelitian ini adalah:

1. Bagaimana algoritma model sistem rekomendasi film yang dibangun menggunakan data *open source* hasil *scraping* dari platform Letterboxd?

2. *Optimizer* apakah yang menghasilkan kinerja optimal pada model sistem rekomendasi film yang dibuat menggunakan data *open source* hasil *scraping* dari platform Letterboxd?

1.3. Tujuan penelitian

Tujuan penelitian ini adalah:

1. Untuk mengetahui algoritma model sistem rekomendasi film yang dibangun menggunakan data *open source* hasil *scraping* dari platform Letterboxd.
2. Untuk mengetahui *optimizer* apa yang menghasilkan kinerja optimal pada model sistem rekomendasi film yang dibuat menggunakan data *open source* hasil *scraping* dari platform Letterboxd.

1.4. Manfaat penelitian

Manfaat penelitian ini adalah:

1. Platform atau sosial media berbasis film dapat menggunakan model hasil penelitian untuk meningkatkan kualitas pelayanan yang diberikan kepada pengguna.
2. Hasil penelitian dapat digunakan sebagai referensi untuk mengembangkan, mengevaluasi, dan memberikan perbaikan pada penelitian selanjutnya.

1.5. Batasan masalah

Batasan masalah yang ada pada penelitian ini berupa:

1. Objek yang dikenai dalam penelitian ini adalah model sistem rekomendasi yang data inputnya berasal dari platform Letterboxd.
2. Input yang digunakan adalah data pengguna, *rating* pengguna, dan judul film. Data genre film hanya digunakan untuk memberi keterangan tambahan pada judul film ketika rekomendasi film ditampilkan, tidak untuk diolah pada saat membangun model.
3. Data diperoleh dari situs Kaggle yang menyimpan *dataset* dari hasil *data scraping* pada platform Letterboxd berisi 4000 data pengguna teraktif yang terakhir diperbarui pada Maret 2022.

1.6. Sistematika penulisan

Sistematika penulisan laporan tugas akhir ini terdiri dari beberapa bab yang akan diuraikan sebagai berikut.

BAB I PENDAHULUAN

Bab ini berisi uraian tentang latar belakang permasalahan penelitian, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan laporan penelitian.

BAB II KAJIAN LITERATUR

Bab ini berisi kajian deduktif yang menjelaskan tentang pembahasan konsep dasar dan teori-teori yang berkaitan dengan penelitian dan kajian induktif mengenai hasil penelitian-penelitian sebelumnya yang berkaitan dengan penelitian yang dilakukan.

BAB III METODE PENELITIAN

Bab ini menjelaskan langkah-langkah penelitian yang dilakukan menggunakan metode penelitian yang telah ditentukan. Bab ini juga memuat objek dan subjek penelitian, jenis dan sumber data, metode pengumpulan data, metode pengolahan data, metode analisis data, dan diagram alir penelitian.

BAB IV PENGUMPULAN DAN PENGOLAHAN DATA

Bab ini menjabarkan proses pengumpulan dan pengolahan data yang dilakukan pada penelitian menggunakan metode yang telah ditentukan dan diuraikan pada bab sebelumnya.

BAB V ANALISIS DAN PEMBAHASAN

Bab ini berisi hasil analisis dari pengolahan data yang dilakukan dan telah dijabarkan pada bab sebelumnya.

BAB VI KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang diambil dari hasil analisis pada bab pembahasan yang menjawab permasalahan penelitian yang sudah dirumuskan sebelumnya. Bab ini juga mencakup saran untuk penelitian selanjutnya.

BAB II
KAJIAN LITERATUR

2.1. Kajian deduktif

Kajian deduktif berisi tentang literatur rujukan yang mendukung penelitian yang dilakukan oleh peneliti. Berikut kajian literatur yang dijadikan acuan sebagai pendukung penelitian ini.

2.1.1. Artificial Intelligence

Kecerdasan buatan (*Artificial Intelligence*, disingkat AI) merupakan salah satu bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti yang dilakukan oleh manusia, bahkan bisa lebih baik daripada yang dilakukan oleh manusia (Dahria, 2008). Beberapa kegiatan yang dirancang untuk dikerjakan oleh kecerdasan buatan adalah pembelajaran, persepsi, pemecahan masalah, pemahaman bahasa, dan/atau penalaran logika. Kecerdasan buatan menjawab pertanyaan penting tentang: pengetahuan apa yang dibutuhkan pada setiap aspek pemikiran kecerdasan buatan; bagaimana seharusnya pengetahuan itu direpresentasikan; dan bagaimana seharusnya pengetahuan itu digunakan (Mohammed, 2019).

2.1.2. Machine Learning

Machine Learning (ML) adalah kategori algoritma AI yang memungkinkan aplikasi perangkat lunak untuk memprediksi respons secara lebih akurat dan spesifik tanpa perlu memprogramnya secara eksplisit. ML berfokus pada pengembangan algoritma yang mampu menerima data input serta menggunakan analisis statistik untuk memprediksi *output* sembari memperbarui *output* dengan data baru (Mohammed, 2019).

ML melakukan pembelajaran dengan cara menemukan beberapa hubungan antara fitur (data input) dan variabel target (data *output*). Pengujian algoritma ML dilakukan dengan satu set *training data* (data pelatihan) dan satu set *testing data* (data uji) yang keduanya merupakan set data yang sudah dipisahkan. Program akan diberikan contoh-contoh dari

data pelatihan, kemudian set data uji akan diberikan kepada program (Budiharto, 2016). Penerapan ML sendiri ada di beberapa bidang, di antaranya (Mohri et. al., 2018):

1. Klasifikasi. Klasifikasi merupakan tipe ML yang digunakan untuk menetapkan kategori pada setiap item. Misalnya, klasifikasi dokumen yang terdiri dari kategori politik, bisnis, olahraga, atau cuaca. Sementara itu, klasifikasi gambar dapat terdiri dari gambar mobil, kereta api, atau pesawat. Jumlah kategori yang dikerjakan oleh metode klasifikasi sering kali berjumlah kurang dari beberapa ratus buah, tetapi bisa jauh lebih besar pada beberapa tugas yang sulit dan bahkan tidak terbatas, seperti pada *Optical Character Recognition*, klasifikasi teks, atau pengenalan suara.
2. Regresi: Regresi merupakan tipe ML yang berfokus pada masalah prediksi nilai real untuk setiap item. Contoh dari regresi adalah prediksi nilai saham atau variasi variabel ekonomi. Dalam regresi, penalti untuk prediksi yang salah tergantung pada besarnya perbedaan antara nilai aktual dengan nilai yang diprediksi.
3. *Ranking*: Tipe ini merupakan tipe permasalahan yang dilakukan dengan melakukan pembelajaran untuk mengurutkan item menurut beberapa kriteria. Contoh dari tipe ini ada pada pencarian situs yang mengembalikan halaman situs yang relevan dengan hasil pengetikan di alat pencarian.
4. *Clustering*: Tipe ML ini mempelajari masalah untuk membagi satu set item menjadi subset yang bersifat homogen atau sama. *Clustering* sering digunakan untuk menganalisis kumpulan data yang sangat besar. Misalnya, dalam konteks analisis jaringan sosial, algoritma pengelompokan yang ada mencoba mengidentifikasi komunitas alami yang terdapat pada sekelompok besar orang yang terdapat pada jaringan sosial tersebut.
5. *Dimensionality reduction or manifold learning*: Tipe ML yang satu ini mengubah representasi awal item menjadi representasi dimensi yang lebih rendah sambil mempertahankan beberapa properti dari representasi awal. Contoh umumnya adalah *preprocessing* gambar digital pada *computer vision*.

ML sendiri memiliki beberapa jenis skenario pembelajaran dalam menjalankan programnya. Skenario ini memiliki perbedaan terkait jenis data pelatihan yang tersedia,

serta urutan dan metode penerimaan data pelatihan dan data uji yang digunakan untuk mengevaluasi algoritma pembelajaran. Dari Mohri et. al. (2018), jenis-jenis skenario pembelajaran yang pada umumnya diketahui, di antaranya:

1. *Supervised learning*: Jenis skenario ini menerima serangkaian contoh yang diberi label sebagai data pelatihan dan membuat prediksi untuk semua item yang tidak terlihat. Jenis skenario ini merupakan skenario yang paling umum. Tipe ML yang masuk pada skenario ini adalah klasifikasi, regresi, dan *ranking*.
2. *Unsupervised learning*: Jenis skenario ini menerima data pelatihan yang tidak diberi label, dan membuat prediksi untuk semua item yang tidak terlihat. *Clustering* dan *dimensionality reduction* adalah contoh dari *unsupervised learning*.
3. *Reinforcement learning*: Pada jenis ini, pengumpulan informasi dilakukan secara aktif melalui interaksi program dengan lingkungan, dan dalam beberapa kasus memengaruhi lingkungan, kemudian menerima hadiah (*reward*) langsung untuk setiap tindakan yang dilakukan program. Objek program pada jenis skenario ini adalah untuk memaksimalkan pemerolehan hadiah selama tindakan dan iterasi dilakukan dalam lingkungan yang dikenai interaksi.

2.1.3. Deep Learning

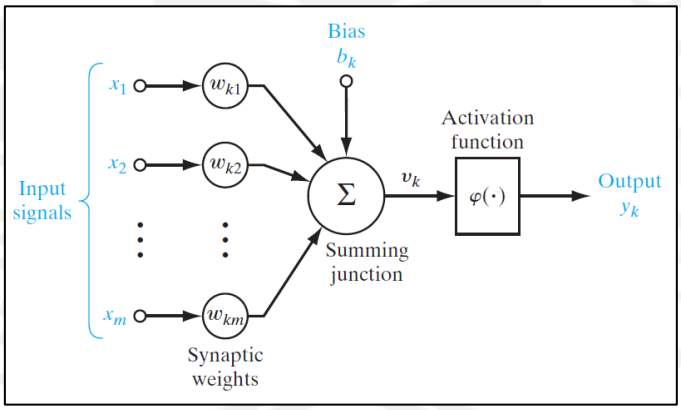
Deep learning adalah bagian dari ML yang berkaitan dengan model pembelajaran yang didasarkan pada jaringan saraf dengan banyak *layer*. Kata ‘deep’ pada *deep learning*, alih-alih merujuk pada segala jenis pemahaman yang lebih dalam yang dilakukan metode ini, kata tersebut mewakili gagasan tentang representasi *layer* yang disusun secara berturutan (Chollet, 2018). Hal yang membedakan *deep learning* dari jenis ML yang lain adalah bahwa operasi yang dilakukan pada masing-masing representasi *layer* yang digunakan dipelajari secara bersamaan.

Masalah pembelajaran dari sinyal audio mentah atau nilai piksel mentah dari sebuah gambar merupakan area keunggulan aplikasi *deep learning*. Model dengan banyak *layer* ini mampu menangani data persepsi tingkat rendah dengan cara yang tidak bisa dilakukan metode tradisional yang sudah ada sebelumnya (Zhang et. al., 2021). Dalam *deep learning*, representasi berlapis ini dipelajari melalui model yang disebut *neural network*

(jaringan saraf) yang disusun secara terstruktur dalam lapisan yang persis ditumpuk di atas satu sama lain.

2.1.4. Neural Network

Neural Network (disebut juga sebagai *Artificial Neural Network* atau Jaringan Saraf Tiruan) merupakan pertemuan *node* atau unit yang saling berhubungan dari elemen pemrosesan sederhana (Gurney, 1997) yang memiliki algoritma yang terinspirasi oleh arsitektur dan dinamika jaringan neuron di otak. Jaringan tersebut dapat melakukan banyak tugas pemrosesan informasi (Mehlig, 2021). *Neural network* memiliki kemampuan pembelajaran yang kuat dan mampu membuat model hubungan nonlinier antara input dan *output*-nya (He et al., 2017). Gambar di bawah menunjukkan model dari sebuah *neural network*.



Gambar 2.1 Model *Neural Network*

Sumber: Haykin, 2009

Rincian dari tiga elemen dasar yang terdapat pada model *neural network* di atas, yaitu (Haykin, 2009):

1. Seperangkat sinapsis yang masing-masing memiliki senilai bobot. Secara khusus, sinyal x_m pada input sinapsis m yang terhubung ke tiap neuron k dikalikan dengan bobot sinapsis w_{km}. Berbeda dengan bobot sinapsis yang terdapat pada otak

manusia, bobot sinapsis sebuah neuron buatan dapat berada pada kisaran nilai negatif dan positif.

2. Sebuah *summing function* untuk menjumlahkan sinyal input yang diterima. Nilai ini diberi pembobotan dari masing-masing neuron.
3. *Activation function* (fungsi aktivasi) untuk membatasi amplitude nilai *output*. Fungsi aktivasi juga disebut sebagai *squashing function*, karena fungsi ini menekan (membatasi) rentang amplitudo sinyal *output* ke beberapa nilai terbatas.

