

TUGAS AKHIR - EC234701

PENERAPAN MODEL MACHINE LEARNING UNTUK MENINGKATKAN EFISIENSI ALGORITMA CLOCK PADA CACHE REPLACEMENT

Muhammad Haekal Muhyidin Al-Araby

NRP 5024 22 1030

Dosen Pembimbing

Reza Fuad Rachmadi, S.T., M.T., Ph.D

NIP 19850403 201212 1 001

Dr. Arief Kurniawan, S.T., M.T

NIP 19740907 200212 1 001

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025



TUGAS AKHIR - EC234701

**PENERAPAN MODEL MACHINE LEARNING UNTUK
MENINGKATKAN EFISIENSI ALGORITMA CLOCK
PADA CACHE REPLACEMENT**

Muhammad Haekal Muhyidin Al-Araby

NRP 5024 22 1030

Dosen Pembimbing

Reza Fuad Rachmadi, S.T., M.T., Ph.D

NIP 19850403 201212 1 001

Dr. Arief Kurniawan, S.T., M.T

NIP 19740907 200212 1 001

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025

[Halaman ini sengaja dikosongkan]



FINAL PROJECT - EC234701

***APPLICATION OF MACHINE LEARNING MODELS TO
IMPROVE THE EFFICIENCY OF THE CLOCK
ALGORITHM IN CACHE REPLACEMENT***

Muhammad Haekal Muhyidin Al-Araby

NRP 5024 22 1030

Advisor

Reza Fuad Rachmadi, S.T., M.T., Ph.D

NIP 19850403 201212 1 001

Dr. Arief Kurniawan, S.T., M.T

NIP 19740907 200212 1 001

Undergraduate Study Program of Computer Engineering

Department of Computer Engineering

Faculty of Electrical Engineering and Intelligent Informatics

Sepuluh Nopember Institute of Technology

Surabaya

2025

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENERAPAN MODEL MACHINE LEARNING UNTUK MENINGKATKAN EFISIENSI ALGORITMA CLOCK PADA CACHE REPLACEMENT

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Teknik pada
Program Studi S-1 Teknik Komputer
Departemen Teknik Komputer
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh: **Muhammad Haekal Muhyidin Al-Araby**
NRP. 5024 22 1030

Disetujui oleh Tim Penguji Tugas Akhir:

Reza Fuad Rachmadi, S.T., M.T., Ph.D
NIP: 19850403 201212 1 001

(Pembimbing I)

.....

Dr. Arief Kurniawan, S.T., M.T
NIP: 19740907 200212 1 001

(Pembimbing II)

.....

Dr. Eko Mulyanto Yuniarno, S.T., M.T.
NIP: 19680601 199512 1 009

(Penguji I)

.....

Dr. Diah Puspito Wulandari, S.T., M.Sc.
NIP: 19801219 200501 2 001

(Penguji II)

.....

Arta Kusuma Hernanda, S.T., M.T.
NIP: 1996202311024

(Penguji III)

.....

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

Dr. Arief Kurniawan, S.T., M.T.
NIP. 19740907 200212 1 001

SURABAYA
November, 2025

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

APPLICATION OF MACHINE LEARNING MODELS TO IMPROVE THE EFFICIENCY OF THE CLOCK ALGORITHM IN CACHE REPLACEMENT

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Engineering at
Undergraduate Study Program of Computer Engineering
Department of Computer Engineering
Faculty of Electrical Engineering and Intelligent Informatics
Sepuluh Nopember Institute of Technology

By: **Muhammad Haekal Muhyidin Al-Araby**
NRP. 5024 22 1030

Approved by Final Project Examiner Team:

Reza Fuad Rachmadi, S.T., M.T., Ph.D
NIP: 19850403 201212 1 001

(Advisor I)

.....

Dr. Arief Kurniawan, S.T., M.T
NIP: 19740907 200212 1 001

(Co-Advisor II)

.....

Dr. Eko Mulyanto Yuniarno, S.T., M.T.
NIP: 19680601 199512 1 009

(Examiner I)

.....

Dr. Diah Puspito Wulandari, S.T., M.Sc.
NIP: 19801219 200501 2 001

(Examiner II)

.....

Arta Kusuma Hernanda, S.T., M.T.
NIP: 1996202311024

(Examiner III)

.....

Acknowledged,
Head of Computer Engineering Department ELECTICS - ITS

Dr. Arief Kurniawan, S.T., M.T.
NIP. 19740907 200212 1 001

SURABAYA
November, 2025

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa / NRP : Muhammad Haekal Muhyidin Al-Araby / 5024 22 1030
Departemen : Teknik Komputer
Dosen Pembimbing / NIP : Reza Fuad Rachmadi, S.T., M.T., Ph.D / 19850403 201212 1 001

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "PENERAPAN MODEL MACHINE LEARNING UNTUK MENINGKATKAN EFISIENSI ALGORITMA CLOCK PADA CACHE REPLACEMENT" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, November 2025

Mengetahui
Dosen Pembimbing

Mahasiswa

Reza Fuad Rachmadi, S.T., M.T., Ph.D
NIP. 19850403 201212 1 001

Muhammad Haekal Muhyidin Al-Araby
NRP. 5024 22 1030

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : Muhammad Haekal Muhyidin Al-Araby / 5024 22 1030
Department : Computer Engineering
Advisor / NIP : Reza Fuad Rachmadi, S.T., M.T., Ph.D / 19850403 201212 1
001

Hereby declared that the Final Project with the title of "*APPLICATION OF MACHINE LEARNING MODELS TO IMPROVE THE EFFICIENCY OF THE CLOCK ALGORITHM IN CACHE REPLACEMENT*" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with provisions that apply at Sepuluh Nopember Institute of Technology.

Surabaya, November 2025

Acknowledged
Advisor

Student

Reza Fuad Rachmadi, S.T., M.T., Ph.D
NIP. 19850403 201212 1 001

Muhammad Haekal Muhyidin Al-Araby
NRP. 5024 22 1030

[Halaman ini sengaja dikosongkan]

ABSTRAK

Nama Mahasiswa : Muhammad Haekal Muhyidin Al-Araby
Judul Tugas Akhir : PENERAPAN MODEL MACHINE LEARNING UNTUK
MENINGKATKAN EFISIENSI ALGORITMA CLOCK PADA
CACHE REPLACEMENT
Pembimbing : 1. Reza Fuad Rachmadi, S.T., M.T., Ph.D
2. Dr. Arief Kurniawan, S.T., M.T

Penelitian ini berfokus pada pengembangan dan analisis algoritma *cache eviction* Clock berbasis *machine learning*. Studi ini bertujuan untuk meningkatkan kinerja algoritma Clock dengan memanfaatkan model *supervised learning* dalam proses pengambilan keputusan untuk *cache eviction*. Simulasi dilakukan menggunakan perangkat lunak kustom berbasis Lib-CacheSim dengan skenario akses data *trace replay*. Kinerja algoritma yang dikembangkan dianalisis menggunakan metrik seperti *hit ratio*, *miss ratio*, dan jumlah promosi. Hasil penelitian menunjukkan bahwa mengintegrasikan *machine learning* ke dalam algoritma Clock dapat meningkatkan kinerja dengan mengurangi jumlah promosi *cache*, menjadikannya lebih optimal dibandingkan algoritma Clock konvensional.

Kata Kunci: *Cache, Clock, LRU, FIFO, Sistem, Machine Learning, Supervised Learning*

[Halaman ini sengaja dikosongkan]

ABSTRACT

Name : Muhammad Haekal Muhyidin Al-Araby
Title : *APPLICATION OF MACHINE LEARNING MODELS TO IMPROVE THE EFFICIENCY OF THE CLOCK ALGORITHM IN CACHE REPLACEMENT*
Advisors : 1. Reza Fuad Rachmadi, S.T., M.T., Ph.D
2. Dr. Arief Kurniawan, S.T., M.T

This research focuses on the development and analysis of a machine learning-based Clock cache eviction algorithm. The study aims to improve the performance of the Clock algorithm by utilizing a supervised learning model in the decision-making process for cache eviction. Simulations were conducted using custom software based on LibCacheSim with a trace replay data access scenario. The performance of the developed algorithm was analyzed using metrics such as hit ratio, miss ratio, and the number of promotions. The results indicate that integrating machine learning into the Clock algorithm can enhance performance by reducing the number of cache promotions, making it more optimal than the conventional Clock algorithm.

Keywords: *Cache, Clock, LRU, FIFO, System, Machine Learning, Supervised Learning*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kehadiran Tuhan Yang Maha Esa, yang atas rahmat dan karunia-Nya telah memberikan kekuatan dan kelancaran sehingga penulis dapat menyelesaikan Laporan Tugas Akhir yang berjudul **"PENERAPAN MODEL MACHINE LEARNING UNTUK MENINGKATKAN EFISIENSI ALGORITMA CLOCK PADA CACHE REPLACEMENT"** dengan baik dan tepat waktu.

Penelitian dan penyusunan Laporan Tugas Akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember. Proses ini tidak akan berjalan lancar tanpa adanya bimbingan, dukungan, dan doa dari berbagai pihak. Oleh karena itu, pada kesempatan ini, penulis ingin mengucapkan terima kasih yang tulus kepada:

1. Keluarga tercinta, terutama Ayah, Ibu, dan seluruh saudara, yang telah memberikan dukungan moral, materiel, serta doa yang tiada henti sepanjang perjalanan studi penulis.
2. Bapak Reza Fuad Rachmadi, S.T., M.T., Ph.D, selaku Dosen Pembimbing I, dan Bapak Dr. Arief Kurniawan, S.T., M.T, selaku Dosen Pembimbing II, yang telah dengan sabar memberikan bimbingan, arahan, dan masukan yang sangat berharga sejak awal hingga akhir penyusunan tugas akhir ini.
3. Bapak Dr. Arief Kurniawan, S.T., M.T, selaku Kepala Departemen Teknik Komputer, serta seluruh jajaran dosen dan staf di lingkungan departemen yang telah memberikan bekal ilmu dan kemudahan administrasi selama masa perkuliahan.
4. Seluruh sahabat dan rekan-rekan seperjuangan di Departemen Teknik Komputer yang telah menjadi teman diskusi, memberikan semangat, dan saling membantu dalam suka maupun duka selama menempuh pendidikan.
5. Semua pihak yang tidak dapat penulis sebutkan satu per satu, yang telah memberikan kontribusi dan dukungan dalam penyelesaian laporan ini.

Penulis menyadari bahwa Laporan Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, segala bentuk kritik dan saran yang membangun akan penulis terima dengan lapang dada demi perbaikan di masa mendatang. Akhir kata, semoga laporan ini dapat memberikan manfaat dan kontribusi bagi perkembangan ilmu pengetahuan, khususnya di bidang *caching system* dan *machine learning*.

Surabaya, November 2025

Muhammad Haekal Muhyidin Al-Araby

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR	ix
DAFTAR TABEL	xi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan Penelitian	2
1.4 Batasan Masalah	3
1.5 Sistematika Penulisan	3
2 TINJAUAN PUSTAKA	5
2.1 Penelitian Terdahulu	5
2.2 Konsep Dasar	5
2.2.1 Cache	5
2.2.2 Algoritma Eviction	6
2.2.3 FIFO (First-In, First-Out)	6
2.2.4 LRU (Least Recently Used)	6
2.2.5 CLOCK	6
2.2.6 Offline CLOCK	6
2.2.7 Machine Learning	7
2.2.8 Supervised Learning	7
2.2.9 Trace Replay	7
2.2.10 LibCacheSim	7
2.2.11 ONNX (Open Neural Network Exchange)	7

2.2.12	ONNX Runtime	7
3	METODOLOGI	9
3.1	Datasets	9
3.2	Pengumpulan Data	10
3.3	Pengolahan dan Analisis Data	11
3.4	Pengembangan Model Machine Learning	11
3.5	Implementasi Model pada Perangkat Lunak Simulasi	12
3.6	Evaluasi Model	13
3.7	Perangkat Keras dan Lunak	13
3.8	Perangkat Keras	14
3.9	Perangkat Lunak	14
4	PENGUJIAN DAN ANALISIS	15
4.1	Hasil yang Diharapkan	15
4.2	Hasil Pengujian Awal	15
4.2.1	Hasil Eksperimen Offline CLOCK	15
4.2.2	Hasil Pengembangan Model Awal	16
5	PENUTUP	19
5.1	Kesimpulan	19
5.2	Saran	19
	DAFTAR PUSTAKA	21
	BIOGRAFI PENULIS	25

DAFTAR GAMBAR

3.1	Diagram alur dari program simulasi yang dikembangkan.	10
3.2	Contoh visualisasi distribusi dari metadata <code>lifetime_freq</code>	11
3.3	Diagram proses pelatihan model <i>machine learning</i>	12
3.4	Arsitektur model <i>machine learning</i> yang digunakan.	12
3.5	Diagram alur implementasi model pada simulator.	13
4.1	Grafik hasil dari simulasi Offline CLOCK pada <i>trace</i> Meta Key-Value.	16
4.2	Grafik perbandingan hasil dari model regresi logistik.	17

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

Penelitian ini didorong oleh kemajuan pesat dalam teknologi komputasi modern, yang menuntut adanya teknik-teknik yang lebih canggih untuk manajemen *cache* objek. Dalam beberapa tahun terakhir, *machine learning* telah menjadi topik yang sangat populer dalam riset ilmu komputer. Berbagai studi telah dilakukan untuk memahami sejauh mana kapabilitas *machine learning* dapat diterapkan dalam bidang *caching* [1], [2], [3], [4], [5], [6]. Penelitian ini secara spesifik menyelidiki bagaimana *machine learning* dapat diimplementasikan dalam sistem *cache* untuk meningkatkan efisiensi algoritma populer seperti CLOCK terhadap distribusi zipfian [7].

1.1 Latar Belakang

Perkembangan teknologi komputasi telah melaju dengan kecepatan yang luar biasa selama dekade terakhir. Pada masa-masa awal, komputer terbatas pada prosesor *single-core* [8], [9], [10], [11]. Keterbatasan ini memaksa para insinyur dan pemrogram untuk merancang perangkat lunak yang dioptimalkan untuk eksekusi *single-threaded*. Seiring waktu, ketika peningkatan kinerja mulai melambat karena batasan fisik dan termal, industri beralih ke arsitektur *multi-core* [12], [13], [14], [15] dan paralel [16], [17], [18]. Transisi ini memperkenalkan tantangan baru dalam desain perangkat lunak, menuntut pengembang untuk mengadopsi model pemrograman konkuren dan paralel untuk memanfaatkan sepenuhnya kapabilitas perangkat keras modern. Akibatnya, pemahaman dan optimalisasi aplikasi *multi-threaded* telah menjadi aspek penting dalam riset dan pengembangan komputasi modern.

Caching telah lama menjadi bagian integral dari sistem komputer. Mekanisme ini memainkan peran krusial dalam meningkatkan kinerja sistem dengan menyimpan data yang sering diakses di lokasi penyimpanan yang lebih cepat [8], [19], [20]. Selama bertahun-tahun, berbagai strategi *caching* telah dikembangkan untuk mengoptimalkan efisiensi pengambilan data di berbagai lingkungan, mulai dari sistem operasi dan server web hingga sistem terdistribusi dan infrastruktur *cloud* [21], [22], [23].

Arsitektur *multi-core*, yang telah menjadi standar industri, juga menghadirkan tantangan signifikan bagi teknik *caching* yang ada [24], [25], [26]. Dalam lingkungan seperti ini, beberapa inti prosesor sering kali mengakses dan memodifikasi data *cache* yang sama secara bersamaan, yang dapat menyebabkan kondisi balapan (*race conditions*) dan inkonsistensi data [13]. Untuk menjaga kebenaran data, sebagian besar implementasi *cache* tradisional menekankan penggunaan kunci (*locks*) selama operasi penyisipan, promosi, dan modifikasi [27], [28], [29]. Namun, penggunaan *lock* yang berlebihan dapat menimbulkan persaingan (*contention*) dan mengurangi paralelisme, yang pada akhirnya membatasi kinerja sistem pada prosesor *multi-core* modern.

Berbagai kebijakan penggantian *cache* (*cache replacement policies*) telah diusulkan untuk mencapai tujuan desain yang berbeda. Kebijakan **Least Recently Used (LRU)** umum digunakan karena kemampuannya untuk mencapai *miss ratio* yang rendah dengan mempertahankan data yang sering diakses [30], [31], [32]. Namun, LRU memerlukan pembaruan dan sinkronisasi yang sering untuk memelihara informasi kebaruan data secara presisi, yang meningkatkan

overhead dari *locking*. Di sisi lain, kebijakan yang lebih sederhana seperti **First-In First-Out (FIFO)** sering kali lebih disukai karena skalabilitasnya dan biaya sinkronisasi yang lebih rendah, karena hampir tidak memerlukan *locking* [33], [34], [35].

Untuk mengatasi keterbatasan LRU dan FIFO, algoritma **CLOCK** dikembangkan sebagai aproksimasi dari LRU yang tetap mempertahankan skalabilitas tinggi [19]. **CLOCK** menggunakan *buffer* melingkar (*circular buffer*) dari entri *cache*, di mana setiap entri memiliki bit referensi yang menandakan akses terkini. Alih-alih memelihara daftar yang terurut sepenuhnya seperti LRU, algoritma ini menggerakkan "jarum jam" untuk menemukan entri yang akan diganti, sambil membersihkan bit referensi di sepanjang jalan. Desain ini secara signifikan mengurangi kebutuhan akan *locking* yang berlebihan dan pembaruan yang sering, sambil tetap mempertahankan manfaat dari LRU. Hasilnya, **CLOCK** menawarkan keseimbangan praktis antara mencapai *miss ratio* yang rendah dan menjaga skalabilitas serta kesederhanaan yang menjadi ciri khas FIFO [36].