Algoritma pembelajaran yang terjadi pada sebuah neuron sesungguhnya adalah proses mengubah nilai bobot yang terdapat pada neuron tersebut. Model *neural network* pada gambar tersebut juga menyertakan bias sebagai input tambahan yang dilambangkan dengan b_k. Bias b_k memberi efek untuk meningkatkan atau menurunkan input bersih dari fungsi aktivasi yang dijalankan. Dalam istilah matematika, dapat digambarkan neuron *k* pada model di atas dengan persamaan:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

Di mana:

$x_1, x_2, \ldots, x_m$: input

$w_{k1}, w_{k2}, \ldots, w_{km}$: bobot sinapsis dari neuron *k*

u_k : *output* gabungan linier dari input (tidak ditunjukkan pada gambar)

b_k : bias

$\varphi(\cdot)$: fungsi aktivasi

y_k : *output* dari neuron

Penetapan bias b_k memberikan efek transformasi terhadap *output* u_k dari penggabung linier dalam model seperti yang ditunjukkan oleh persamaan berikut ini, sehingga v_k nantinya akan dikenai fungsi aktivasi sebelum *output* final dihasilkan.

$$v_k = u_k + b_k \tag{2.3}$$

Fungsi aktivasi terdiri dari dua jenis, di antaranya (Haykin, 2009):

1. Fungsi *threshold*, yang memiliki rumus:

$$\varphi(v) = \begin{cases} 1 & jika\ v \geq 0 \\ 0 & jika\ v < 0 \end{cases} \tag{2.4}$$

   Sejalan dengan rumus di atas, *output* neuron *k* yang menggunakan fungsi *threshold* tersebut dinyatakan dengan:

$$y_k = \begin{cases} 1 & jika\ v_k \geq 0 \\ 0 & jika\ v_k < 0 \end{cases} \tag{2.5}$$

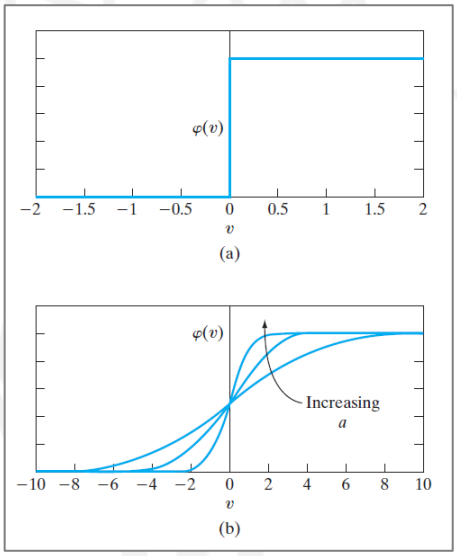
2. Fungsi sigmoid, yang merupakan bentuk paling umum dari fungsi aktivasi yang digunakan dalam membangun jaringan saraf. Fungsi sigmoid merupakan fungsi yang meningkat secara ketat, menunjukkan keseimbangan anggun antara perilaku linier dan nonlinier. Contoh dari fungsi sigmoid adalah fungsi logistik, yang dirumuskan sebagai:

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{2.6}$$

   Di mana *a* adalah parameter kemiringan fungsi sigmoid. Dengan memberi variasi pada parameter *a*, fungsi sigmoid dengan kemiringan yang berbeda dapat diperoleh. Dalam batasnya, ketika parameter kemiringan mendekati tak terhingga, fungsi sigmoid berubah menjadi sekadar fungsi ambang (*threshold*). Ketika fungsi ambang mengasumsikan nilai 0 atau 1, fungsi sigmoid mengasumsikan rentang nilai kontinu dari 0 hingga 1. Maka, fungsi ambang pada fungsi sigmoid dinotasikan sebagai:

$$\varphi(v) = \begin{cases} 1 & jika\ v \geq 0 \\ 0 & jika\ v = 0 \\ -1 & \text{jika } v < 0 \end{cases} \quad (2.7)$$

Berikut merupakan grafik yang menunjukkan perbedaan antara fungsi ambang dan fungsi sigmoid.



Gambar 2.2 Grafik fungsi *threshold* (a) dan grafik fungsi sigmoid (b)

Sumber: Haykin, 2009

2.1.5. Deep Neural Network

Deep Neural Network (DNN) adalah sebutan yang mewakili banyaknya *layer* yang digunakan pada *deep learning*, sehingga dinamakan demikian. DNN dapat melakukan transformasi hierarki secara mendalam dari input data yang diberikan dan memiliki kinerja yang lebih baik dibandingkan dengan *shallow neural network* (Golovko et. al, 2016). Keunggulan dari DNN adalah DNN memiliki kinerja yang tinggi untuk mengatasi *dataset* dalam jumlah yang besar karena karakteristiknya yang memiliki banyak *layer* untuk mengolah data kompleks (Miotto et. al., 2018).

Proses pembelajaran yang dilakukan pada DNN dilakukan dengan memberikan sebuah bobot (*weight*) pada input yang masuk ke dalam *layer*. Pembelajaran yang dilakukan oleh

tiap *layer* pada DNN adalah dengan mencari sekumpulan nilai bobot yang sesuai untuk setiap *layer* sehingga jaringan saraf pada DNN dapat memetakan data input menuju *output* yang diharapkan secara benar. Untuk mengontrol *output* dari jaringan saraf, digunakan pengukuran seberapa jauh *output* yang dihasilkan dari *output* yang diharapkan. Hal ini merupakan tugas dari *loss function*, yang terkadang juga disebut *objective function*. *Loss function* mengambil hasil prediksi dari jaringan saraf dan target yang sebenarnya dan menghitung jarak nilainya untuk mengetahui seberapa baik pembelajaran yang telah dilakukan oleh jaringan tersebut. Kemudian, skor *loss function* digunakan sebagai sinyal umpan balik untuk menyesuaikan nilai bobot untuk menurunkan skor *loss*. Penyesuaian yang dilakukan pada jaringan saraf ini merupakan tugas dari *optimizer* (Chollet, 2018).

Pada DNN, metode *training* yang populer digunakan adalah algoritma *backpropagation*. Langkah-langkah pemrosesan pada algoritma *backpropagation* adalah (Nisbet et. al., 2018):

1. Bobot diberikan secara acak untuk setiap koneksi.
2. Menghitung nilai pada setiap unit sebagai jumlah input dikalikan dengan bobotnya.
3. Menentukan nilai ambang di mana jika *output* berada di atas nilai ambang maka *output* dievaluasi menjadi "1" dan jika di bawahnya dievaluasi menjadi "0".
4. Menghitung kesalahan prediksi dengan rumus:

$$\textit{Error} = \text{nilai prediksi yang diharapkan (nilai aktual)} - \text{nilai prediksi} \quad (2.8)$$

5. Menyesuaikan nilai bobot dengan cara:

$$\textit{Adjustment} = \text{error} \times \text{nilai bobot } \textit{output} \quad (2.9)$$

6. Menghitung nilai bobot baru dengan cara:

$$\text{Nilai bobot baru} = \text{nilai bobot input lama} + \textit{adjustment} \quad (2.10)$$

7. Melakukan hal yang sama untuk semua input.
8. Mengulangi sejumlah iterasi melalui data.

2.1.6. Sistem Rekomendasi

Sistem rekomendasi adalah sebuah program yang berupaya memprediksi sebuah item berdasarkan informasi yang diperoleh dari pengguna (Priyono, 2016). Sistem rekomendasi menjadi sarana untuk membantu pengguna menemukan dan memilih produk, layanan, atau informasi pada situs atau aplikasi tertentu untuk mengurangi kelebihan informasi (Hadi et. al., 2020; Champiri et. al., dalam Li et. al., 2019). Sistem rekomendasi ditujukan bagi individu yang belum memiliki pengalaman untuk menentukan pilihan dari banyaknya alternatif yang ditawarkan (Ritdrix & Wirawan, 2018). Secara umum, metode sistem rekomendasi dapat dibagi menjadi dua jenis, yaitu (Ha & Lee, 2017):

1. *Content-based filtering*. Metode ini merekomendasikan item berdasarkan deskripsi yang terdapat pada item. Misalnya, item musik dapat menggunakan penyanyi, genre, ritme, dan tempo sebagai dasar dalam merekomendasikan item. Jika pengguna memiliki banyak dan beragam item, metode *content-based filtering* dapat bekerja dengan baik. Namun, jika item tidak mencukupi atau bias untuk jenis tertentu, kinerjanya bisa buruk. Kelebihan dari metode ini adalah metode ini tidak bergantung pada preferensi pengguna lain. Kelemahan metode ini adalah pengguna dengan jumlah pembelian yang banyak atau mempunyai selera yang bervariasi membuat metode ini sedikit lebih sulit untuk membuat rekomendasi yang valid.
2. *Collaborative filtering*. Metode ini berfokus pada hubungan pengguna-ke-pengguna atau item-ke-item. *Collaborative filtering* dibagi lagi menjadi dua pendekatan, yaitu:
    a. *User-based*. *User-based collaborative filtering* mengasumsikan bahwa pengguna serupa akan lebih menyukai item serupa. Metode ini merekomendasikan item kepada pengguna karena pengguna lain yang mirip dengannya menyukai item tersebut.

b. *Item-based*. *Item-based collaborative filtering* mengasumsikan bahwa pengguna akan lebih memilih item yang serupa dengan yang telah mereka pilih. Metode tersebut mengelompokkan item serupa berdasarkan riwayat item yang disukai pengguna, kemudian merekomendasikannya kepada pengguna. Keunggulan dari metode ini adalah metode ini memberikan hasil rekomendasi yang lebih baik ketika jumlah pengguna dan *ratings* tersedia dalam jumlah yang melimpah (Sharma & Singh, 2016).

2.1.6.1. Item-based Collaborative Filtering

Item-based collaborative filtering merupakan teknik *collaborative filtering* yang paling banyak dikembangkan hingga saat ini. Pendekatan metode ini dapat dilihat dari ketika dua item a dan b memiliki pengguna serupa yang menyukai dan tidak menyukainya, maka kedua item tersebut dianggap serupa dan pengguna dianggap memiliki minat yang sama untuk item serupa (Sharma & Singh, 2016). Model ini digambarkan dalam bentuk matriks $i \times j$ di mana i adalah pengguna dan j adalah item, sehingga matriks *rating* dirumuskan sebagai $R = [r_{ij}]$ di mana r_{ij} merupakan *rating* pengguna i terhadap item j (Aggarwal, 2016).

Untuk memprediksi *rating* yang akan diberikan pengguna, maka notasi yang digunakan adalah $\hat{r}_{ij}$ di mana simbol "^" menandakan estimasi, sehingga $\hat{r}_{ij}$ adalah estimasi *rating* yang diberikan pengguna i terhadap item j. $\hat{r}_{ij}$ dirumuskan sebagai (Lazy Programmer Inc., 2018):

$$\hat{r}_{ij} = \frac{\sum_{j \in \Omega_i} r_{ij}}{|\Omega_i|} \tag{2.11}$$

di mana:

i = 1 … N, N = jumlah pengguna

j = 1 … M, M = jumlah item

Ω_i = kumpulan semua item yang diberi *rating* oleh pengguna i

r_{ij} = *rating* yang diberikan pengguna i terhadap item j

2.1.7. Matrix Factorization

Matrix Factorization (MF) merupakan algoritma yang digunakan untuk menemukan faktor laten (tersembunyi) sebagai korespondensi antara item dan pengguna yang disimpulkan dari pola *rating* item (Koren et al., 2009). Misalkan p_i dan q_j masing-masing menunjukkan *hidden vector* untuk pengguna i dan item j, maka estimasi interaksi $\hat{r}_{ij}$ sebagai *inner product* dari p_i dan q_j adalah sebagai berikut (He et al., 2017):

$$\hat{r}_{ij} = {p_i}^T q_j = \sum_{k=1}^{K} p_{ik} q_{jk} \tag{2.12}$$

K menunjukkan dimensi ruang laten yang terdapat pada vektor laten antara pengguna i dan item j. MF memodelkan interaksi dua arah dari *hidden factors* yang terdapat pada pengguna dan item, dengan asumsi bahwa setiap dimensi ruang laten bersifat independen satu sama lain dan menggabungkannya secara linier dengan bobot yang sama. Dengan demikian, MF dapat dianggap sebagai model linier dari *hidden factor* yang ada antara item dan pengguna.

2.1.8. Letterboxd

Letterboxd adalah sebuah *platform* media sosial kecil yang masih berkembang untuk komunitas yang sangat gemar terhadap film. Tidak seperti situs basis data film lain, seperti IMDb, *platform* ini menampilkan profil pengguna atau pemberi *rating* dalam gaya buku harian (*diary-style*). Artinya, pemilik akun dan pengguna lain dapat melihat daftar catatan film yang sudah ditonton berdasarkan tanggal yang disematkan pada profil seorang pengguna (Ehrlich & Ma, 2017).

2.2. Kajian induktif

Kajian induktif berisi tentang kajian mengenai penelitian terdahulu. Dalam melakukan penelitian ini, penulis tidak terlepas dari acuan yang ada pada penelitian terdahulu baik dari pembahasan topik, penggunaan metode, dan permasalahan yang diusung.

Penelitian pertama oleh Ortega et. al. (2016) mengusung tiga jenis pendekatan dengan *matrix factorization* dalam membangun sistem rekomendasi dengan metode *collaborative filtering*., yaitu dengan: (1) *After Factorization* (AF); (2) *Before Factorization* (BF); dan (3) *Weighted* BF. *Evaluation metrics* yang digunakan untuk mengukur prediksi sistem rekomendasi yang dibuat adalah *precision* dan *recall*. Penelitian ini juga menggunakan dua set data, yaitu set data dari MovieLens dan Netflix. Skenario evaluasi juga dibagi lagi ke dalam tiga grup, yaitu grup kecil (2–4 pengguna), grup sedang (5–8 pengguna), dan grup besar (9–12 pengguna). Hasil penelitian menunjukkan bahwa MovieLens mendapatkan rekomendasi yang lebih baik saat AF digunakan dan Netflix mendapatkan rekomendasi yang lebih baik saat BF atau WBF digunakan untuk grup kecil. Untuk grup besar, BF menjadi pendekatan rekomendasi terbaik di antara semua skenario. Metode yang diusung juga dibandingkan dengan metode *K-Nearest Neighbor* (KNN) dari penelitian yang sudah dilakukan sebelumnya. Perbandingan dari kedua metode tersebut (dengan metode MF disertai keseluruhan tiga skenario cakupan grup) menunjukkan bahwa kualitas rekomendasi meningkat pada metode MF, yaitu sekitar 125% hingga 150% lebih baik dibanding KNN pada setiap skenario.