Namun, terlepas dari keunggulannya, algoritma **CLOCK** masih mengalami masalah promosi yang tidak perlu [36], [37]. Karena setiap akses akan mengatur bit referensi tanpa memandang frekuensi atau kepentingan akses, bahkan akses tunggal atau yang jarang terjadi dapat mencegah sebuah entri untuk digusur. Akibatnya, *cache* dapat mempertahankan item yang memberikan sedikit manfaat sambil menggusur item lain dengan potensi penggunaan kembali yang lebih tinggi. Hal ini menyebabkan inefisiensi dalam pemanfaatan *cache*.

Perkembangan terkini dalam *machine learning* telah membuka kemungkinan untuk menggunakannya sebagai komponen pengambilan keputusan tambahan untuk mengurangi promosi yang tidak perlu dalam sistem *cache*. Dengan belajar dari pola akses dan mengidentifikasi objek yang kemungkinan besar tidak akan digunakan kembali, model *machine learning* dapat membantu mencegah *cache* mempromosikan entri yang memberikan sedikit manfaat. Pendekatan ini memungkinkan mekanisme *caching* untuk membuat keputusan yang lebih selektif dan terinformasi tentang objek mana yang layak dipertahankan. Dengan demikian, *machine learning* dapat melengkapi algoritma tradisional seperti **CLOCK** dengan meningkatkan efisiensi, mengurangi *overhead* manajemen, dan beradaptasi secara dinamis terhadap karakteristik beban kerja yang berubah [38], [39], [40], [41].

1.2 Rumusan Masalah

Berdasarkan pembahasan pada Latar Belakang, masalah utama yang dibahas dalam penelitian ini adalah bagaimana mengintegrasikan teknik *machine learning* ke dalam algoritma penggantian *cache* **CLOCK** untuk mengurangi promosi yang tidak perlu dan meningkatkan efisiensi sistem secara keseluruhan. Secara spesifik, studi ini bertujuan untuk mengeksplorasi bagaimana pengambilan keputusan berbasis data dapat membantu algoritma **CLOCK** untuk lebih mendekati perilaku penggantian yang optimal. Dengan demikian, penelitian ini berupaya untuk meningkatkan akurasi dan skalabilitas dari manajemen *cache* berbasis **CLOCK**.

1.3 Tujuan Penelitian

Tujuan utama dari penelitian ini adalah sebagai berikut:

1. Mengintegrasikan model *machine learning* ke dalam algoritma penggantian *cache* **CLOCK** untuk menyempurnakan proses pengambilan keputusannya.
2. Mengevaluasi efisiensi dan efektivitas algoritma **CLOCK** berbasis *machine learning* dalam

mengurangi jumlah promosi dan meningkatkan kinerja *cache* secara keseluruhan.

1.4 Batasan Masalah

Studi ini berfokus pada integrasi teknik *machine learning* ke dalam algoritma penggantian *cache* CLOCK untuk meningkatkan efisiensinya. Penelitian ini terbatas pada lingkungan simulasi *cache* berbasis perangkat lunak, bukan implementasi perangkat keras. Studi ini mengevaluasi metrik kinerja seperti *miss ratio* dan tingkat promosi.

Ruang lingkup penelitian tidak mencakup pengembangan arsitektur *machine learning* baru atau perbandingan dengan sistem *caching* berbasis *reinforcement learning* [42] yang canggih. Sebaliknya, penelitian ini menekankan analisis tentang bagaimana model ringan yang ada dapat meningkatkan pengambilan keputusan di CLOCK tanpa meningkatkan kompleksitas secara signifikan. Penelitian ini hanya berfokus pada promosi dan *miss ratio* serta tidak menekankan performa asli pada perangkat keras nyata. Penelitian ini juga terbatas pada *datasets* berbasis zipfian.

1.5 Sistematika Penulisan

Laporan tugas akhir ini dibagi menjadi lima bab sebagai berikut:

1. BAB I Pendahuluan

Bab ini menyajikan latar belakang studi, masalah utama yang akan dibahas, tujuan penelitian, dan ruang lingkup studi. Bab ini memberikan gambaran umum tentang motivasi dan signifikansi topik penelitian.

2. BAB II Tinjauan Pustaka

Bab ini membahas karya-karya sebelumnya dan studi terkait yang menjadi landasan penelitian ini. Bab ini menyoroti algoritma penggantian *cache* yang ada, pengembangan algoritma CLOCK, dan aplikasi *machine learning* sebelumnya dalam sistem *caching*.

3. BAB III Perancangan dan Implementasi Sistem

Bab ini menjelaskan desain dan implementasi sistem yang diusulkan. Bab ini menjelaskan arsitektur sistem secara keseluruhan, alur data, integrasi model *machine learning*, dan proses adaptasi model ke algoritma penggantian *cache* CLOCK.

4. BAB IV Pengujian dan Analisis

Bab ini menyajikan prosedur pengujian, pengaturan eksperimental, dan metrik evaluasi yang digunakan untuk mengukur kinerja sistem. Bab ini juga memberikan analisis hasil, membandingkan pendekatan yang diusulkan dengan metode dasar dalam hal efisiensi dan pengurangan promosi.

5. BAB V Penutup

Bab ini merangkum temuan dan kontribusi penelitian. Bab ini juga membahas kemungkinan perbaikan dan rekomendasi untuk pekerjaan di masa depan terkait manajemen *cache* berbasis *machine learning*.

[Halaman ini sengaja dikosongkan]

BAB II

TINJAUAN PUSTAKA

Bab ini menyajikan tinjauan mendalam mengenai konsep-konsep dasar dan penelitian terdahulu yang menjadi landasan bagi studi ini. Pembahasan mencakup prinsip-prinsip fundamental *caching*, algoritma penggantian yang sudah mapan, serta berbagai upaya sebelumnya dalam meningkatkan sistem manajemen *cache*.

2.1 Penelitian Terdahulu

Algoritma CLOCK memiliki sejarah panjang dalam dunia komputasi. Algoritma ini pertama kali dikembangkan pada awal tahun 1970-an oleh para peneliti IBM untuk memenuhi kebutuhan sistem penggantian halaman (*page-replacement*) pada komputer *mainframe* yang memiliki *overhead* lebih rendah dibandingkan dengan algoritma LRU yang populer pada masanya [43].

Sejak diperkenalkan, beberapa pengembangan signifikan telah dibangun di atas algoritma CLOCK asli. Salah satu contoh yang paling terkenal adalah **Clock with Adaptive Replacement (CAR)**. Algoritma ini secara cerdas memanfaatkan dua buah daftar terpisah untuk mengelola objek yang baru diakses dan objek yang sering diakses. Keunggulan utama CAR terletak pada kemampuannya untuk secara dinamis menyeimbangkan kedua daftar ini berdasarkan pada kebaruan (*recency*) dan frekuensi (*frequency*) akses objek. Hal ini memungkinkan CAR untuk beradaptasi secara lebih efektif terhadap perubahan pola akses data dibandingkan dengan algoritma CLOCK standar atau LRU [44].

Pengembangan penting lainnya adalah algoritma **Clock-Pro** [45]. Algoritma ini memperkenalkan pendekatan yang lebih kompleks dengan mengkategorikan objek ke dalam tiga daftar: *Hot*, *Cold*, dan *Test*. Objek yang sering diakses akan dipromosikan ke daftar *Hot*, di mana mereka mendapatkan perlindungan dari proses penggusuran (*eviction*). Namun, jika sebuah objek di daftar *Hot* tidak diakses dalam periode yang lama, objek tersebut akan diturunkan ke daftar *Cold*. Daftar *Cold* berisi objek-objek yang baru diakses dan menjadi kandidat utama untuk digusur. Yang paling unik adalah daftar *Test*, yang tidak menyimpan objek secara langsung, melainkan hanya metadata dari objek yang baru saja digusur. Jika sebuah objek yang metadatanya ada di daftar *Test* diakses kembali dalam waktu singkat setelah penggusurannya, objek tersebut akan langsung dipromosikan ke daftar *Hot*. Mekanisme ini memungkinkan Clock-Pro untuk mencapai keseimbangan yang baik antara kebaruan dan frekuensi, beradaptasi dengan berbagai pola akses, dan mencapai *hit ratio* yang sebanding dengan LRU namun dengan *overhead* yang tetap rendah seperti CLOCK.

2.2 Konsep Dasar

2.2.1 Cache

Cache adalah sebuah lapisan penyimpanan berukuran kecil namun berkecepatan sangat tinggi yang berfungsi untuk menyimpan data yang sering diakses. Tujuan utama dari penggunaan *cache* adalah untuk mempercepat waktu akses data, yang pada akhirnya akan meningkatkan

kinerja sistem secara keseluruhan [21], [22], [23]. Mekanisme *caching* diimplementasikan di hampir semua sistem komputasi modern untuk memastikan bahwa objek yang baru atau sering digunakan dapat diakses secara instan tanpa harus mengambilnya dari media penyimpanan yang lebih lambat atau melalui proses komputasi ulang yang memakan waktu.

2.2.2 Algoritma Eviction

Algoritma *eviction* atau *replacement* (penggusuran) adalah metode yang digunakan untuk menentukan data mana yang harus dikeluarkan dari *cache* ketika kapasitasnya sudah penuh dan ada data baru yang perlu dimasukkan [19]. Algoritma ini sangat krusial untuk menjaga efisiensi *cache* dengan memastikan bahwa data yang paling relevan tetap tersimpan sementara data yang sudah jarang digunakan dapat digantikan. Contoh umum dari algoritma ini adalah FIFO, LRU, dan CLOCK.