Pada penelitian kedua oleh He et al. (2017), penelitian ini mengusung tiga jenis model, yaitu model sistem rekomendasi yang dibangun dengan *Multilayer Perceptron* (MLP), *Generalized Matrix Factorization* (GMF), dan penggabungan keduanya yang disebut sebagai *Neural Matrix Factorization* (NeuMF). Model yang dibangun dengan MLP dan GMF dijalankan dengan *optimizer* Adam, sedangkan model NeuMF dijalankan dengan *optimizer* SGD. Pada perbandingan performansi model yang dibangun, model yang diusung mencapai performa terbaik pada kedua kumpulan data yang diuji (MovieLens dan Pinterest), dan secara signifikan mengungguli metode canggih eALS dan BPR dari penelitian terdahulu dengan selisih yang besar. Ini menunjukkan ekspresi tinggi dari NeuMF dengan menggabungkan model linier MF dan nonlinier MLP. Dua metode lainnya, GMF dan MLP, juga menunjukkan performa yang cukup kuat.

Pada penelitian ketiga oleh Lian et. al. (2018), peneliti mengusulkan model sistem rekomendasi yang dinamakan *Compressed Interaction Network* (CIN), yang bertujuan

untuk menghasilkan interaksi fitur secara eksplisit dan pada tingkat vektor. Model ini dibangun menggunakan *Convolutional Neural Netowrk* (CNN) dan *Recurrent Neural Network* (RNN). Peneliti menggabungkan model CIN dan DNN klasik menjadi satu model terpadu, dan menamakan model baru ini eXtreme Deep Factorization Machine (xDeepFM). Hasil penelitian menunjukkan bahwa xDeepFM mengungguli model sistem rekomendasi dari yang sudah ada.

Pada penelitian keempat oleh Pandey et. al. (2019), peneliti menggunakan beberapa metode dari metode tradisional hingga model NeuMF seperti yang dibangun pada penelitian kedua, hanya saja pada penelitian ini data yang digunakan merupakan *explicit feedback* berupa *rating* pengguna, tidak seperti penelitian kedua yang menggunakan *implicit feedback*, yaitu apakah terjadi interaksi antara pengguna dan item. Pada metode *collaborative filtering* tradisional (KNN), nilai *Mean Absolute Error* (MAE) yang diperoleh adalah 1,3. Pada metode *Matrix Factorization* dengan *Neural Networks*, nilai MAE yang diperoleh adalah 0,938. Pada metode *collaborative filtering* menggunakan FastAI, nilai MAE yang diperoleh adalah 0,843. Pada metode *collaborative filtering* dengan *Neural Networks* dengan FastAI, nilai MAE yang diperoleh adalah 0,824. Pada metode NCF yang diusung pada penelitian sebelumnya (NeuMF), nilai MAE yang diperoleh adalah 0,818.

Pada penelitian kelima oleh Naumov et. al. (2019), penelitian ini mengusung model yang disebut dengan *Deep Learning Recommendation Model* (DLRM). DLRM terdiri dari MLP bawah untuk memproses fitur padat yang terdiri dari tiga *hidden layer* dengan masing-masing neuron berjumlah 512, 256 dan 64, dan MLP atas yang terdiri dari dua *hidden layer* dengan jumlah neuron 512 dan 256. Model ini dijalankan dengan dua skenario *optimizer*, yaitu SGD dan Adagrad. Hasil penelitian menunjukkan bahwa kedua model menunjukkan performa yang baik dengan model yang dijalankan dengan *optimizer* Adagrad sedikit lebih baik dibanding model yang dijalankan dengan SGD.

Pada penelitian Aljunid & D (2020), *evaluation metric* yang digunakan adalah RMSE. *Dataset* dibagi secara acak menjadi 80% data pelatihan dan 20% sisanya sebagai data uji. Untuk mencegah *overfitting*, *dropout layer* telah ditambahkan ke input dan *hidden layer* dari model yang diusulkan. Model dijalankan dengan 100 epochs, dengan *batch size* 64.

Peneliti telah menggabungkan *output layer* film dan pengguna menjadi satu *layer* diikuti oleh tiga *hidden layer* yang terhubung sepenuhnya dengan 150 neuron untuk *layer* pertama, 100 untuk *layer* kedua, dan 50 untuk *layer* ketiga. Algoritma *optimizer* Adam digunakan. Metode ini terbukti mengungguli metode-metode tradisional yang digunakan pada penelitian sebelumnya, juga mengungguli metode SVD dan MF.

Pada penelitian oleh Erlangga & Sutrisno (2020), penelitian dilakukan dengan tujuan membangun aplikasi toko kecantikan *online* yang dapat memberikan rekomendasi toko kecantikan di Bandar Lampung berdasarkan *rating* yang ada pada toko kecantikan tersebut. Aplikasi dibangun dengan sistem rekomendasi menggunakan metode *Collaborative Filtering* dengan pendekatan *adjusted cosine similarity*. Uji kelayakan perangkat lunak yang dibangun menunjukkan bahwa aplikasi toko kecantikan *online* yang dibangun berhasil diuji dengan teknik angket skala Gutman dengan hasil yang baik.

Penelitian yang dilakukan oleh Jaja et. al. (2020) menunjukkan hasil model berupa sistem rekomendasi film yang dibangun dengan menggunakan dataset berasal dari MovieLens. Sistem rekomendasi dibangun dengan metode *Item-based Collaborative Filtering*. Perhitungan kemiripan antaritem dilakukan dengan bantuan recommenderlab danodel dibuat dengan program R menggunakan dataset MovieLens yang berisi 100.000 *rating* dari 943 pengguna pada 1664 flm.

Penelitian selanjutnya yang dilakukan oleh Tyas et. al. (2021) menggunakan *rule-based sentiment analysis* untuk proses analisis ulasan pengguna dari halaman Tripadvisor terdapat yang pernah mengunjungi tempat wisata di kota Bandung. Proses analisis ini dilakukan menggunakan bantuan vaderSentiment untuk menimbang nilai positif dan negatif. Nilai positif dikurangi dari nilai negatif untuk mendapatkan nilai majemuk dan diubah menjadi nilai *rating*. Nilai *rating* yang diperoleh kemudian diolah dengan menggunakan metode *Cosine Similarity* dan *Singular Value Decomposition* (SVD) untuk mendapatkan rekomendasi tempat wisata di Kota Bandung. *Root Mean Square Error* digunakan sebagai ukuran tingkat akurasi antar nilai yang diprediksi. Hasil pengukuran tingkat akurasi menghasilkan nilai sebesar 3,489 pada metode *Cosine Similarity*, sedangkan metode *Singular Value Decomposition* mendapatkan nilai sebesar 1,231. Nilai

pada metode *Singular Value Decomposition* lebih kecil dibandingkan dengan metode *Cosine Similarity* dengan selisih sebesar 2,258.

Pada penelitian yang dilakukan oleh Setiaji et. al. (2022), sistem rekomendasi dibuat menggunakan metode K-*Nearest Neighbor* (KKN). Metode perhitungan pada penelitian ini menggunakan 3 perhitungan KNN dan menggunakan konsep perhitungan gabungan untuk mencari hasil rekomendasi yang maksimal. Hasil dari sistem rekomendasi dengan menggunakan metode K-*Nearest Neighbor* berupa *review* yang menyatakan setuju atau tidak setujunya pengguna. Dalam studi saat ini, ada 100 ulasan dari pengguna, dan memiliki persentase 78% untuk sukses dan 22% gagal.

Dari perbandingan penelitian terdahulu di atas, maka peneliti mengusulkan untuk membuat sistem rekomendasi berbasis *neural network* atau *deep learning* karena sistem rekomendasi yang dibangun dengan *deep learning* terbukti mengungguli sistem rekomendasi tradisional yang sudah ada sebelumnya. Berikut ini merupakan tabel yang berisi ringkasan penelitian-penelitian terdahulu yang serupa dengan penelitian yang akan dibuat dan menjadi dasar kajian induktif peneliti.

Tabel 2.1 Kajian induktif

No.	Judul Penelitian	Metode	Hasil
1.	*Recommending items to group of users using Matrix Factorization based Collaborative Filtering* (Ortega et. al., 2016)	*Collaborative Filtering* dengan *Matrix Factorization*	Hasil penelitian menunjukkan bahwa MovieLens mendapatkan rekomendasi yang lebih baik saat AF digunakan dan Netflix mendapatkan rekomendasi yang lebih baik saat BF atau WBF digunakan untuk grup kecil. Untuk kelompok sedang, pendekatan BF dan WBF sangat unggul dibandingkan AF.

No.	Judul Penelitian	Metode	Hasil
			Untuk grup besar, BF menjadi pendekatan rekomendasi terbaik di antara semua skenario. Pendekatan BF lebih baik daripada WBF karena lebih sulit untuk menemukan item di saat *rating* yang diberikan oleh semua pengguna grup serupa ketika ada terlalu banyak pengguna dalam grup. Metode yang diusung juga dibandingkan dengan metode KNN dari penelitian yang sudah dilakukan sebelumnya. Penelitian tersebut menunjukkan bahwa algoritma (MF) yang dapat menangkap faktor laten dari pengguna dan item lebih unggul dibanding algoritma tradisional K-*Nearest Neighbor* (KNN).
2.	*Neural Collaborative Filtering* (He et al., 2017)	*Collaborative Filtering* dengan *Neural Network* berbasis algoritma *Matrix Factorization*	Pada perbandingan performansi model yang dibangun, model yang diusung mencapai performa terbaik pada kedua kumpulan data yang diuji (MovieLens dan Pinterest), dan secara signifikan mengungguli

No.	Judul Penelitian	Metode	Hasil
			metode canggih eALS dan BPR dari penelitian terdahulu dengan selisih yang besar. Ini menunjukkan ekspresi tinggi dari NeuMF dengan menggabungkan model linier MF dan nonlinier MLP. Dua metode lainnya, GMF dan MLP, juga menunjukkan performa yang cukup kuat.
3.	*xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems* (Lian et. al., 2018)	*Deep Neural Network* dengan CNN dan RNN	Model sistem rekomendasi dibangun menggunakan *Convolutional Neural Netowrk* (CNN) dan *Recurrent Neural Network* (RNN). Peneliti menggabungkan model yang diusulkan dengan DNN klasik menjadi satu model terpadu, yang dinamakan dengan eXtreme Deep Factorization Machine (xDeepFM). Hasil menunjukkan bahwa xDeepFM mengungguli model sistem rekomendasi dari yang sudah ada.
4.	*Collaborative Filtering Using Neural Networks for*	*Collaborative Filtering* dengan *Neural Network*	Pada metode *collaborative filtering* tradisional (KNN), nilai *Mean Absolute Error*

No.	Judul Penelitian	Metode	Hasil
	Explicit Feedback Recommendation Systems (Pandey et. al., 2019)		(MAE) yang diperoleh adalah 1,3. Pada metode *Matrix Factorization* dengan *Neural Networks*, nilai MAE yang diperoleh adalah 0,938. Pada metode *collaborative filtering* menggunakan FastAI, nilai MAE yang diperoleh adalah 0,843. Pada metode *collaborative filtering* dengan *Neural Networks* dengan FastAI, nilai MAE yang diperoleh adalah 0,824. Pada metode NCF yang diusung pada penelitian sebelumnya (NeuMF), nilai MAE yang diperoleh adalah 0,818.
5.	*Deep Learning Recommendation Model for Personalization and Recommendation Systems* (Naumov et. al., 2019)	*Matrix factorization* dengan *Deep Learning*	Hasil penelitian menunjukkan bahwa kedua model menunjukkan performa yang baik dengan model yang dijalankan dengan *optimizer* Adagrad sedikit lebih baik dibanding model yang dijalankan dengan SGD.
6.	*An Efficient Deep Learning Approach*	*Collaborative Filtering* dengan	Model yang diusulkan lebih unggul dibandingkan dengan metode konvensional seperti

No.	Judul Penelitian	Metode	Hasil
	for Collaborative Filtering Recommender System (Aljunid & D, 2020)	algoritma *Deep Learning*	*User Avg*, *Movie Avg*, *Movie-User Avg*, *users-based cosine similarity*, *items-based cosine similarity*, *Singular Value Decomposition* (SVD), dan *Matrix Factorization* (MF). Performa model yang ditunjukkan pada set data MovieLens 1M lebih baik daripada MovieLens 100k karena ukuran data pelatihan yang lebih besar. Model yang diusulkan mencapai RMSE yang lebih rendah sehubungan dengan pendekatan yang ada.
7.	Sistem Rekomendasi *Beauty Shop* Berbasis *Collaborative Filtering* (Erlangga & Sutrisno, 2020)	*Collaborative Filtering* dengan *Adjusted Cosine Similarity*	Setelah dilakukan uji instrumen dengan 17 pertanyaan pada 5 responden dan kesalahan total 0, koefisien fungsional sistem adalah 1. Rumus pengukuran fungsionalitas menjelaskan bahwa nilai yang mendekati angka 1 (0 <= Kr <= 1) dianggap sudah baik. Berdasarkan hasil tersebut, maka hasil uji instrumen pada sistem rekomendasi *beauty*

No.	Judul Penelitian	Metode	Hasil
			shop ini dianggap memenuhi kriteria layak.
8.	*Penerapan Metode Item-based Collaborative Filtering untuk Sistem Rekomendasi Data MovieLens* (Jaja et. al., 2020)	*Item-based Collaborative Filtering*	Hasil penelitian berupa model sistem rekomendasi berbasis *Item-based Collaborative Filtering*. Perhitungan kemiripan antaritem dilakukan dengan bantuan recommenderlab. Model dibuat dengan program R menggunakan dataset MovieLens yang berisi 100.000 *rating* dari 943 pengguna pada 1664 flm.
9.	*Tourist Places Recommender System Using Cosine Similarity and Singular Value Decomposition Methods* (Tyas et. al., 2021)	*Cosine Similarity* dan *Singular Value Decomposition* (SVD)	Hasil pengukuran tingkat akurasi dengan RMSE menghasilkan nilai sebesar 3,489 pada metode *Cosine Similarity*, sedangkan metode SVD mendapatkan nilai sebesar 1,231. Nilai pada metode SVD lebih kecil dibandingkan dengan metode *Cosine Similarity* dengan selisih nilai sebesar 2,258.