2.2.3 FIFO (First-In, First-Out)

FIFO adalah salah satu algoritma *eviction* yang paling sederhana [33], [46]. Sesuai dengan namanya, kebijakan ini akan mengeluarkan data yang pertama kali masuk ke dalam *cache*. FIFO tidak mempertimbangkan frekuensi atau kebaruan akses data, sehingga data yang paling lama berada di dalam *cache* akan selalu menjadi yang pertama untuk digantikan. Meskipun memiliki *overhead* paling rendah, algoritma ini umumnya memiliki *miss ratio* yang paling tinggi dibandingkan algoritma populer lainnya.

2.2.4 LRU (Least Recently Used)

LRU adalah algoritma *eviction* yang menggantikan data yang paling lama tidak diakses [33]. Algoritma ini beroperasi berdasarkan asumsi bahwa data yang baru saja diakses memiliki kemungkinan besar untuk diakses kembali dalam waktu dekat. LRU biasanya diimplementasikan menggunakan struktur data antrian (*queue*) di mana setiap objek yang diakses akan dipindahkan ke kepala antrian, sehingga melindunginya dari proses *eviction*.

2.2.5 CLOCK

Algoritma CLOCK adalah variasi dari kebijakan *eviction* yang memberikan "kesempatan kedua" pada data sebelum dikeluarkan dari *cache* [36]. Setiap entri dalam *cache* memiliki sebuah bit referensi. Ketika proses *eviction* diperlukan, algoritma ini akan memeriksa bit referensi dari entri yang ditunjuk oleh "jarum jam". Jika bit referensi bernilai 0, data tersebut akan diganti. Jika bernilai 1, bit tersebut akan di-reset menjadi 0 dan jarum jam akan bergerak ke entri berikutnya. Proses ini terus berulang hingga ditemukan entri dengan bit referensi 0. Karena mekanisme ini, CLOCK sering disebut sebagai *Second Chance Algorithm*. Algoritma ini pada dasarnya berbasis FIFO, namun dengan modifikasi di mana item akan dimasukkan kembali ke kepala antrian jika bit referensinya 1, sehingga sering juga disebut sebagai *FIFO-Reinsertion*. Algoritma ini mengkombinasikan *hit ratio* tinggi dari LRU dengan *overhead* rendah dari FIFO.

2.2.6 Offline CLOCK

Offline CLOCK bukanlah algoritma praktis, melainkan sebuah tolok ukur teoretis. Dalam skenario ini, algoritma CLOCK dijalankan secara berulang-ulang pada set data dan konfigurasi yang sama. Pada setiap iterasi, setiap promosi objek yang ternyata tidak diakses kembali (promosi sia-sia) akan dicatat. Pada iterasi berikutnya, promosi sia-sia tersebut akan diabaikan. Proses ini secara dramatis mengurangi jumlah total promosi yang dilakukan dan digunakan

sebagai batas atas (*upper bound*) dari kinerja optimal yang bisa dicapai [36].

2.2.7 Machine Learning

Machine learning adalah cabang dari kecerdasan buatan (AI) yang berfokus pada pengembangan algoritma dan model yang memungkinkan komputer untuk "belajar" dari data dan membuat prediksi atau keputusan tanpa diprogram secara eksplisit. Dalam *machine learning*, sebuah sistem mempelajari pola atau hubungan dari data latih (*training data*) untuk kemudian diterapkan pada data baru yang belum pernah dilihat sebelumnya [38], [39], [42].

2.2.8 Supervised Learning

Supervised learning (pembelajaran terarah) adalah salah satu pendekatan utama dalam *machine learning* di mana sebuah model dilatih menggunakan data yang sudah berlabel. Artinya, setiap data masukan (*input*) dipasangkan dengan keluaran (*output*) yang sudah diketahui. Tujuannya adalah untuk menciptakan sebuah model yang mampu memetakan *input* ke *output* secara akurat, sehingga dapat digunakan untuk melakukan prediksi pada data baru [38].

2.2.9 Trace Replay

Trace replay adalah sebuah metode yang umum digunakan dalam evaluasi sistem komputer, khususnya dalam penelitian terkait *cache*, memori, atau sistem file [47]. Dalam metode ini, urutan akses data (disebut *trace*) dari sebuah sistem nyata direkam terlebih dahulu, kemudian "diputar ulang" (*replayed*) pada sistem atau model yang sedang diuji. *Trace replay* memungkinkan peneliti untuk menganalisis kinerja sebuah algoritma dengan skenario akses data yang realistis tanpa harus menjalankan aplikasi aslinya secara langsung.

2.2.10 LibCacheSim

LibCacheSim adalah sebuah pustaka (*library*) yang digunakan untuk melakukan simulasi *cache* pada sistem komputer [47]. Pustaka ini memungkinkan peneliti dan pengembang untuk menguji dan menganalisis kinerja berbagai kebijakan penggantian *cache* (seperti LRU, FIFO, CLOCK) dalam berbagai skenario akses data.

2.2.11 ONNX (Open Neural Network Exchange)

ONNX adalah sebuah format *open-source* yang dirancang untuk merepresentasikan model *machine learning* [48], [49], [50]. Format ini berfungsi sebagai standar terbuka yang memungkinkan pengembang untuk memindahkan model antar-kerangka kerja (*-framework*) *machine learning* yang berbeda, seperti PyTorch dan TensorFlow [51]. Tujuannya adalah untuk memfasilitasi interoperabilitas, sehingga model yang dilatih di satu *framework* dapat dengan mudah dijalankan di *framework* lain [50].

2.2.12 ONNX Runtime

ONNX Runtime adalah sebuah *inference engine* berkinerja tinggi yang dikembangkan oleh Microsoft untuk menjalankan model yang disimpan dalam format ONNX [48], [49], [50], [51]. Tujuannya adalah untuk memaksimalkan kinerja inferensi (proses penggunaan model yang sudah dilatih) di berbagai jenis perangkat keras (CPU, GPU, dll.) dan sistem operasi.

[Halaman ini sengaja dikosongkan]

BAB III

METODOLOGI

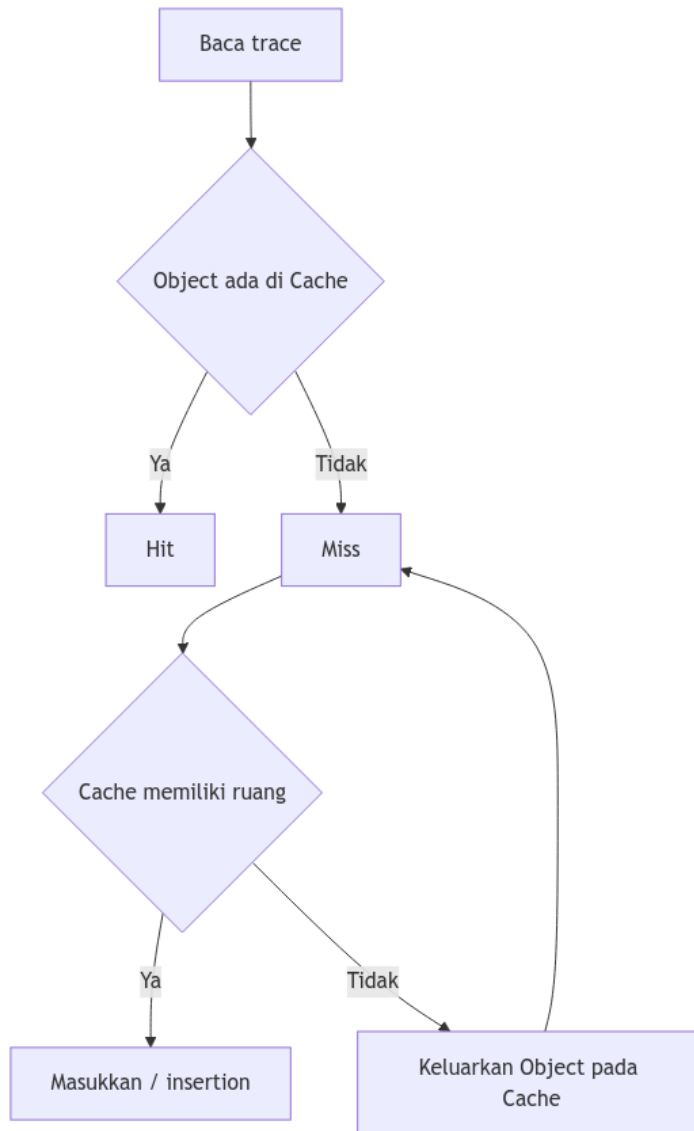
Bab ini merinci metodologi yang digunakan dalam penelitian ini, mulai dari datasets, perancangan dan pengembangan perangkat lunak simulasi, proses pengumpulan dan pengolahan data, hingga implementasi dan evaluasi model *machine learning*.

3.1 Datasets

lunak simulasi kustom yang mampu memenuhi kebutuhan analisis. Perangkat lunak ini dirancang dengan beberapa persyaratan utama:

1. **Integrasi Model Machine Learning:** Perangkat lunak harus mampu memanggil dan menjalankan model *machine learning* secara efisien di dalam fungsi *eviction* dari algoritma *cache*.
2. **Dukungan Algoritma Pembanding:** Selain algoritma yang diusulkan, perangkat lunak harus dapat menjalankan algoritma CLOCK standar dan Offline CLOCK sebagai tolok ukur perbandingan kinerja.
3. **Pengumpulan Data dari Offline CLOCK:** Perangkat lunak harus mampu menjalankan mode Offline CLOCK untuk mengumpulkan data-data krusial, antara lain:
 - (a) Nilai *miss ratio* dan jumlah promosi untuk setiap iterasi simulasi.
 - (b) Metadata dari setiap objek pada saat keputusan promosi dibuat, beserta label apakah promosi tersebut pada akhirnya sia-sia (*wasted*) atau tidak.
 - (c) Kemampuan untuk mengekspor semua data yang terkumpul ke dalam format CSV (*Comma-Separated Values*) untuk memfasilitasi proses pengolahan dan analisis data lebih lanjut.

Untuk mempercepat proses pengembangan, penulis memanfaatkan **LibCacheSim**, sebuah pustaka sumber terbuka yang dirancang khusus untuk pengembangan simulator *cache*. Pustaka ini ditulis dalam bahasa C/C++, yang menjamin kinerja komputasi yang sangat efisien, terutama untuk tugas-tugas berat seperti pengumpulan data dari *trace* berukuran besar.



Gambar 3.1: Diagram alur dari program simulasi yang dikembangkan.

3.2 Pengumpulan Data

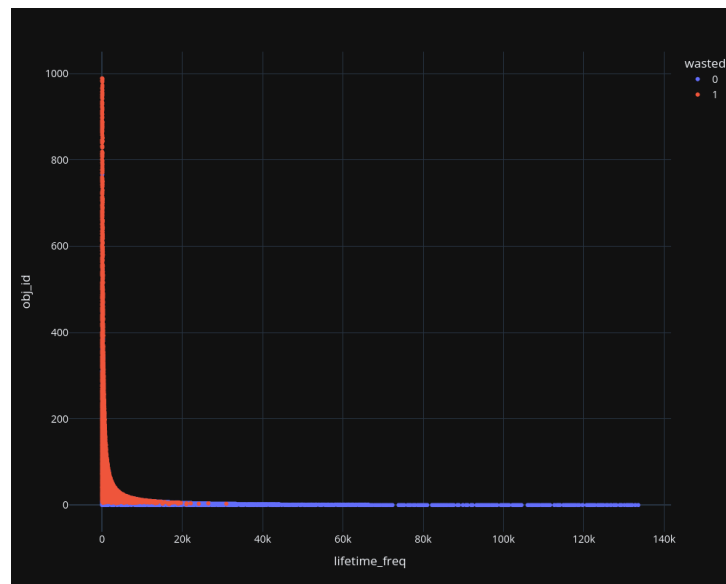
Pengumpulan data awal dilakukan sepenuhnya menggunakan perangkat lunak simulasi yang telah dikembangkan. Tujuannya adalah untuk mendapatkan metrik kinerja dasar seperti jumlah promosi dan *miss ratio* dari algoritma CLOCK standar. Selain itu, pada fase ini juga dilakukan pembuatan dataset yang akan digunakan untuk melatih model *machine learning*. Dataset ini dibuat dengan cara mencatat (*me-logging*) metadata dari setiap objek pada saat keputusan promosi akan dibuat, lalu memberinya label 'wasted' atau 'not wasted' berdasarkan informasi dari simulasi Offline CLOCK.

Trace yang digunakan dalam penelitian ini adalah *trace* sumber terbuka yang juga digunakan oleh penelitian-penelitian sebelumnya di bidang ini. *Trace* ini mencakup berbagai jenis beban kerja (*workload*) yang realistis, seperti *BlockStore*, *KVStorage*, *CDN*, *Proxy*, dan *Photo Storage*, yang berasal dari berbagai institusi teknologi terkemuka seperti Twitter, Meta, Ten-

cent, dan Alibaba. Selain itu, penulis juga menggunakan *trace* sintetis yang dibuat berdasarkan distribusi Zipfian untuk mempermudah pemahaman pola data dalam lingkungan yang lebih terkontrol.

3.3 Pengolahan dan Analisis Data

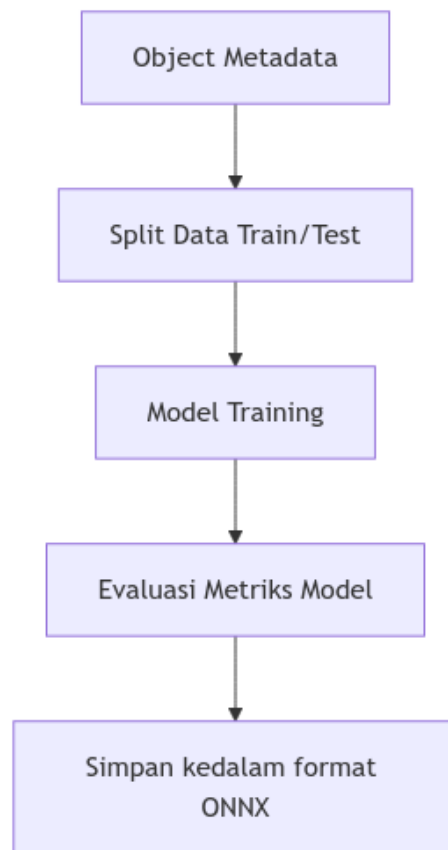
Untuk mempermudah proses analisis, penulis mengembangkan beberapa skrip Python. Skrip ini bertugas untuk mengolah data mentah dari format CSV, melakukan plotting data menggunakan pustaka seperti Plotly dan Pandas, dan menyajikannya dalam sebuah laporan HTML interaktif. Visualisasi ini sangat membantu dalam mengidentifikasi pola, korelasi, dan anomali dalam data sebelum tahap pengembangan model.



Gambar 3.2: Contoh visualisasi distribusi dari metadata `lifetime_freq`.

3.4 Pengembangan Model Machine Learning

Dengan menggunakan dataset yang telah diolah, sebuah model *machine learning* dikembangkan. Model ini dirancang untuk menerima masukan berupa metadata objek (seperti frekuensi akses, waktu sejak akses terakhir, ukuran objek, dll.) dan menghasilkan keluaran berupa probabilitas (nilai antara 0 hingga 1) yang mengindikasikan kelayakan objek tersebut untuk dipromosikan. Nilai yang tinggi menandakan bahwa objek tersebut kemungkinan besar tidak layak untuk dipromosikan.



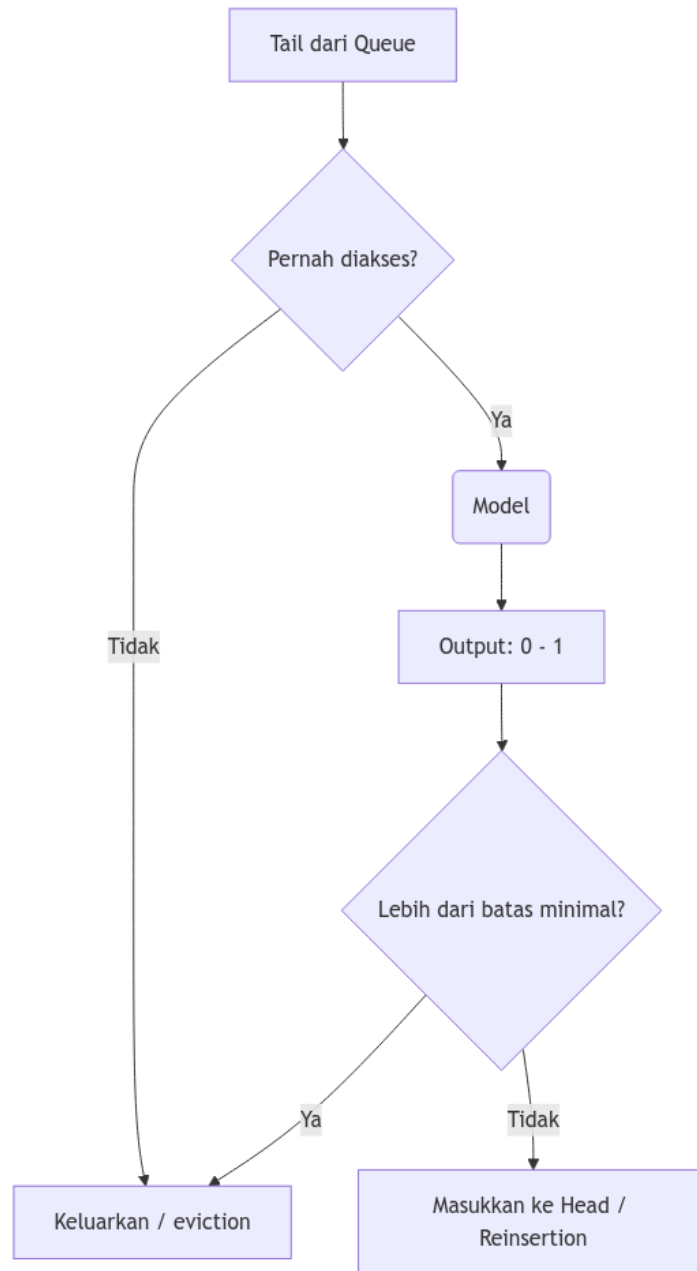
Gambar 3.3: Diagram proses pelatihan model *machine learning*.