No.	Judul Penelitian	Metode	Hasil
10.	*Smartphone Purchase Recommendation System Using the K-Nearest Neighbor (KNN) Algorithm* (Setiaji et. al., 2022)	K-*Nearest Neighbor* (KKN)	Penelitian ini dibuat menggunakan metode K-*Nearest Neighbor* (KKN) untuk melihat *rating* yang di-*review* oleh pengguna lain yang telah mencoba menggunakan *smartphone* mereka dengan merek ponsel yang berbeda. Metode perhitungan pada penelitian ini menggunakan 3 perhitungan KNN dan menggunakan konsep perhitungan gabungan untuk mencari hasil rekomendasi yang maksimal. Hasil dari sistem rekomendasi dengan menggunakan metode K-*Nearest Neighbor* berupa *review* yang menyatakan setuju atau tidak setujunya pengguna. Dalam studi saat ini, ada 100 ulasan dari pengguna, dan memiliki persentase 78% untuk sukses dan 22% gagal.

BAB III
METODE PENELITIAN

3.1. Objek dan subjek penelitian

Objek pada penelitian ini adalah model rekomendasi yang dihasilkan dari algoritma hasil eksperimen berbasis jaringan saraf pada bahasa pemrograman Phyton. Subjek pada penelitian ini adalah *dataset* Letterboxd Movie Ratings yang ada pada situs Kaggle.

3.2. Jenis dan sumber data

Jenis data yang digunakan adalah data sekunder berupa *dataset* berisi nama pengguna, *rating* pengguna, dan judul film. Data ini diperoleh dari situs Kaggle yang menyimpan *dataset* dari hasil *data scraping* pada platform Letterboxd.

3.3. Metode pengumpulan data

Metode pengumpulan data dilakukan dengan mengakses dan mengunduh *dataset* yang disimpan pada situs Kaggle yang bersifat *open source*.

3.4. Metode pengolahan data

Metode pengolahan data dibagi menjadi dua bagian, yaitu metode sistem rekomendasi yang digunakan dan metode pembuatan algoritma model. Berikut merupakan penjabarannya.

3.4.1. Metode Sistem Rekomendasi

Sistem rekomendasi film yang dibuat pada penelitian ini menggunakan metode *collaborative filtering*. Metode *collaborative filtering* merekomendasikan item berdasarkan ukuran kesamaan antara pengguna dan/atau item, sehingga sistem ini merekomendasikan item-item yang disukai oleh pengguna yang sejenis (Kumar et. al., 2015). Dalam metode *collaborative filtering*, terdapat dua jenis pendekatan yang dapat dilakukan, yaitu *user-based* dan *item-based*. Penelitian ini menggunakan *item-based*

collaborative filtering dalam membangun sistem rekomendasi. Pendekatan *item-based* memodelkan preferensi pengguna terhadap item berdasarkan *rating* item yang serupa oleh pengguna yang sama, sehingga yang akan direkomendasikan adalah sejumlah atau kumpulan item yang serupa kepada pengguna tertentu (Ricci et. al., 2010). *Item-based collaborative filtering* digunakan karena keunggulannya yang hanya memerlukan *rating* pengguna sebagai *explicit feedback* di mana *rating* dianggap sebagai nilai kualitas yang diberikan pengguna secara eksplisit. Metode ini lebih disukai karena pengguna lebih familier terhadap item yang sebelumnya disukai daripada pengguna lain dengan kesukaan yang sama (Bel & Koren dalam Treerattanapitak & Jaruskulchai, 2012).

3.4.2. Metode pembuatan algoritma model

Pengolahan data dilakukan dengan membangun model menggunakan algoritma bahasa pemrograman. Pada penelitian ini, bahasa pemrograman yang digunakan adalah bahasa pemrograman Phyton. Bahasa pemrograman Python merupakan bahasa pemrograman beraras-tinggi yang diciptakan oleh Guido van Rossum pada tahun 1989. Bahasa inti Python hanya menyediakan sedikit tata bahasa dan kosakata sehingga mudah untuk diingat. Python mendukung banyak pustaka yang tersimpan dalam modul-modul. Sejumlah pustaka yang tersedia di antaranya mendukung jaringan, antarmuka grafis, pencitraan, analisis dan komputasi numerik, *hypertext* (HTML, XML, dll.), akses *database*, dan sebagainya (Kadir, 2005).

Algoritma yang dibangun pada sistem rekomendasi ini merupakan algoritma berbasis *neural network*. Algoritma yang digunakan pada sistem rekomendasi tradisional hanya dapat menghitung kedekatan atau kesamaan antaritem dengan masih memiliki masalah terkait skalabilitas di mana jumlah pengguna dan item pada suatu platform terus meningkat. Dengan menggunakan *neural network*, dibanding hanya melakukan perhitungan berulang-ulang setiap data diperbarui, model sistem rekomendasi dapat dibangun dari *dataset ratings* yang sudah ada sehingga model tersebut nantinya dapat digunakan untuk dijadikan sistem rekomendasi (Lazy Programmer Inc., 2018).

Model sistem rekomendasi *collaborative filtering* dengan *neural network* menggabungkan konsep algoritma *Matrix Factorization* (MF) dengan *Multi-Layer*

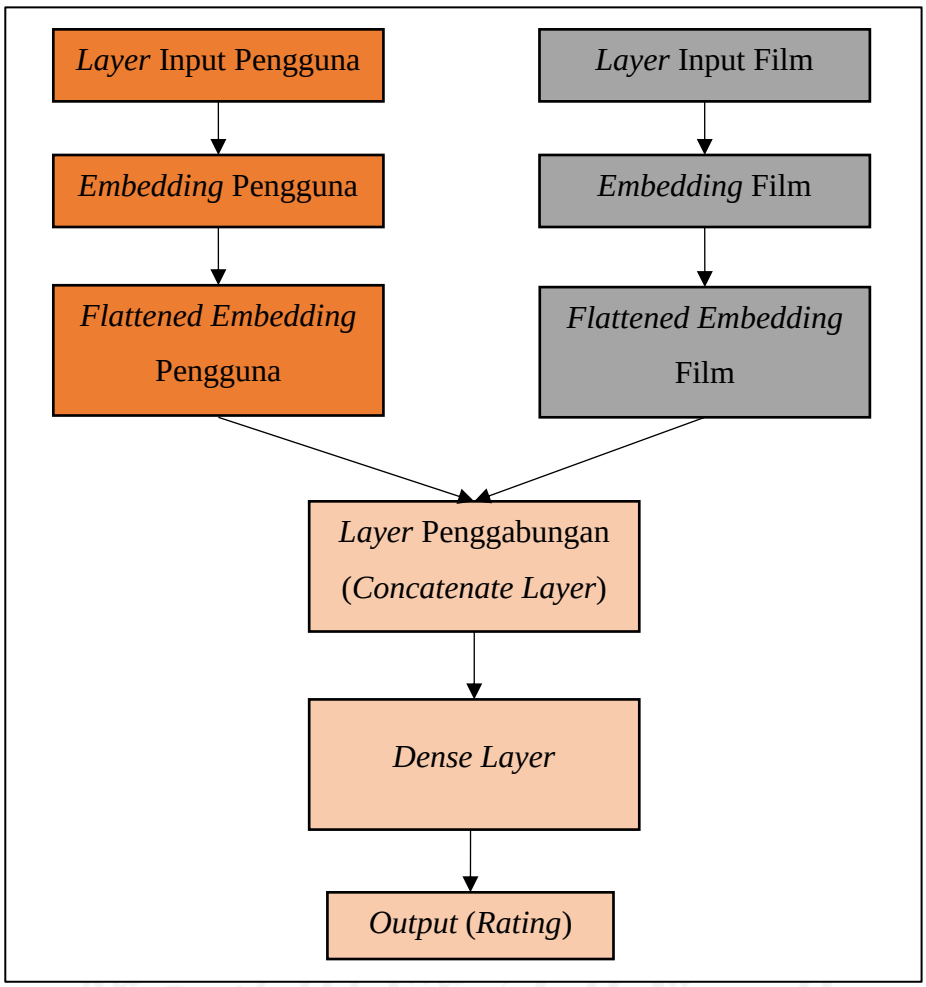
Perceptron (MLP) (Faizin & Surjandari, 2020). MF merupakan algoritma yang digunakan dalam menemukan faktor laten (tersembunyi) untuk menggambarkan korespodensi antara item dan pengguna yang disimpulkan dari pola *rating* yang diberikan pada item (Koren et. al., 2009). Misalkan p_u dan q_i masing-masing menunjukkan vektor laten untuk pengguna *u* dan item *i*, maka estimasi interaksi $\hat{r}_{ui}$ sebagai *inner product* dari p_u dan q_i adalah sebagai berikut (He et. al., 2017):

$$\hat{r}_{ui} = {p_u}^T q_i = \sum_{k=1}^{K} p_{uk} q_{ik} \tag{3.1}$$

K menunjukkan dimensi ruang laten. MF memodelkan interaksi dua arah dari faktor laten pengguna dan item, dengan asumsi bahwa setiap dimensi ruang laten bersifat independen satu sama lain dan menggabungkannya secara linier dengan bobot yang sama. Dengan demikian, MF dapat dianggap sebagai model linier dari faktor laten.

Cara kerja MF dengan *inner product* sederhana dari fitur *embedding* (faktor laten) pengguna dan item seringkali tidak cukup untuk menangkap dan merepresentasikan hubungan kompleks yang ada pada pengguna dan item. Maka dari itu, digunakanlah *neural network* untuk menemukan hubungan nonlinear yang ada dengan mengganti *inner product* dari MF dengan jaringan berbasis saraf (*neural*). Sistem rekomendasi dengan jaringan saraf ini dibuat dengan menggabungkan *layer* dari pengguna dan film (Pandey et. al., 2019). *Neural network* dapat menemukan pola untuk memetakan data input (pengguna dan film) menuju *output* berupa prediksi *rating* dengan mencari sekumpulan nilai bobot yang sesuai untuk setiap *layer*, sehingga akan menghasilkan prediksi *rating* yang sesuai dengan target yang diberikan (data *rating* aktual) (Chollet, 2018). Model ini juga dapat menemukan hubungan kompleks antara penguna dan film dengan menggunakan fitur *embedding* pada model sehingga nantinya model dapat memprediksi *rating* yang diberikan pengguna pada film yang belum diberi *rating* oleh pengguna. Metode ini juga dapat melakukan komputasi dan pembelajaran pada ukuran *dataset* yang sangat besar (Miotto et. al., 2018).

Model algoritma sistem rekomendasi yang dibangun pada penelitian ini dibuat dengan menggunakan Google Collaboratory. Berikut merupakan arsitektur dari model yang akan dibangun.



Gambar 3.1 Arsitektur DNN Sistem Rekomendasi

Dari gambar tersebut, model dibangun dengan *layer* input sebagai masukan dari pengguna dan film. Kemudian, input masuk ke dalam *layer embedding*, *flatten layer*, digabungkan pada *concatenate layer*, kemudian masuk ke *dense layer*, dan terakhir menghasilkan *output* prediksi nilai *rating* pada *output layer*.

3.5. Metode analisis data

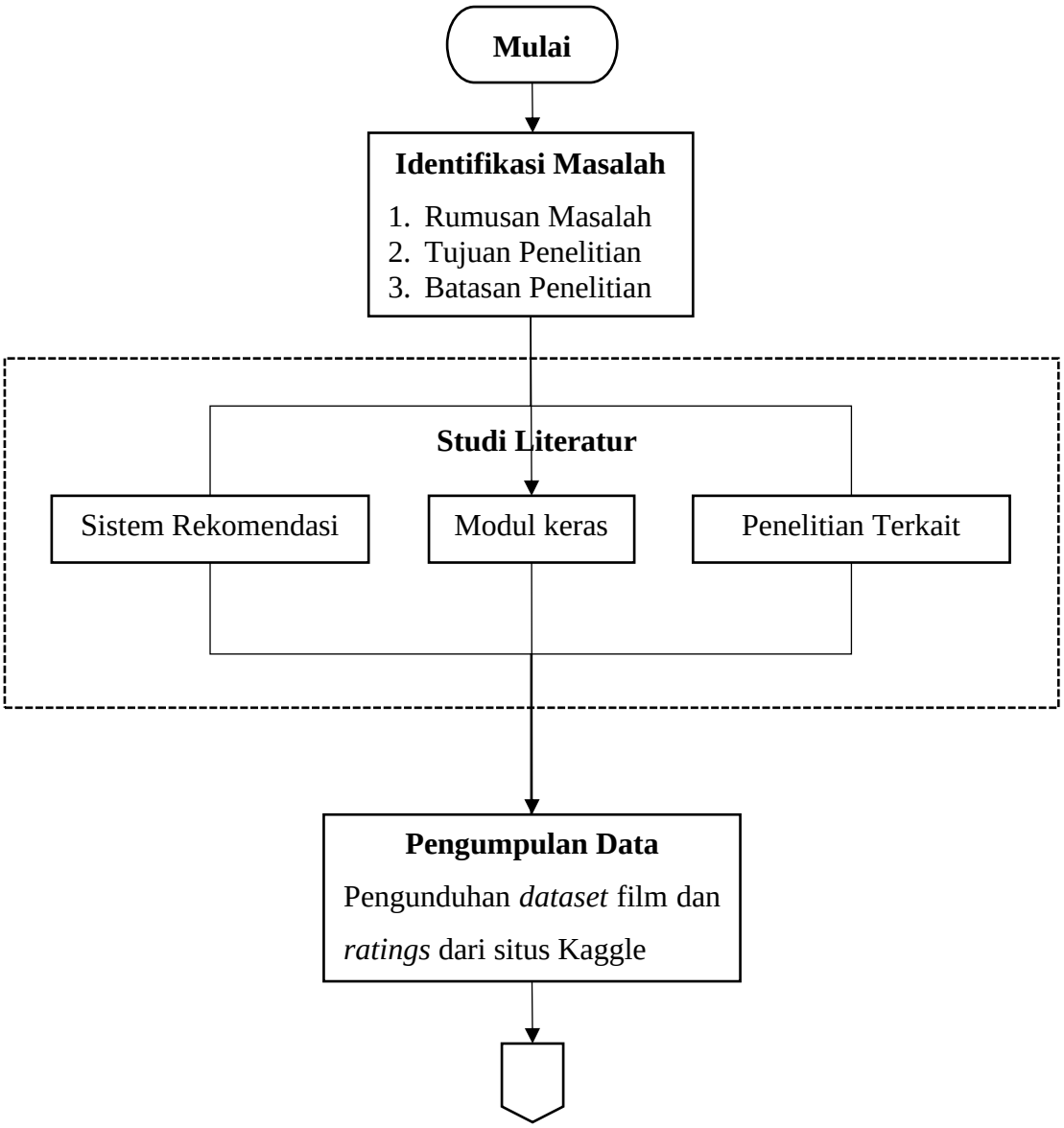
Analisis data dilakukan dengan menganalisis dan memberikan interpretasi hasil perhitungan *evaluation metrics* yang digunakan untuk melihat tingkat akurasi model algoritma yang dibangun. Tujuan dari sistem rekomendasi adalah untuk memprediksi nilai *rating* yang akan diberikan pengguna, yang merupakan variabel bernilai *real*, sehingga jika diselesaikan dengan *machine learning*, sistem rekomendasi masuk ke dalam kategori regresi (Lazy Programmer Inc., 2018). Pada penelitian ini, hasil *output* yang dihasilkan oleh *layer* pada model yang dibangun dievaluasi melalui *loss function* RMSE. Jika *y* adalah nilai *rating* target, y_i adalah nilai *rating* yang diprediksi, dan *n* adalah jumlah *ratings* atau prediksi, perhitungan *loss function* yang digunakan dirumuskan seperti bawah ini:

$$RMSE = \frac{\sqrt{\sum(y-y_i)^2}}{\sqrt{n}} \tag{3.2}$$

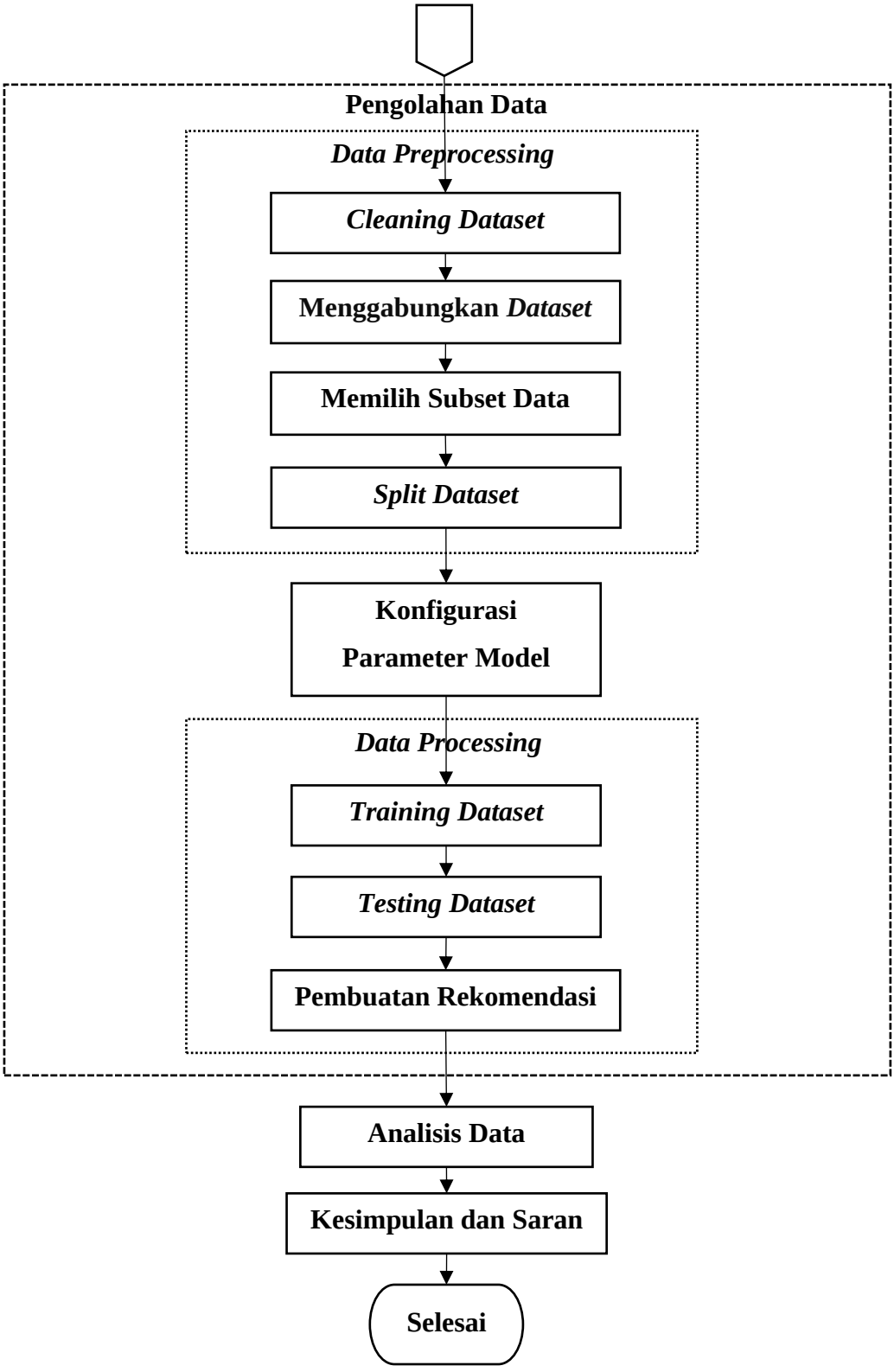
Asumsi yang mendasari RMSE adalah bahwa *error* tidak bersifat bias dan mengikuti distribusi normal. RMSE membantu memberikan gambaran lengkap tentang distribusi *error* yang ada. RMSE juga tidak menggunakan nilai mutlak, yang jauh lebih baik secara matematis saat menghitung jarak, gradien, atau metrik lainnya. Dalam bidang asimilasi data, jumlah kesalahan kuadrat sering didefinisikan sebagai *cost function* yang harus diminimalkan dengan menyesuaikan parameter model. Dalam aplikasi seperti itu, RMSE terbukti sangat efektif dalam meningkatkan kinerja model dengan menghukum nilai *error* yang besar melalui kuadrat terkecil yang sudah ditentukan. RMSE biasanya lebih baik dalam mengungkapkan perbedaan kinerja model ketika pemberian bobot yang lebih tinggi dilakukan pada kondisi yang tidak menguntungkan (Chai & Draxler, 2014).

3.6. Diagram alir penelitian

Berikut merupakan alur pada penelitian ini yang disajikan dalam bentuk diagram alir.



Gambar 3.2 Diagram alir penelitian



Gambar 3.3 Diagram alir penelitian lanjutan

Tahap-tahap yang dilakukan pada penelitian ini dapat dilihat pada Gambar 3.2 dan 3.3 dengan rincian proses sebagai berikut.

1. Mulai

   Peneliti memulai penelitian dengan memahami objek dan subjek yang akan diteliti.
2. Identifikasi Masalah

   Peneliti mengidentifikasi permasalahan yang terjadi, yaitu adanya kebutuhan untuk menentukan *optimizer* terbaik dalam membuat model sistem rekomendasi film sehingga kebutuhan ini menjadi tujuan penelitian untuk mengetahui *optimizer* apakah yang menghasilkan kinerja optimal pada model sistem rekomendasi film yang dibuat menggunakan data *open source* hasil *scraping* dari platform Letterboxd.
3. Studi Literatur

   Peneliti melakukan studi literatur dengan mencari sumber-sumber yang dibutuhkan dan penelitian terdahulu melalui jurnal penelitian dan referensi yang berkaitan dengan penelitian.
4. Pengumpulan Data

   Peneliti melakukan pengumpulan data dengan mengunduh *dataset* Letterboxd Movie Ratings yang disimpan pada situs Kaggle.
5. Pengolahan Data

   Peneliti melakukan pengolahan data yang dimulai dengan tahap *preprocessing* untuk membersihkan data, kemudian mengatur konfigurasi parameter yang digunakan. Lalu, pada tahap *processing*, data diolah ke dalam tiga bagian, yaitu *training dataset*, *validation dataset*, dan *testing dataset*, lalu membuat sistem rekomendasi untuk merekomendasikan film kepada pengguna.
6. Analisis Data

   Analisis data dilakukan dengan menganalisis hasil perhitungan tingkat akurasi prediksi *rating* dari model sistem rekomendasi yang dibuat melalui *evaluation metric* yang dipilih.
7. Kesimpulan dan Saran

Kesimpulan dan saran diberikan sebagai penutup dari penelitian yang dilakukan. Kesimpulan yang diambil menjadi jawaban rumusan masalah penelitian dan saran yang diberikan akan menjadi referensi untuk penelitian selanjutnya.

8. Selesai

BAB IV
PENGUMPULAN DAN PENGOLAHAN DATA

4.1. Pengumpulan data

Pada penelitian ini, data diperoleh dari situs Kaggle yang bersifat *open source* sehingga data dapat langsung diunduh. Data yang diunduh merupakan hasil *scraping data* dari *platform* Letterboxd pada bulan Maret 2022 yang berisi *rating* pengguna Letterboxd. Data ini memuat *rating* yang diberikan oleh 4000 pengguna teraktif pada *platform* Letterboxd. Data yang diunduh terdiri dari data film dan data *rating* dalam format .csv. Berikut ini merupakan tampilan *dataset rating* beserta rincian lainnya yang terdapat pada data tersebut.

Tabel 4.1 *Dataset* film

genres	image_url	imdb_id	imdb_link	…	movie_id	movie_title	year_released
["Music","Animation"]	film-poster/4/6/4/4/4/0/464440-football-freaks...	NaN	NaN	…	football-freaks	Football Freaks	1971
[]	film-poster/2/4/5/5/0/0/245500-aftermath-0-230...	tt0586129	http://www.imdb.com/title/tt0586129/maindetails	…	aftermath-1960	Aftermath	1960

genres	image_url	imdb_ id	imdb _link	…	movie_id	movie_ title	year_ released
["Drama"]	film-poster/9/3/3/1/8/93318-where-chimneys-are...	tt0045731	http://www.imdb.com/title/tt0045731/maindetai..ls	…	where-chimneys-are-seen	Where Chimneys Are Seen	1953
...	...	...	...	…	...	...	...
NaN	NaN	NaN	NaN	…	the-french-democracy	NaN	NaN

Tabel 4.2 *Dataset rating*

movie_id	rating_val	user_id
feast-2014	7	deathproof
loving-2016	7	deathproof
scripted-content	7	deathproof
...	...	...
x-2022	5	turnitip

Tabel 4.3 Rincian *dataset* yang digunakan

Dataset	**Jumlah Kolom**	**Jumlah Baris**	**Keterangan Nama Kolom**
Film	19	285.963	"_id", "genres", "image_url", "imdb_id", "imdb_link", "movie_id", "movie_title", "original_language", "overview", "popularity", "production_countries", "release_date", "runtime", "spoken_languages", "tmdb_id", "tmdb_link",

Dataset	Jumlah Kolom	Jumlah Baris	Keterangan Nama Kolom
			"vote_average", "vote_count", dan "year_released"
Rating	4	11.078.167	"_id", "movie_id", "rating_val", "user_id"

4.2. Data *preprocessing*

Tahapan *preprocessing* dilakukan untuk membersihkan data dari elemen-elemen yang tidak diperlukan serta menghilangkan *missing values* yang terdapat dalam *dataset*. Data *preprocessing* memiliki beberapa tahapan sebagai berikut.

4.2.1. Mengimpor library dan modul

Langkah pertama yang dilakukan pada tahap *preprocessing* adalah mengimpor *library* dan modul yang diperlukan untuk membangun model sistem rekomendasi. *Library* yang digunakan pada penelitian ini adalah pandas, numpy, matplotlib, tensorflow, dan scikit-learn.

4.2.2. Mengimpor data

Setelah data diunduh dari situs Kaggle, data kemudian diimpor pada Google Collaboratory dan ditampilkan dalam bentuk *dataframe* dengan menggunakan *library* pandas yang tersedia pada *coding environment* Google Collaboratory. Data yang diimpor adalah *dataset* film dan *dataset rating*.

4.2.3. Memilih kolom yang diperlukan

Pada *dataset* film, banyak kolom yang memberikan informasi yang tidak akan digunakan pada pembangunan sistem rekomendasi, sehingga hanya dipilih kolom tertentu saja. Kolom yang dipilih adalah kolom "movie_id", "movie_title", dan "genres".

4.2.4. Melakukan cleaning data

Missing values merupakan nilai kosong yang terdapat dalam data, biasanya disimbolkan dengan null atau NaN dalam *dataframe*. Adanya *missing values* dapat menyebabkan masalah yang berbeda seperti penurunan kinerja program, masalah analisis data, dan hasil yang bias, disebabkan oleh perbedaan antara nilai yang hilang dan nilai yang ada secara lengkap (Ayilara et. al. dalam Emmanuel et al., 2021). Pada *dataset* film, terdapat 347 *missing values* pada kolom "movie_id", 2509 *missing values* pada kolom "movie_title", dan 10977 *missing values* pada kolom "genres". Dalam penelitian ini, *missing values* diatasi dengan menghapus tiap NaN yang ada pada kolom "movie_id" dan "movie_title".

Sementara itu, pada kolom "genres" terdapat baris yang hanya diisi dengan simbol "[]" (tidak memberi informasi apakah *genre* film tersebut). Dalam kasus ini, baik baris yang berisi NaN maupun simbol "[]" diubah menjadi "Others" karena film tidak memberikan informasi *genre* spesifik, sehingga dianggap tidak masuk ke dalam kategori *genre* yang sudah ada pada *dataset*. Tampilan data yang terdapat pada kolom "genres" juga sedikit diubah demi kerapian saat kolom "genres" digunakan ketika membuat rekomendasi nanti. Setelah *missing values* dihilangkan, data yang awalnya terdiri dari 285,963 baris menjadi 283,452 baris. Berikut merupakan perbedaan tampilan datanya.

Tabel 4.4 Tampilan *dataframe* sebelum diubah

No.	movie_id	movie_title	genres
1	football-freaks	Football Freaks	["Music","Animation"]
2	aftermath-1960	Aftermath	[]
3	where-chimneys-are-seen	Where Chimneys Are Seen	["Drama"]
...	...	...	...
285962	the-french-democracy	NaN	NaN

Tabel 4.5 Tampilan *dataframe* setelah diubah

No.	movie_id	movie_title	Genres
1	football-freaks	Football Freaks	Music, Animation
2	aftermath-1960	Aftermath	Others
3	where-chimneys-are-seen	Where Chimneys Are Seen	Drama
...	...	...	...
285956	attic-2022	Attic	Horror, Thriller

4.2.5. Menggabungkan data

Penggabungan data dilakukan dengan mengambil kolom-kolom tertentu dari kedua *dataset*, kemudian digabungkan pada satu tabel. Kolom yang diambil dari *dataset* film adalah kolom "movie_id", "movie_title", dan "genres", lalu kolom yang diambil dari *dataset rating* adalah seluruh kolom (satu tabel penuh). Pada saat kedua *dataset* sudah menjadi satu *dataframe*, selanjutnya kolom "_id" dihapus. Kolom ini hanya menunjukkan informasi ID *scraping* tiap baris data yang ada pada *dataset*. Data pada kolom ini tidak dibutuhkan pada saat proses *training* nanti.

4.2.6. Memilih subset data

Pada penelitian ini, data yang akan digunakan pada pembangunan model sistem rekomendasi hanya akan diambil dalam ukuran 1.000.000 data. Hal ini dilakukan karena mempertimbangkan durasi *training* dan kemampuan GPU dalam memproses data.

4.2.7. Membuat indeks

Tiap judul film yang ada pada kolom "movie_id" dan pengguna pada kolom "user_id" diberi penomoran melalui pembuatan indeks di kolom baru karena data akan diolah dalam bentuk matriks *numpy*. Pengindeksan nomor juga dimulai dari angka 0 karena pengindeksan pada Phyton dimulai dari angka 0.

4.3. Konfigurasi parameter model

Pada penelitian ini, sesi pelatihan dibagi ke dalam beberapa skenario sesuai jenis *optimizer* yang digunakan. Pengaturan *optimizer* yang berbeda dilakukan untuk melihat performansi masing-masing model dalam menghasilkan *output* sebagai prediksi nilai *rating*. Berikut merupakan rincian konfigurasi yang diatur dalam membuat model pada sesi pelatihan.

Tabel 4.6 Konfigurasi parameter model

No.	Parameter	Training 1	Training 2	Training 3	Training 4	Training 5
1.	*Batch size*	64	64	64	64	64
2.	*Split dataset*	80:20	80:20	80:20	80:20	80:20
3.	*Optimizer*	Adam	SGD	Adagrad	Adamax	RMSProp
4.	Epochs	20	20	20	20	20

4.3.1. Batch size

Batch size adalah jumlah sampel yang akan diteruskan ke *network* pada satu waktu. Semakin besar nilai *batch size*, semakin cepat model yang dibuat menyelesaikan setiap epoch selama pelatihan. Bagaimanapun, meski *hardware* yang digunakan dalam menjalankan program dapat menangani kumpulan yang sangat besar, kualitas model yang dijalankan dapat menurun ketika *batch size* diatur dengan nilai yang lebih besar. Hal ini dapat menyebabkan model tidak dapat menggeneralisasi prediksi dengan baik untuk dilakukan pengujian nanti. Terdapat beberapa nilai yang umum digunakan sebagai *batch size*, yaitu 16, 32, 64, 128, 256. Pada penelitian ini, nilai *batch size* diatur sebesar 64, yang artinya model mengambil 64 sampel saat melakukan pelatihan pada satu iterasi di setiap epoch.

4.3.2. Optimizer

Optimizer adalah konsep umum yang digunakan dalam *neural network* yang melibatkan inisialisasi acak dan memanipulasi nilai bobot pada setiap epoch untuk meningkatkan

potensi akurasi model. Perbandingan dibuat di setiap epoch antara *output* dari data pelatihan dan data aktual, yang membantu model menghitung kesalahan dan mengetahui *loss function* dan pembaruan lebih lanjut dengan bobot yang sesuai. Beberapa jenis *optimizer* yang terdapat pada modul keras antara lain:

1. Adam, atau *Adaptive Moment estimation*, yaitu metode penurunan gradien yang ditingkatkan untuk tugas optimisasi model. Adam menggunakan gradien sebelumnya untuk menghitung gradien terbaru. Adam menggunakan konsep momentum dengan menambahkan pecahan dari gradien sebelumnya ke yang terbaru. Adam adalah *optimizer* paling populer di kalangan pengembang *neural network.*
2. *Stochastic Gradient Descent* (SGD) adalah *optimizer* yang melakukan komputasi redundan untuk kumpulan data yang lebih besar, karena SGD menghitung ulang gradien untuk contoh yang sama sebelum setiap pembaruan parameter yang digunakan. SGD sering melakukan pembaruan dengan variansi yang tinggi sehingga menyebabkan *objective function* sangat berfluktuasi.
3. Adagrad adalah metode optimisasi yang menyesuaikan *learning rate* secara khusus dengan fitur individual. Artinya, beberapa bobot dalam kumpulan data yang digunakan akan memiliki *learning rate* yang berbeda dari yang lain. Adagrad selalu bekerja paling baik dalam kumpulan data yang jarang di mana banyak *input* yang hilang.
4. Adamax adalah variansi dari Adam yang didasarkan pada *norm* tak terhingga. Terkadang Adamax dianggap lebih unggul dari Adam, terutama pada model yang menggunakan *embedding*.
5. RMSProp adalah *optimizer* versi eksklusif Adagrad yang dikembangkan oleh Geoffrey Hinton. RMSProp mempertahankan nilai *moving average* (yang terpotong) dari kuadrat gradien, lalu membagi gradien dengan akar nilai rata-rata tersebut.

4.4. Data *processing*

Pada tahap ini, *dataset* yang dipilih dibagi menjadi data *training*, data validasi, dan data *testing*. Pada penelitian ini, dilakukan skenario pembagian data *training*, data validasi, dan data *testing* dengan perbandingan 80:10:10. Setelah data dibagi, maka proses selanjutnya pun dilakukan, yaitu *training* data dan *testing* data.

Data validasi merupakan sekumpulan data yang terpisah dari data *training*. Kumpulan data ini digunakan untuk memvalidasi kinerja model yang dibangun saat *training* dilakukan. Proses validasi ini memberikan informasi untuk membantu pembuat model menyesuaikan *hyperparameter* model dan konfigurasi agar sesuai, serta memberitahu apakah model berjalan ke arah yang benar atau tidak. Pada saat model dilatih saat sesi *training*, secara bersamaan, evaluasi model dilakukan pada data validasi di setiap epochs Gagasan utama dari mengambil data validasi dari *dataset* adalah untuk mencegah model yang dibangun dari *overfitting*, yaitu ketika model menjadi sangat bagus dalam mengklasifikasikan sampel pada data pelatihan, tetapi tidak dapat menggeneralisasi dan membuat klasifikasi akurat pada data yang belum pernah dilihat sebelumnya (data *testing*, salah satunya) (Baheti, 2022).

4.4.1. Training data

Training data merupakan salah satu tahapan *processing* data di mana algoritma sistem rekomendasi dibangun. Algoritma dibangun dengan *neural network* yang tersedia pada modul Keras dari *library* Tensorflow. Sebelum membangun model *neural network*, variabel jumlah pengguna dan jumlah film dibuat.

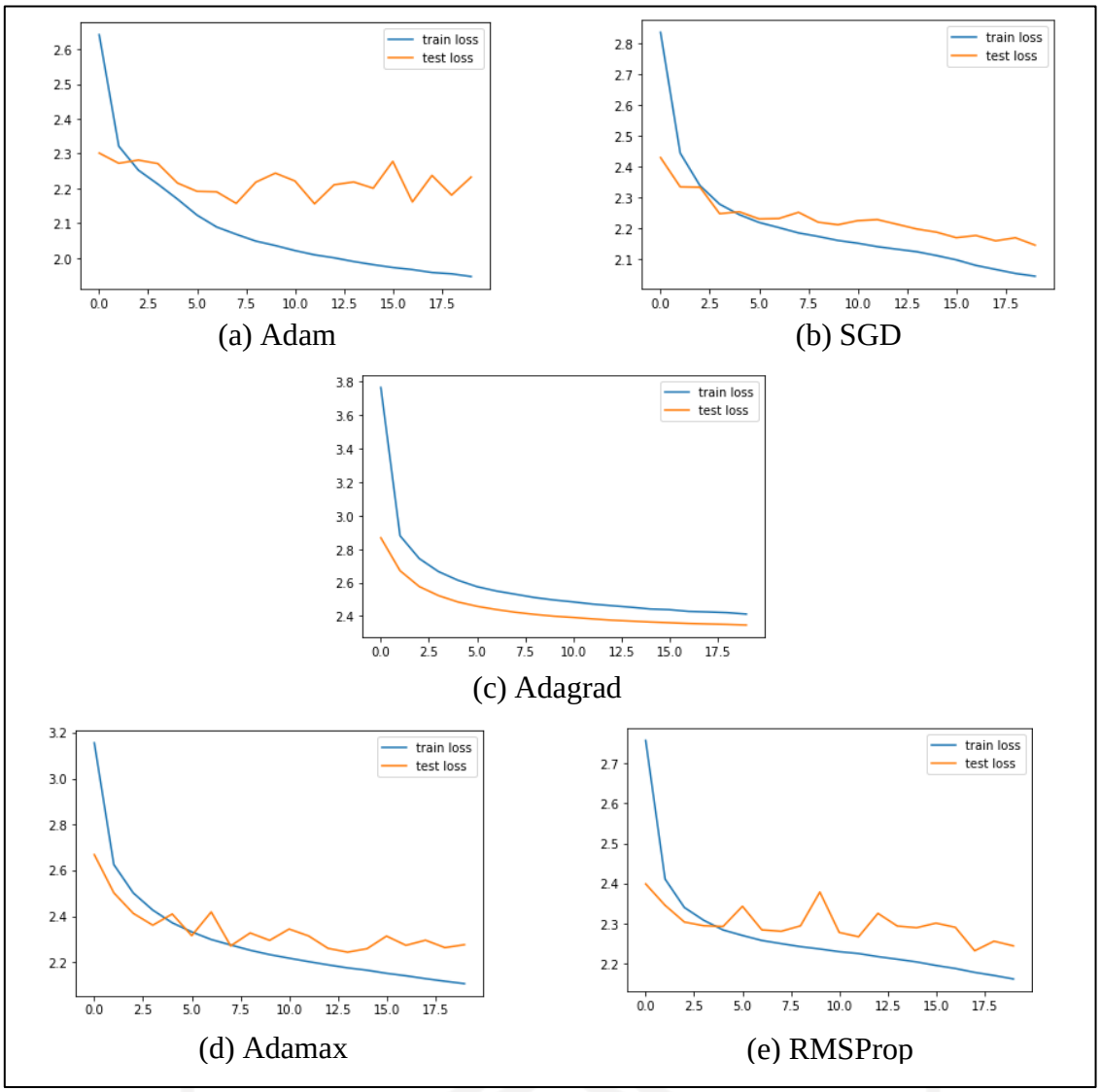
Urutan pertama pada pembuatan arsitektur model *neural network* adalah dengan mendefinisikan variabel input, lalu *embedding layer*, kemudian *flatten layer*, dan *dropout layer*. Urutan ini dilakukan pada masing-masing input pengguna dan input film. *Dropout layer* diberikan untuk mencegah terjadinya *overfitting* pada model ketika *training* dijalankan nanti. Setelah itu, masing-masing *layer* dari pengguna dan film digabungkan dalam *concatenate layer*. Setelah *layer* digabungkan menjadi satu, *layer* ini kemudian menjadi *input* untuk dimasukkan ke dalam *dense layer* yang terdiri dari tiga *layer*, kemudian menghasilkan *layer output* pada *layer* terakhir selanjutnya. Jumlah neuron

yang diberikan pada setiap *dense layer* adalah 400, 200, dan 100 neuron. Setelah membuat arsitektur model, model di-*compile* dengan mengatur *optimizer* dan *evaluation metric* yang digunakan. Pada penelitian ini, dilakukan skenario pelatihan data dengan melakukan perbandingan *optimizer* dan nilai *loss* yang dihasilkan dengan rincian tertera pada Tabel 4.7 di bawah.

Tabel 4.7 Hasil *training* data

No.	***Optimizer***	**Nilai *Loss***	
		Training Loss	***Validation Loss***
1.	Adam	1,9472	2,2372
2.	SGD	2,0641	2,1467
3.	Adagrad	2,4115	2,3462
4.	Adamax	2,1076	2,2772
5.	RMSProp	2,1606	2,2434

Pada tabel tersebut, dapat dilihat bahwa *optimizer* Adam memiliki nilai *loss* paling kecil di antara semua *optimizer* yang digunakan. Nilai *loss* yang dihasilkan pada saat menjalankan pelatihan menunjukkan seberapa baik model memetakan *output* sesuai nilai aktual yang ada pada *dataset*. Meski begitu, pada model yang dijalankan dengan *optimizer* Adam terdapat perbedaan yang jauh antara nilai *loss training* dan nilai *loss* validasi yang dihasilkan. Sementara itu, pada model yang dijalankan dengan *optimizer* yang lain, jarak antara nilai *loss training* dan nilai *loss* validasi menunjukkan perbedaan yang tidak cukup jauh, terutama pada model yang dijalankan dengan Adagrad. Berikut merupakan visualisasi grafik perbandingan antara *loss training* dan *loss validation* setiap *optimizer* yang digunakan.