Gambar 3.4: Arsitektur model *machine learning* yang digunakan.

3.5 Implementasi Model pada Perangkat Lunak Simulasi

Model *machine learning* yang telah dilatih kemudian diimplementasikan kembali ke dalam perangkat lunak simulasi. Untuk menjembatani antara lingkungan Python (untuk pelatihan) dan C++ (untuk simulasi), model diekspor ke dalam format **ONNX (Open Neural Network Exchange)**. Model dalam format ONNX ini kemudian dijalankan di dalam simulator menggunakan **ONNX Runtime**, sebuah *inference engine* berkinerja tinggi. Di dalam fungsi *eviction* pada algoritma CLOCK, sebelum sebuah objek dipromosikan, metadatanya akan diekstrak dan dijadikan masukan untuk model. Jika keluaran dari model melebihi ambang batas (*threshold*) yang telah ditentukan, objek tersebut akan langsung digusur alih-alih dipromosikan.



Gambar 3.5: Diagram alur implementasi model pada simulator.

3.6 Evaluasi Model

Tahap terakhir adalah evaluasi kinerja model yang telah terintegrasi. Kinerja algoritma CLOCK yang telah dimodifikasi dengan *machine learning* ini kemudian dibandingkan secara langsung dengan algoritma CLOCK standar dan tolok ukur teoretis dari Offline CLOCK. Metrik utama yang menjadi fokus perbandingan adalah jumlah total promosi dan *miss ratio*.

3.7 Perangkat Keras dan Lunak

Penelitian ini didukung oleh beberapa perangkat keras dan lunak sebagai berikut:

3.8 Perangkat Keras

1. **Testbed Utama:** Sebuah PC server dengan prosesor Intel Xeon Silver 4114 (40 *cores*) @ 3.000GHz dan RAM sebesar 192 GiB.
2. **Laptop Pribadi:** Digunakan untuk pengembangan dan analisis awal.

3.9 Perangkat Lunak

1. **Python:** Bahasa pemrograman utama untuk pengolahan data dan pengembangan model.
2. **Plotly & Pandas:** Pustaka Python untuk analisis dan visualisasi data.
3. **C++:** Bahasa pemrograman utama untuk perangkat lunak simulasi.
4. **libCacheSim:** Pustaka dasar untuk simulator *cache*.
5. **ONNX Runtime:** Untuk menjalankan inferensi model di lingkungan C++.
6. **SSH:** Untuk mengakses dan mengelola *testbed* dari jarak jauh.

BAB IV

PENGUJIAN DAN ANALISIS

Bab ini menyajikan hasil yang dicapai selama penelitian, dimulai dari hasil yang diharapkan, temuan-temuan dari hasil pengujian awal, serta analisis terhadap data tersebut untuk memvalidasi pendekatan yang diusulkan.

4.1 Hasil yang Diharapkan

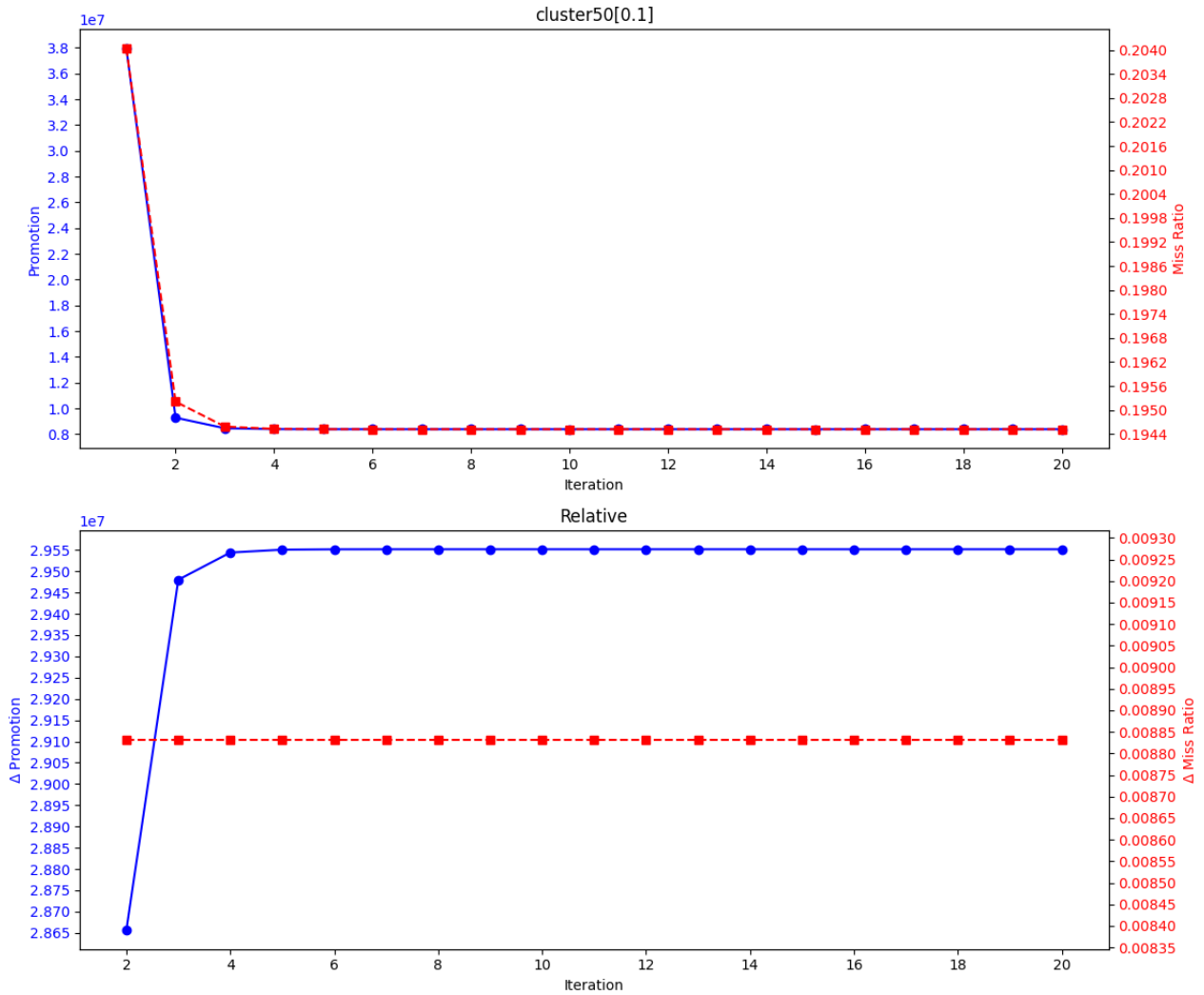
Ekspektasi utama dari penelitian ini adalah bahwa model *machine learning* yang dikembangkan akan mampu secara signifikan mengurangi jumlah total promosi objek di dalam algoritma CLOCK. Pengurangan ini diharapkan dapat dicapai tanpa mengorbankan kinerja utama *cache*, yaitu dengan tidak meningkatkan *miss ratio* secara berarti. Dengan kata lain, tujuannya adalah mencapai efisiensi manajemen *cache* yang lebih tinggi dengan membuang promosi yang tidak perlu, sambil tetap mempertahankan data yang relevan di dalam *cache*.

4.2 Hasil Pengujian Awal

Hingga tahap ini, penulis telah berhasil menyelesaikan beberapa tonggak penting. Perangkat lunak simulasi untuk akuisisi data dari *trace* telah berhasil dikembangkan. Selain itu, skrip Python untuk memproses dan memvisualisasikan data yang dihasilkan juga telah dibuat, memungkinkan analisis yang lebih cepat dan interaktif.