Gambar 4.1 Grafik *loss training* dan *validation*

4.4.2. Testing data

Testing data dilakukan untuk menguji model setelah menyelesaikan *training data*. *Testing data* memberikan ukuran kinerja model akhir yang tidak bias dalam hal akurasi, presisi, dan sebagainya yang menunjukkan seberapa baik kinerja model yang sudah dibangun. Pada penelitian ini, *testing* atau pengujian dilakukan menggunakan lima skenario berdasarkan konfigurasi parameter model yang sudah dilakukan. Berikut ini merupakan rincian hasil nilai *loss* pada saat *testing data* dilakukan.

Tabel 4.8 Hasil *testing* data

No.	*Optimizer*	*Prediction Loss*
1.	Adam	2,2524
2.	SGD	2,1742
3.	Adagrad	2,3740
4.	Adamax	2,2908
5.	RMSProp	2,2398

Pada tabel tersebut, nilai *loss* terkecil dihasilkan pada model yang dijalankan dengan *optimizer* SGD, diikuti RMSProp, Adam, Adamax, dan Adagrad. Nilai ini menunjukkan seberapa jauh prediksi model dari nilai aktual yang ada pada *dataset*.

4.4.3. Membuat rekomendasi film kepada pengguna

Pembuatan rekomendasi dilakukan setelah model sistem rekomendasi dibangun. Pembuatan rekomendasi film menggunakan model yang dijalankan pada *optimizer* SGD, mengingat model terbaik pada penelitian ini ditinjau dari nilai *loss* yang dihasilkan saat model diuji. Meskipun pada saat *training* model yang dijalankan dengan Adam memiliki nilai *loss* yang paling kecil, tetapi saat *testing* dilakukan, SGD menunjukkan nilai *loss* prediksi yang paling rendah, menunjukkan model dengan *optimizer* SGD merupakan model dengan kinerja terbaik.

Mula-mula, pembuatan rekomendasi dilakukan dengan mengambil sampel pengguna secara acak. Kemudian, daftar film yang sudah ditonton dan belum ditonton oleh pengguna tersebut dikumpulkan. Lalu, indeks untuk setiap daftar film yang belum ditonton dibuat. Indeks pengguna juga diambil karena indeks film dan pengguna akan dijadikan satu dalam bentuk *numpy array*. Setelah kedua indeks ini berada dalam *array* yang menunjukkan pasangan antara indeks pengguna dengan indeks film yang belum ditonton dilakukan prediksi *rating* dari pengguna untuk setiap film yang terindeks pada *array* dengan model yang sudah dibuat sebelumnya. Hasil prediksi ini kemudian diurutkan dari nilai tertinggi. Pada pembuatan rekomendasi kali ini, dibuat rekomendasi

sepuluh film teratas yang mungkin akan disukai pengguna sehingga pengurutan nilai diambil dari sepuluh nilai prediksi *rating* tertinggi. Setelah didapatkan urutan sepuluh *rating* tertinggi, maka dibuat tampilan judul film beserta *genre* terhadap pengguna yang dipilih. Berikut ini merupakan contoh sampel pengguna beserta rekomendasi film yang ditawarkan oleh sistem rekomendasi yang sudah dibangun.

Tabel 4.9 Hasil rekomendasi film kepada pengguna

Nama pengguna: wilbus
Top 10 Movie Recommendations --- Compasso de Espera | Drama, Romance The Apartment | Comedy, Drama, Romance A Separation | Drama The Human Condition | Others GoodFellas | Drama, Crime Psalm II: Walking Distance | Others Amadeus | Music, Drama, History Portrait of a Lady on Fire | Drama, Romance Horace and Pete | Others PNYC: Portishead - Roseland New York | Music, Documentary
Nama pengguna: cinemagazine
Top 10 Movie Recommendations --- The Lion King | Family, Animation, Drama Raiders of the Lost Ark | Adventure, Action A Separation | Drama The Human Condition | Others Oldboy | Drama, Thriller, Mystery, Action GoodFellas | Drama, Crime

La La Land | Comedy, Drama, Romance, Music Amadeus | Music, Drama, History Portrait of a Lady on Fire | Drama, Romance The Lord of the Rings: The Fellowship of the Ring | Adventure, Fantasy, Action
Nama pengguna: willthethrill
Top 10 Movie Recommendations ---------- The Apartment | Comedy, Drama, Romance A Separation | Drama The Human Condition | Others Oldboy | Drama, Thriller, Mystery, Action GoodFellas | Drama, Crime La La Land | Comedy, Drama, Romance, Music Mulholland Drive | Thriller, Drama, Mystery Amadeus | Music, Drama, History Portrait of a Lady on Fire | Drama, Romance The Lord of the Rings: The Fellowship of the Ring | Adventure, Fantasy, Action
Nama pengguna: frenchhornhero
Top 10 Movie Recommendations ---------- Compasso de Espera | Drama, Romance The Apartment | Comedy, Drama, Romance A Separation | Drama The Human Condition | Others GoodFellas | Drama, Crime Mulholland Drive | Thriller, Drama, Mystery Amadeus | Music, Drama, History Portrait of a Lady on Fire | Drama, Romance Horace and Pete | Others

PNYC: Portishead - Roseland New York | Music, Documentary
Nama pengguna: itz_jolly
Top 10 Movie Recommendations --- Prince: Lovesexy Live | Music Compasso de Espera | Drama, Romance Habib | Drama The Apartment | Comedy, Drama, Romance The Human Condition | Others The Great Beauty: Colombia | Others La La Land | Comedy, Drama, Romance, Music Mulholland Drive | Thriller, Drama, Mystery Persona | Drama Portrait of a Lady on Fire | Drama, Romance

BAB V
ANALISIS DAN PEMBAHASAN

Berdasarkan model sistem rekomendasi yang sudah dibuat, diperoleh nilai *loss training*, *validation*, dan *testing* yang berbeda-beda. Nilai *loss training* terendah diperoleh pada model skenario 1 dengan *optimizer* Adam, sedangkan nilai *loss* prediksi atau *testing* diperoleh pada model skenario 2 dengan *optimizer* SGD. Hal ini dapat terjadi karena *optimizer* Adam merupakan *optimizer* yang paling cepat untuk membuat model *neural network* mencapai konvergen sehingga mampu memetakan input menuju *output* yang diharapkan pada data *training*. Seperti yang dinyatakan pada penelitian yang dilakukan oleh He et. al. (2017) dan Aljunid & D. (2020), bahwa Adam mampu menurunkan nilai *loss* secara cepat untuk mencapai konvergen, hanya saja kekurangan pada Adam adalah model akan cenderung melakukan *overfitting* sehingga kinerja model ketika diuji dengan data di luar *training* akan menunjukkan performansi yang buruk. Berbeda dengan SGD, *optimizer* ini dinilai lebih mampu menggeneralisasi pola prediksi yang tercipta dalam model *neural network* yang dibangun. Pada penelitian yang dilakukan oleh Raghuwanshi & Pateriya (2021) menggunakan SGD, bahwa SGD mampu melakukan generalisasi pada prediksi model yang dirancang dengan tingkat konvergensi lebih tinggi dan akurasi klasifikasi yang lebih baik.

Pada grafik perbandingan antara *loss training* dan *loss validation* (yang dalam Gambar 4.1 diberi label “test loss”) yang ditampilkan, nilai *loss training* dan *validation* dengan *optimizer* SGD terus mengalami penurunan, yang menunjukkan bahwa kemungkinan terjadinya *overfitting* kecil, sehingga model mampu memprediksi nilai *rating* dengan baik. Berbeda dengan model dengan *optimizer* Adam, *overfitting* dengan cepat terjadi dengan nilai *validation loss* yang hanya menurun hingga mencapai epoch tertentu. Pada penelitian yang dilakukan oleh Aljunid & D (2020), *training* dilakukan dengan jumlah epoch 100. Lebih banyaknya epoch memungkinkan model untuk terus melakukan pelatihan dengan menurunkan nilai *loss function* hingga mencapai konvergen sehingga model yang dibuat sudah mencapai maksimal dan kinerja terbaiknya dalam memprediksi

output yang diharapkan. Meski begitu, dinyatakan bahwa dalam penelitian Keskar & Socher (2017), metode *adaptive optimization* seperti Adam, Adagrad atau RMSprop cenderung bekerja dengan baik di awal *training*, tetapi melakukan generalisasi yang buruk.

Pada Gambar 4.1, *optimizer* Adagrad tampak melakukan penurunan nilai *loss function* secara halus, tetapi hal ini disebabkan oleh nilai *learning rate* bawaannya yang bernilai 0.001 sehingga Adagrad menurunkan nilai *loss function* secara perlahan sedikit demi sedikit dibandingkan dengan *optimizer* yang lain. Dengan jumlah epoch yang sama, Adagrad berhenti melakukan *training* dengan nilai RMSE senilai 2,4115, sementara SGD berhasil melakukan penurunan nilai RMSE sebesar 2,0641 yang menunjukkan bahwa SGD lebih unggul pada tahap *training* selanjutnya dan mampu melakukan generalisasi model dengan lebih baik.

Untuk memperoleh model yang optimal, parameter pada tiap *optimizer* dapat diubah sesuai kebutuhan, seperti parameter *learning rate*, *momentum*, dan sebagainya. Pada penelitian ini, parameter pada tiap *optimizer* diatur pada mode *default* dengan nilai parameter bawaan masing-masing *optimizer* karena penelitian ini ingin membandingkan kinerja model berdasarkan *optimizer* yang digunakan untuk menjalankan *training* pada model yang sudah dibuat.

Dengan adanya model sistem rekomendasi ini, harapannya model ini dapat digunakan dan diimplementasikan sebagai sistem rekomendasi yang dapat membantu pengguna mendapatkan rekomendasi film yang akan disukai ketika ditonton oleh pengguna. Penelitian ini masih memiliki banyak kekurangan seperti yang dijelaskan pada paragraf sebelumnya dan dapat dikembangkan lagi melalui saran-saran yang diberikan dari penelitian ini.

BAB VI
KESIMPULAN DAN SARAN

6.1. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah:

1. Algoritma model sistem rekomendasi film pada penelitian ini terdiri dari dua *layer* input, yaitu *layer* input pengguna dan film. Masing-masing *layer* diteruskan ke *embedding layer*, kemudian *flatten layer*, dan *dropout layer*. Setelah itu, *output* dari kedua *dropout layer* disatukan sebagai satu input dalam *concatenate layer*. *Concatenate layer* kemudian mengeluarkan *output* untuk dijadikan *input* ke dalam *dense layer* yang terdiri dari tiga *layer*, kemudian diteruskan ke *layer output* pada *layer* terakhir. Jumlah neuron yang diberikan pada setiap *dense layer* adalah 400, 200, dan 100 neuron.
2. Dari hasil model sistem rekomendasi yang dibangun, model dengan *optimizer* SGD merupakan model dengan kinerja model terbaik karena memiliki nilai *loss* prediksi yang paling rendah, yaitu sebesar 2,1742.

6.2. Saran

Berdasarkan penelitian ini, saran yang dapat diberikan pada penelitian selanjutnya adalah:

1. Memanfaatkan fitur yang tersedia pada *dataset* dengan metode yang sama, atau menggabungkan metode *collaborative filtering* dengan yang lain secara *hybrid*, seperti *content-based*, untuk mengatasi masalah *cold-start* pada sistem rekomendasi yang dibangun dan memberikan rekomendasi yang lebih bervariasi bagi pengguna untuk mengeksplor film dengan kriteria baru.
2. Mengembangkan model yang mampu menggunakan *dataset* dengan ukuran yang lebih besar lagi dengan durasi yang singkat.
3. Melakukan *training* data pada komputer dengan spesifikasi GPU yang lebih tinggi agar proses *training* dapat berlangsung lebih cepat.

4. Melakukan konfigurasi parameter *optimizer* yang diberikan pada model dengan lebih baik agar diperoleh hasil prediksi yang optimal.

DAFTAR PUSTAKA

Aggarwal, C. C. 2016. *Recommender Systems: The Textbook.* Yorktown Heights, NY: Springer Cham.

Aljunid, M. F. & D, H. M. 2020. An Efficient Deep Learning Approach for Collaborative Filtering. *Procedia Computer Science* **171**: 829–836.

Baheti, P. 2022. Train Test Validation Split: How To & Best Practices [2022]. (*online*): https://www.v7labs.com/blog/train-validation-test-set (23 November 2022)

Budiharto, W. 2016. *Machine Learning dan Computational Intelligence.* Yogyakarta: Penerbit ANDI.

Chai, T., & Draxler, R. R. 2014. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. *Geoscientific Model Development* **7**(3): 1247–1250.

Chollet, F. 2018. *Deep Learning with Python.* Shelter Island, NY: Manning Publication Co.

Cineflex. 2022. What We REALLY Want from Letterboxd. (*online*): https://www.youtube.com/watch?v=IiJw-JKACA0 (8 Januari 2023)

Dahria, M. 2008. Kecerdasan Buatan (*Artificial Intelligence*). *Jurnal SAINTIKOM* **5**(2): 185–196.

Desai, C. 2020. Comparative Analysis of Optimizers in Deep Neural Networks. *International Journal of Innovative Science and Research Technology* **5**(10): 959–962

Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B., & Tabona, O. 2021. A survey on missing data in machine learning. *Journal of Big Data* **8**(140): 1–37.

Erlangga, E. & Sutrisno, H. 2020. Sistem Rekomendasi Beauty Shop Berbasis Collaborative Filtering. *Jurnal Manajemen Sistem Informasi dan Teknologi* **10**(2): 47– 52.