4.2.1 Hasil Eksperimen Offline CLOCK

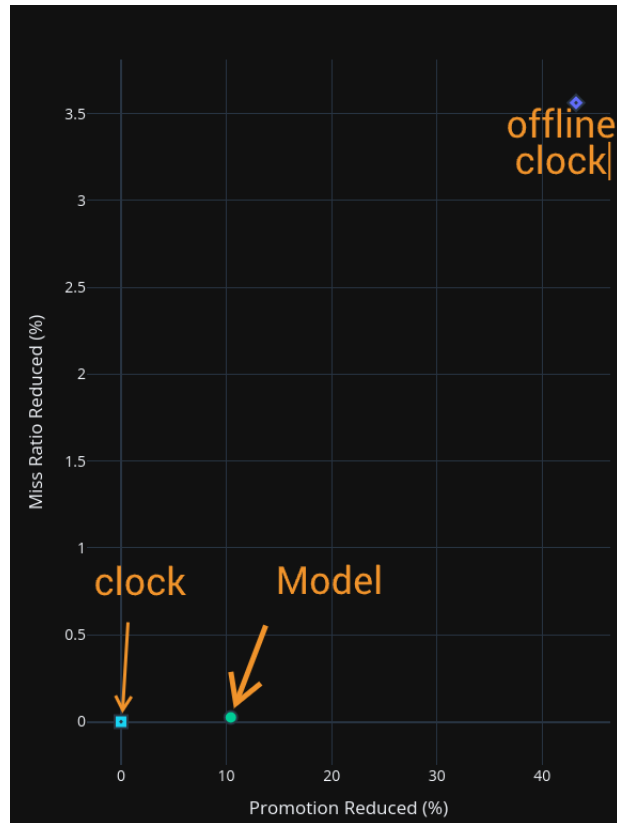
Eksperimen awal dilakukan dengan menggunakan algoritma Offline CLOCK pada *trace* Meta Key-Value untuk menetapkan batas atas kinerja. Hasil dari simulasi ini, seperti yang ditunjukkan pada Gambar 4.1, menunjukkan adanya penurunan drastis pada jumlah promosi dan juga pada *miss ratio*. Penurunan yang signifikan ini merupakan indikasi kuat bahwa pada implementasi standar algoritma CLOCK, terjadi banyak sekali promosi yang sia-sia (*wasted promotions*). Promosi ini tidak hanya membuang sumber daya komputasi, tetapi juga berpotensi "mengotori" *cache* dengan objek-objek yang sebenarnya tidak akan diakses kembali.



Gambar 4.1: Grafik hasil dari simulasi Offline CLOCK pada *trace* Meta Key-Value.

4.2.2 Hasil Pengembangan Model Awal

Pada tahap pengembangan model, penulis telah melakukan eksperimen awal dengan model regresi logistik (*logistic regression*). Model ini dipilih karena kesederhanaannya dan kemampuannya untuk diinterpretasikan. Bobot dan ambang batas (*threshold*) dari model ini telah disesuaikan untuk mendapatkan hasil yang optimal pada data validasi. Hasil dari model ini dapat dilihat pada Gambar 4.2.



Gambar 4.2: Grafik perbandingan hasil dari model regresi logistik.

Gambar 4.2 menunjukkan perbandingan rata-rata pengurangan jumlah promosi dan perubahan pada *miss ratio* setelah penerapan model. Seperti yang terlihat, jumlah promosi berhasil dikurangi secara signifikan, sekitar **10%**, sementara pada saat yang sama, *miss ratio* tidak terlalu terpengaruh dan tetap stabil. Hasil awal ini sangat menjanjikan dan selaras dengan tujuan utama penelitian, yaitu mengurangi promosi tanpa mengorbankan *hit ratio*.

[Halaman ini sengaja dikosongkan]

BAB V

PENUTUP

Bab ini merangkum keseluruhan hasil dari penelitian yang telah dilakukan. Bagian pertama, Kesimpulan, menyajikan ringkasan temuan utama, sementara bagian kedua, Saran, memberikan rekomendasi untuk pengembangan penelitian ini di masa mendatang.

5.1 Kesimpulan

Berdasarkan perancangan, implementasi, dan pengujian yang telah dilakukan dalam penelitian ini, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Integrasi model *machine learning* ke dalam algoritma *cache eviction* CLOCK terbukti dapat meningkatkan efisiensi manajemen *cache*. Model yang dikembangkan mampu mengidentifikasi dan mencegah promosi objek yang tidak perlu atau sia-sia.
2. Penerapan model regresi logistik yang telah disesuaikan berhasil mengurangi jumlah total promosi objek sekitar **10%** pada data pengujian. Pengurangan ini dicapai tanpa memberikan dampak negatif yang signifikan terhadap *miss ratio* sistem, yang berarti kinerja utama *cache* dalam melayani permintaan data tetap terjaga.
3. Penelitian ini menunjukkan bahwa bahkan model *machine learning* yang relatif sederhana dan ringan dapat memberikan peningkatan kinerja yang berarti pada algoritma *cache* klasik. Hal ini membuka peluang untuk implementasi serupa pada sistem nyata di mana *overhead* komputasi menjadi pertimbangan penting.

Secara keseluruhan, penelitian ini berhasil menunjukkan bahwa pendekatan berbasis data menggunakan *machine learning* adalah arah yang menjanjikan untuk mengoptimalkan algoritma CLOCK, menjadikannya lebih cerdas dan lebih efisien dalam mengelola sumber daya *cache*.

5.2 Saran

Meskipun penelitian ini telah mencapai tujuannya, terdapat beberapa area yang dapat dieksplorasi lebih lanjut untuk pengembangan di masa depan. Beberapa saran untuk penelitian selanjutnya antara lain:

1. **Eksplorasi Model Machine Learning yang Lebih Kompleks:** Penelitian ini menggunakan model regresi logistik. Studi di masa depan dapat mengeksplorasi penggunaan arsitektur model yang lebih canggih, seperti *neural network* yang lebih dalam atau model berbasis *ensemble* (misalnya, Gradient Boosting atau Random Forest) untuk melihat apakah peningkatan kompleksitas model dapat memberikan akurasi dan efisiensi yang lebih tinggi.
2. **Penerapan Reinforcement Learning:** Mengingat sifat dinamis dari manajemen *cache*, pendekatan menggunakan *reinforcement learning* (RL) dapat menjadi alternatif yang sangat menarik. Model RL dapat dilatih untuk membuat kebijakan *eviction* yang adaptif secara *real-time* terhadap perubahan pola beban kerja.
3. **Pengujian pada Lingkungan Produksi Nyata:** Penelitian ini terbatas pada lingkun-

gan simulasi. Langkah selanjutnya yang ideal adalah mengimplementasikan dan menguji algoritma yang diusulkan pada sistem produksi nyata untuk memvalidasi kinerjanya di bawah beban kerja yang sesungguhnya dan mengukur *overhead* inferensi model secara akurat.

4. **Analisis Fitur yang Lebih Mendalam:** Melakukan analisis yang lebih mendalam terhadap fitur-fitur (metadata objek) yang paling berpengaruh terhadap keputusan promosi dapat membantu dalam merancang model yang lebih efisien dan ringan di masa depan.

DAFTAR PUSTAKA

- [1] H. Herodotou, "AutoCache: Employing Machine Learning to Automate Caching in Distributed File Systems," 2019. doi: <https://doi.org/10.1109/icdew.2019.00-21>.
- [2] S. Sethumurugan, J. Yin, and J. Sartori, "Designing a Cost-Effective Cache Replacement Policy using Machine Learning," 2021. doi: <https://doi.org/10.1109/hpca51647.2021.00033>.
- [3] H. Sun, Q. L. Cui, J. Huang, and X. Qin, "NCache: A Machine-Learning Cache Management Scheme for Computational SSDs," 2022. doi: <https://doi.org/10.1109/tcad.2022.3208769>.
- [4] J. Shuja, K. Bilal, W. Alasmary, H. Sinky, and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge networks: A comprehensive survey," 2021. doi: <https://doi.org/10.1016/j.jnca.2021.103005>.
- [5] H. Wang, X. Yi, P. Huang, B. Cheng, and K. Zhou, "Efficient SSD Caching by Avoiding Unnecessary Writes using Machine Learning," 2018. doi: <https://doi.org/10.1145/3225058.3225126>.
- [6] D. S. Berger, "Towards Lightweight and Robust Machine Learning for CDN Caching," 2018. doi: <https://doi.org/10.1145/3286062.3286082>.
- [7] L. Breslau, P. Cao, F. Li, G. M. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," 1999. doi: <https://doi.org/10.1109/infcom.1999.749260>.
- [8] J. S. Liptay, "Structural aspects of the System/360 Model 85, II: The cache," 1968. doi: <https://doi.org/10.1147/sj.71.0015>.
- [9] A. Turing, "Lecture on the Automatic Computing Engine (1947)," 2004. doi: <https://doi.org/10.1093/oso/9780198250791.003.0015>.
- [10] M. Shima, "The Birth, Evolution and Future of the Microprocessor," 2005. doi: <https://doi.org/10.1109/cit.2005.182>.
- [11] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: comments on the history and current state," 1997. doi: <https://doi.org/10.1109/4235.585888>.
- [12] G. M. Papadopoulos and K. R. Traub, "Multithreading," 1991. doi: <https://doi.org/10.1145/115952.115986>.
- [13] M. Herlihy, "The art of multiprocessor programming," 2006. doi: <https://doi.org/10.1145/1146381.1146382>.
- [14] J. M. Borkenhagen, R. J. Eickemeyer, R. Kalla, and S. R. Kunkel, "A multithreaded PowerPC processor for commercial servers," 2000. doi: <https://doi.org/10.1147/rd.446.0885>.
- [15] A. B. Abdallah, "Advanced Multicore Systems-On-Chip," 2017. doi: <https://doi.org/10.1007/978-981-10-6092-2>.

- [16] A. C. Sodan, J. Machina, A. Deshmeh, K. Macnaughton, and B. Esbaugh, "Parallelism via Multithreaded and Multicore CPUs," 2010. doi: <https://doi.org/10.1109/mc.2010.75>.
- [17] J. Kim, S. Y. Yu, and J. Park, "Performance Evaluation of Multithreaded Computations for CPU Bounded Task," 2016. doi: <https://doi.org/10.1109/platcon.2016.7456816>.
- [18] R. J. Eickemeyer, R. E. Johnson, S. R. Kunkel, M. S. Squillante, and S. Liu, "Evaluation of multithreaded uniprocessors for commercial application environments," 1996. doi: <https://doi.org/10.1145/232973.232994>.
- [19] L. A. Belady, "A study of replacement algorithms for a virtual-storage computer," 1966. doi: <https://doi.org/10.1147/sj.52.0078>.
- [20] M. V. Wilkes, "Slave Memories and Dynamic Storage Allocation," 1965. doi: <https://doi.org/10.1109/pgec.1965.264263>.
- [21] J. Zhang, G. Wu, X. Hu, and X. Wu, "A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services," 2012. doi: <https://doi.org/10.1109/grid.2012.17>.
- [22] K. Liu, J. Peng, J. Wang, Z. Huang, and J. Pan, "Adaptive and Scalable Caching With Erasure Codes in Distributed Cloud-Edge Storage Systems," 2022. doi: <https://doi.org/10.1109/tcc.2022.3168662>.
- [23] J. Zhang, Q. Li, and W. Zhou, "HDCache: A Distributed Cache System for Real-Time Cloud Services," 2016. doi: <https://doi.org/10.1007/s10723-015-9360-9>.
- [24] D. Eklöv, D. Black-Schaffer, and E. Hägersten, "Fast modeling of shared caches in multicore systems," 2011. doi: <https://doi.org/10.1145/1944862.1944885>.
- [25] J. M. Calandrino and J. H. Anderson, "On the Design and Implementation of a Cache-Aware Multicore Real-Time Scheduler," 2009. doi: <https://doi.org/10.1109/ecrts.2009.13>.
- [26] S. Seo, J. Lee, and Z. Sura, "Design and implementation of software-managed caches for multicores with local memory," 2009. doi: <https://doi.org/10.1109/hpca.2009.4798237>.
- [27] X. Vera, B. Lisper, and J. Xue, "Data cache locking for higher program predictability," 2003. doi: <https://doi.org/10.1145/781027.781062>.
- [28] W. Hu, W. Shi, Z. Tang, and M. Li, "A lock-based cache coherence protocol for scope consistency," 1998. doi: <https://doi.org/10.1007/bf02946599>.
- [29] C. Mohan and I. Narang, "Efficient locking and caching of data in the multisystem shared disks transaction environment," 2005. doi: <https://doi.org/10.1007/bfb0032448>.
- [30] C. Fricker, P. Robert, and J. W. Roberts, "A versatile and accurate approximation for LRU cache performance," 2012. doi: <https://doi.org/10.48550/arxiv.1202.3974>.
- [31] E. O'Neil, P. O'Neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," 1993. doi: <https://doi.org/10.1145/170036.170081>.
- [32] E. O'Neil, P. E. O'Neil, and G. Weikum, "An optimality proof of the LRU-K page replacement algorithm," 1999. doi: <https://doi.org/10.1145/300515.300518>.

- [33] A. Dan and D. Towsley, “An approximate analysis of the LRU and FIFO buffer replacement schemes,” 1990. doi: <https://doi.org/10.1145/98457.98525>.
- [34] M. Moir, D. A. Nussbaum, O. Shalev, and N. Shavit, “Using elimination to implement scalable and lock-free FIFO queues,” 2005. doi: <https://doi.org/10.1145/1073970.1074013>.
- [35] C. E. Molnar, I. W. Jones, W. Coates, and J. Lexau, “A FIFO ring performance experiment,” 2002. doi: <https://doi.org/10.1109/async.1997.587181>.
- [36] Q. Chen, M. H. M. Al-Araby, Z. Qiu, Z. Chen, R. Vinayak, and J. Yang, “Demystifying and improving lazy promotion in cache eviction,” To appear in VLDB 2026.
- [37] G. Eads, “NbQ-CLOCK: A Non-blocking Queue-based CLOCK Algorithm for Web-Object Caching,” 2014. doi: <https://doi.org/10.13140/2.1.2919.2329>.
- [38] P. Cunningham, M. Cord, and S. J. Delany, “Supervised Learning,” 2008. doi: https://doi.org/10.1007/978-3-540-75171-7_2.
- [39] O. Chapelle, B. Schölkopf, and A. Zien, “Semi-Supervised Learning,” 2006. doi: <https://doi.org/10.7551/mitpress/9780262033589.001.0001>.
- [40] J. M. Hilbe, “Logistic Regression Models,” 2009. doi: <https://doi.org/10.1201/9781420075779>.
- [41] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, “Applied Logistic Regression,” 2013. doi: <https://doi.org/10.1002/9781118548387>.
- [42] R. S. Sutton and A. Barto, “Reinforcement Learning: An Introduction,” 2005. doi: <https://doi.org/10.1109/tnn.2004.842673>. [Online]. Available: <https://doi.org/10.1109/tnn.2004.842673>.
- [43] C. Tang, “Cache system design in the tightly coupled multiprocessor system,” 1976. doi: <https://doi.org/10.1145/1499799.1499901>.
- [44] S. Bansal and D. S. Modha, “CAR: Clock with Adaptive Replacement,” 2004. doi: None. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.138.583>.
- [45] S. Jiang, F. Chen, and X. Zhang, “CLOCK-Pro: an effective improvement of the CLOCK replacement,” 2005. doi: None. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.4250>.
- [46] J. Yang, Y. Zhang, Z. Qiu, Y. Yue, and R. Vinayak, “FIFO queues are all you need for cache eviction,” 2023. doi: <https://doi.org/10.1145/3600006.3613147>. [Online]. Available: <https://doi.org/10.1145/3600006.3613147>.
- [47] J. Yang, Y. Yue, and K. V. Rashmi, “A Large-scale Analysis of Hundreds of In-memory Key-value Cache Clusters at Twitter,” 2021. doi: <https://doi.org/10.1145/3468521>. [Online]. Available: <https://doi.org/10.1145/3468521>.
- [48] W.-F. Lin et al., “ONNC: A Compilation Framework Connecting ONNX to Proprietary Deep Learning Accelerators,” 2019. doi: <https://doi.org/10.1109/aicas.2019.8771510>. [Online]. Available: <https://doi.org/10.1109/aicas.2019.8771510>.
- [49] J. Tian et al., “Compiling ONNX Neural Network Models Using MLIR,” 2020. doi: <https://doi.org/10.48550/arxiv.2008.08272>. [Online]. Available: <http://arxiv.org/abs/2008.08272>.

- [50] P. Jajal et al., “Interoperability in Deep Learning: A User Survey and Failure Analysis of ONNX Model Converters,” 2024. doi: <https://doi.org/10.1145/3650212.3680374>. [Online]. Available: <https://doi.org/10.1145/3650212.3680374>.
- [51] M. Sever and S. Ogut, “A Performance Study Depending on Execution Times of Various Frameworks in Machine Learning Inference,” 2021. doi: <https://doi.org/10.1109/uys54260.2021.9659677>. [Online]. Available: <https://doi.org/10.1109/uys54260.2021.9659677>.

BIOGRAFI PENULIS



Penulis, **Muhammad Haekal Muhyidin Al-Araby**, lahir di [Tempat Lahir] pada tanggal [Tanggal Lahir]. Penulis merupakan anak dari pasangan [Nama Ayah] dan [Nama Ibu].

Penulis menempuh pendidikan dasar di [Nama SD] dan lulus pada tahun [Tahun Lulus SD]. Kemudian, penulis melanjutkan pendidikan menengah pertama di [Nama SMP] dan lulus pada tahun [Tahun Lulus SMP]. Setelah itu, penulis menyelesaikan pendidikan menengah atas di [Nama SMA/SMK] pada tahun [Tahun Lulus SMA/SMK].

Pada tahun [Tahun Masuk Kuliah], penulis diterima sebagai mahasiswa di Program Studi Teknik Komputer, Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember (ITS) melalui jalur [Jalur Masuk, contoh: SNMPTN/SBMPTN/Mandiri].

Selama masa perkuliahan, penulis memiliki minat khusus pada bidang [Sebutkan Minat Bidang Anda, contoh: rekayasa perangkat lunak, kecerdasan buatan, sistem komputer]. Untuk memperdalam minat tersebut, penulis aktif dalam [Sebutkan kegiatan, contoh: penelitian di laboratorium, organisasi, atau proyek pribadi].

Untuk memenuhi salah satu syarat kelulusan dan memperoleh gelar Sarjana Teknik, penulis menyusun Tugas Akhir yang berjudul "" di bawah bimbingan Bapak Reza Fuad Rachmadi, S.T., M.T., Ph.D dan Bapak Dr. Arief Kurniawan, S.T., M.T.

Penulis dapat dihubungi melalui surel di [Alamat Email Anda]. Email Anda].

[Halaman ini sengaja dikosongkan]