Faizin, A. & Surjandari, I. 2020. Product recommender system using neural collaborative filtering for marketplace in Indonesia. *International Conference on Advanced Mechanical and Industrial engineering*. Banten.

Golovko, V., Kroshchanka, A., & Treadwell, D. 2016. The Nature of Unsupervised Learning in Deep Neural Networks: A New Understanding and Novel Approach. *Optical Memory and Neural Networks* **25**(3): 127–141.

Gurney, K. 1997. *An Introduction to Neural Networks.* London: UCL Press.

Ha, T. & Lee, S. 2017. Item-network-based collaborative filtering: A personalized recommendation method based on a user's item network. *Information Processing and Management* **53**: 1171–1184.

Hadi, I., Santoso, L. W., & Tjondrowiguno, A. N. 2020. Sistem Rekomendasi Film menggunakan User-based Collaborative Filtering dan K-modes Clustering. *Jurnal Infra* **8**(1): 228–234.

Haykin, S. 2009. *Neural networks and learning machines* (3rd ed.). New Jersey: Pearson Education Incorporation.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. 2017. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web*. Perth.

Jaja, Y. V., Susanto, B., & Sasongko, L. R. 2020. Penerapan Metode Item-based Collaborative Filtering untuk Sistem Rekomendasi Data MovieLens. *Jurnal Matematika dan Aplikasi* 9(2): 78–83.

Kadir, A. 2005. *Dasar Pemrograman Python.* Yogyakarta: Penerbit ANDI.

Keskar, N. S. & Socher, R. 2017. Improving Generalization Performance by Switching from Adam to SGD. *arXiv*, 1–10. doi:10.48550/ARXIV.1712.07628

Koren, Y., Bell, R., & Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* **42**(8): 30-37.

Lazy Programmer Inc. 2018. *Recommender Systems and Deep Learning in Python*. (*online*): https://deeplearningcourses.com/c/recommender-systems (15 Sep

Li, K., Zhou, X., Lin, F., Zeng, W., Wang, B., & Alterovitz, G. 2019. Sparse online collaborative filtering with dynamic regularization. *Information Sciences* **505**: 535–548.

Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., & Sun, G. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. London.

Mehlig, B. 2021. *Machine learning with neural networks.* Göteborg: University of Gothenburg.

Miotto, R., Wang, F., Wang, S., Jiang, X., & Dudley, J. T. 2018. Deep learning for healthcare: review, opportunities and challanges. *Briefings in Bioinformatics* **19**(6): 1236–1246.

Mohammed, Z. 2019. *Artificial Intelligence: Definition, Ethics and Standards.* Egypt: The British University in Egypt.

Mohri, M., Rostamizadeh, A., & Talwakar, A. 2018. *Foundations of Machine Learning.* London: The MIT Press.

Naumov, M., Mudigere, D., Shi, H. J., Huang, J., Sundaraman, N., Park, J., . . . Chen, X. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *arXiv*, 1–10. doi:10.48550/ARXIV.1906.00091

Nisbet, R., Miner, G., & Yale, K. 2018. Basic Algorithms for Data Mining: A Brief Overview. dalam R. Nisbet, G. Miner, & K. Yale, *Handbook of Statistical Analysis and Data Mining Applications* (2nd ed., hal. 121–147). Academic Press. doi:https://doi.org/10.1016/C2012-0-06451-4

Nozari, R. B. & Koohi, H. 2021. Novel implicit-trust-network-based recommendation methodology. *Expert Systems with Applications* **186**: 115709.

Ortega, F., Hernando, A., Bobadilla, J., & Kang, J. H. 2016. Recommending items to group of users using Matrix Factorization based Collaborative Filtering. *Information Sciences* **345**: 313–324.

Pandey, A., Bhawna, & Jha, S. S. 2019. Collaborative Filtering Using Neural Networks for Explicit Feedback Recommendations System. *A Preprint*, 1–5.

Priyono, A. B. 2016. Performa Apriori dan Collaborative Filtering untuk Sistem Rekomendasi. *Jurnal Ilmiah Informatika Komputer* **21**(1): 51–59.

Raghuwanshi, S. K. & Pateriya, R. K. 2021. Accelerated Singular Value Decomposition (ASVD) using momentum based Gradient Descent Momentum. *Journal of King Saud University – Computer and Information Sciences* **33**: 447–452.

Ricci, F., Rokach, L., & Shapira, B. 2010. *Recommender System Handbook.* New York: Springer.

Ritdrix, A. H. & Wirawan, P. W. 2018. Sistem Rekomendasi Buku Menggunakan Metode Item-based Collaborative Filtering. *Jurnal Masyarakat Informatika* **9**(2): 24–31.

Setiaji, B. R., Utama, D. Q., & Adiwijaya. 2022. Smartphone Purchase Recommendation System Using the K-Nearest Neighbor (KNN) Algorithm. *Media Informatika Budidarma* **6**(4): 2180–2186.

Sharma, R. & Singh, R. K. 2016. Evolution of Recommender Systems from Ancient Times to Modern Era: A Survey. *Indian Journal of Science and Technology* **9**(20): 1–12.

Statista. 2021. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025. (*online*): https://www.statista.com/statistics/871513/worldwide-data-created/ (20 Desember 2022)

Tobias, S. 2020. The Future of Film Talk Is on Letterboxd. (*online*): https://www.theringer.com/movies/2020/9/18/21444082/letterboxd-film-discussion-site-streaming-movies (7 Januari 2023)

Treerattanapitak, K. & Jaruskulchai, C. 2012. Exponential Fuzzy C-Means for Collaborative Filtering. *Journal of Computer Science and Technology* **27**(3): 567–576.

Tyas, T. A., Baizal, Z. K., & Dharayani, R. 2021. Tourist Places Recommender System Using Cosine Similarity and Singular Value Decomposition Methods. *Jurnal Media Informatika Budidarma* **5**(4): 1201– 1207.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. 2021. *Dive into Deep Learning.* arXiv preprint arXiv:2106.11342.

LAMPIRAN

Lampiran *script* *import library* dan modul

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from keras.layers import Input, Embedding, Flatten, Dropout, Dense, Concatenate
from keras.models import Model
from keras.optimizers import SGD
```

Lampiran *script* *mount* Google Drive

```
from google.colab import drive

drive.mount('/content/drive')
```

Lampiran *script* *load dataset* film dan pengguna

```
movies = open('/content/drive/My Drive/Skripsi/movie_data.csv', 'r')
ratings = open('/content/drive/My Drive/Skripsi/ratings_export.csv', 'r')

df_movies = pd.read_csv(movies)
df_ratings = pd.read_csv(ratings)
```

Lampiran *script* *cleaning dataset*

```
df_movies = df_movies.loc[:,["movie_id","movie_title","genres"]]
df_movies.isnull().sum()
```

```
df_movies = df_movies.dropna(subset=['movie_id','movie_title'])
df_movies['genres'] = df_movies['genres'].fillna('Others')
df_movies['genres'] = df_movies['genres'].replace(['[]'], 'Others')
df_movies['genres'] = df_movies['genres'].str.replace('[^\w\s^,]|_', '')
df_movies['genres'] = df_movies['genres'].str.replace(',', ', ')
```

Lampiran *script* menggabungkan kedua *dataset* menjadi satu *dataframe*

```
df = pd.merge(df_movies, df_ratings)
```

```
df = df.drop(columns=['_id'])
df = df.loc[:,['user_id','rating_val','movie_title','movie_id','genres']]
```

Lampiran *script* memilih *subset* data dan membuat indeks film dan pengguna

```
df = df.iloc[:1000000]

movie_ids = df["movie_id"].unique().tolist()
movie_dict = {x: i for i, x in enumerate(movie_ids)}
df["movie_idx"] = df["movie_id"].map(movie_dict)

user_ids = df["user_id"].unique().tolist()
user_dict = {x: i for i, x in enumerate(user_ids)}
df["user_idx"] = df["user_id"].map(user_dict)

movie2movie = {i: x for i, x in enumerate(movie_ids)}
```

Lampiran *script* *split dataset*

```
n_users = len(df.user_idx.unique())
n_movies = len(df.movie_idx.unique())


train, test = train_test_split(df, test_size=0.2, random_state=42)
val, true_test = train_test_split(test, test_size=0.1, random_state=42)
```

Lampiran *script* membangun arsitektur model *neural network*

```
# membuat jalur embedding film
movie_input = Input(shape=[1], name="Movie-Input")
movie_embedding = Embedding(n_movies+1, 10, name="Movie-
Embedding")(movie_input)
movie_vec = Flatten(name="Flatten-Movie")(movie_embedding)
movie_vec = Dropout(0.2)(movie_vec)
# membuat jalur embedding pengguna
user_input = Input(shape=[1], name="User-Input")
user_embedding = Embedding(n_users+1, 10, name="User-Embedding")(user_input)
user_vec = Flatten(name="Flatten-Users")(user_embedding)
user_vec = Dropout(0.2)(user_vec)
# menggabungkan input film dan pengguna
conc = Concatenate()([movie_vec, user_vec])
# membuat fully-connected-layers
```

```
fc1 = Dense(400, activation='relu')(conc)
fc2 = Dense(200, activation='relu')(fc1)
fc3 = Dense(100, activation='relu')(fc2)
out = Dense(1)(fc3)
```

Lampiran *script* membuat model *training* skenario 1

```
# membuat model training dan compile model
model1 = Model([user_input, movie_input], out)
model1.compile('Adam', 'mean_squared_error')
```

Lampiran *script* membuat model *training* skenario 2

```
model2 = Model([user_input, movie_input], out)
model2.compile('SGD', 'mean_squared_error')
```

Lampiran *script* membuat model *training* skenario 3

```
model3 = Model([user_input, movie_input], out)
model3.compile('Adagrad', 'mean_squared_error')
```

Lampiran *script* membuat model *training* skenario 4

```
model4 = Model([user_input, movie_input], out)
model4.compile('Adamax', 'mean_squared_error')
```

Lampiran *script* membuat model *training* skenario 5

```
model5 = Model([user_input, movie_input], out)
model5.compile('RMSProp', 'mean_squared_error')
```

Lampiran *script* menjalankan *training*

```
r1 = model1.fit(
   [train.user_idx, train.movie_idx], train.rating_val,
   epochs=20, batch_size=64,
   validation_data=([val.user_idx, val.movie_idx],
                    val.rating_val), verbose=1)
```

```
r2 = model2.fit(
   [train.user_idx, train.movie_idx], train.rating_val,
```

```
    epochs=20, batch_size=64,
    validation_data=([val.user_idx, val.movie_idx],
                     val.rating_val), verbose=1)
```

```
r3 = model3.fit(
    [train.user_idx, train.movie_idx], train.rating_val,
    epochs=20, batch_size=64,
    validation_data=([val.user_idx, val.movie_idx],
                     val.rating_val), verbose=1)
```

```
r4 = model4.fit(
    [train.user_idx, train.movie_idx], train.rating_val,
    epochs=20, batch_size=64,
    validation_data=([val.user_idx, val.movie_idx],
                     val.rating_val), verbose=1)
```

```
r5 = model5.fit(
    [train.user_idx, train.movie_idx], train.rating_val,
    epochs=20, batch_size=64,
    validation_data=([val.user_idx, val.movie_idx],
                     val.rating_val), verbose=1)
```

Lampiran *script* menjalankan *testing*

```
t1 = model1.evaluate([true_test.user_idx, true_test.movie_idx], true_test.rating_val)
```

```
t2 = model2.evaluate([true_test.user_idx, true_test.movie_idx], true_test.rating_val)
```

```
t3 = model3.evaluate([true_test.user_idx, true_test.movie_idx], true_test.rating_val)
```

```
t4 = model4.evaluate([true_test.user_idx, true_test.movie_idx], true_test.rating_val)
```

```
t5 = model5.evaluate([true_test.user_idx, true_test.movie_idx], true_test.rating_val)
```

Lampiran *script* visualisasi grafik *training*

```
plt.plot(r1.history['loss'], label="train loss")
plt.plot(r1.history['val_loss'], label="test loss")
plt.legend()
plt.show()
```

```
plt.plot(r2.history['loss'], label="train loss")
plt.plot(r2.history['val_loss'], label="test loss")
```

```
plt.legend()
plt.show()
```

```
plt.plot(r3.history['loss'], label="train loss")
plt.plot(r3.history['val_loss'], label="test loss")
plt.legend()
plt.show()
```

```
plt.plot(r4.history['loss'], label="train loss")
plt.plot(r4.history['val_loss'], label="test loss")
plt.legend()
plt.show()
```

```
plt.plot(r5.history['loss'], label="train loss")
plt.plot(r5.history['val_loss'], label="test loss")
plt.legend()
plt.show()
```

Lampiran *script* membuat rekomendasi film kepada pengguna

```
user = df.user_id.sample(1).iloc[0]
movies_watched_by_user = df[df.user_id == user]

movies_not_watched = df_movies[ ~df_movies["movie_id"].isin(movies_watched_by_user.movie_idx.values)]["movie_id"]
movies_not_watched = list(set(movies_not_watched).intersection(set(movie_dict.keys())))

movies_not_watched_index = [[movie_dict.get(x)] for x in movies_not_watched]
user_encoder = user_dict.get(user)
user_movie_array = np.hstack(([[user_encoder]] * len(movies_not_watched), movies_not_watched_index))

ratings = model2.predict([user_movie_array[:,0], user_movie_array[:,1]]).flatten()

top_ratings_indices = ratings.argsort()[-10:][::-1]
recommended_movie_ids = [movie2movie.get(movies_not_watched_index[x][0]) for x in top_ratings_indices]

print("Nama pengguna: {}".format(user))
print("====" * 9)
print("Top 10 Movie Recommendations")
```

```
print("----" * 8)
recommended_movies = df_movies[df_movies["movie_id"].isin(recommended_movie_ids)]
for row in recommended_movies.itertuples():
  print(row.movie_title, "|", row.genres)
```