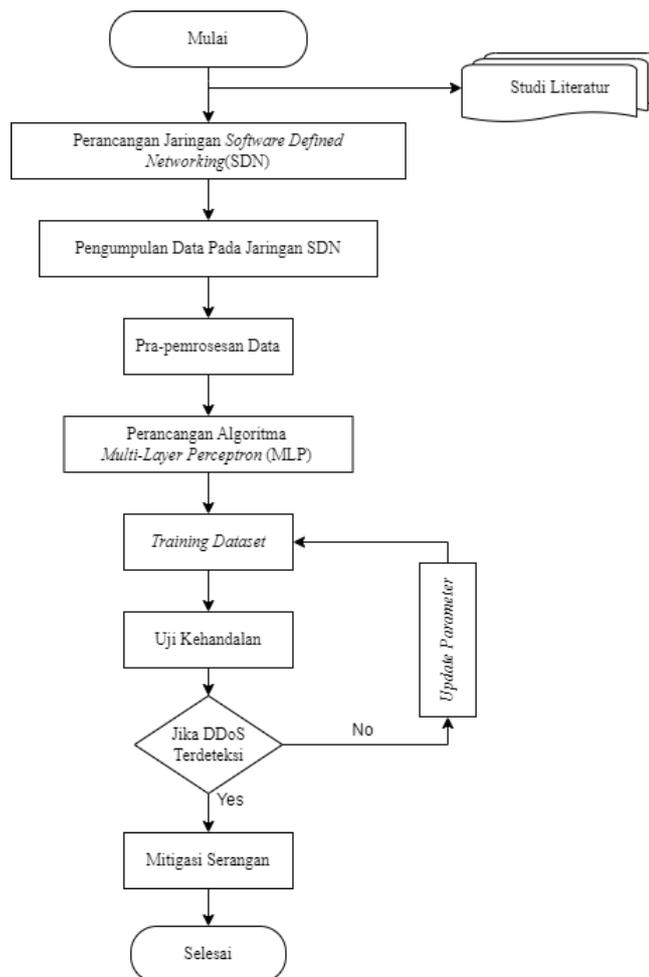


BAB III METODOLOGI PENELITIAN

Metodologi penelitian ini bertujuan untuk merancang algoritma *Multi-Layer Perceptron* (MLP) sebagai sistem pendeteksi serangan DDoS menggunakan metode *deep learning*. Selain itu, perancangan sistem *Software-Defined Network* (SDN) juga akan menjadi komponen penting dalam pengembangan sistem pendeteksi serangan DDoS.

3.1. Alur Penelitian

Penelitian ini melibatkan beberapa tahapan proses untuk mencapai hasil yang diharapkan, di antaranya adalah sebagai berikut.



Gambar 3.1 Diagram alir penelitian

3.2. Komponen Penelitian

Berikut adalah komponen yang digunakan dalam penelitian ini:

3.2.1 Python

Python adalah bahasa pemrograman yang sering digunakan untuk melakukan pengolahan data, termasuk dalam pembuatan topologi jaringan serta mengatur alur dari data *flow*. Python juga dapat digunakan untuk melakukan berbagai tugas lainnya seperti analisis data, atau pembuatan model *machine learning*.

3.2.2 Sistem Operasi Ubuntu 22.04

Ubuntu 22.04 adalah distribusi Linux berbasis Debian yang digunakan sebagai lingkungan pengembangan dan eksekusi untuk komponen-komponen penelitian ini. Ubuntu dipilih karena stabilitasnya, serta menyediakan lingkungan yang ideal untuk pengembangan jaringan dan pengujian perangkat lunak.

3.2.3 Traffic Generator

Traffic generator digunakan untuk menghasilkan lalu lintas jaringan yang digunakan dalam pengujian dan simulasi seperti *ping*, *iPerf*, dan *hping3*. Alat ini memungkinkan pembuatan skenario lalu lintas yang berbeda, termasuk lalu lintas normal dan serangan DDoS, untuk menguji efektivitas sistem deteksi dan mitigasi.

3.2.4 Wireshark

Wireshark adalah alat analisis protokol jaringan yang digunakan untuk menangkap dan memeriksa paket data yang dikirim melalui jaringan. Dengan Wireshark, peneliti dapat menganalisis lalu lintas jaringan secara mendalam, mengidentifikasi pola serangan, dan memverifikasi bahwa mitigasi DDoS bekerja dengan baik.

3.2.5 S-flow RT

sFlow RT adalah alat monitoring jaringan *real-time* yang digunakan untuk mengumpulkan dan menganalisis statistik lalu lintas jaringan. sFlow RT memungkinkan deteksi anomali dalam lalu lintas jaringan dan memberikan data yang diperlukan untuk mendeteksi serangan DDoS.

3.2.6 Jupyter Notebook

Jupyter Notebook adalah aplikasi web *open-source* yang digunakan untuk membuat dan berbagi dokumen yang berisi kode, persamaan, visualisasi, dan narasi teks. Jupyter Notebook pada penelitian ini digunakan dalam melakukan eksplorasi data, analisis data, pembuatan model *machine learning*, dan masih banyak lagi.

3.2.7 Mininet

Mininet adalah emulator jaringan *open-source* yang memungkinkan pengguna untuk membuat jaringan virtual untuk pengujian dan eksperimen. Salah satu fitur utama Mininet adalah kemampuannya membuat topologi jaringan dengan *switch* virtual, *host*, dan koneksi. Selain itu, Mininet terintegrasi dengan *OpenFlow*, protokol untuk jaringan SDN. Integrasi Mininet dengan *OpenFlow* memungkinkan pengguna membuat dan menguji arsitektur dan aplikasi jaringan berbasis SDN.

3.2.8 Ryu controller

Ryu *controller* adalah sebuah platform pengembangan perangkat lunak berbasis *OpenFlow* yang digunakan untuk mengimplementasikan jaringan SDN. Ryu menyediakan berbagai fitur dan fungsionalitas yang memungkinkan pengguna untuk mengendalikan dan mengelola jaringan dengan cara yang lebih fleksibel dan dinamis. Dengan menggunakan protokol *OpenFlow*, Ryu *controller* memperbolehkan pengguna untuk secara sentral mengontrol aliran lalu lintas jaringan dan mengatur kebijakan jaringan secara dinamis. Dengan kombinasi antara Mininet sebagai emulator jaringan dan Ryu *controller* sebagai SDN, pengguna dapat melakukan simulasi dan pengujian jaringan SDN secara efektif. Pengguna dapat membuat topologi jaringan yang kompleks di Mininet dan mengontrolnya melalui Ryu *controller* sehingga memungkinkan pengguna untuk mengembangkan dan menguji berbagai aplikasi dan kebijakan jaringan SDN.

3.2.9 Hardware

Penelitian ini menggunakan sebuah komputer yang dilengkapi dengan spesifikasi yang memadai untuk melakukan training model *Multi-Layer Perceptron* (MLP). Selain itu, juga digunakan sebuah mesin virtual yang berfungsi untuk menjalankan simulasi jaringan dengan tujuan memperoleh hasil yang akurat dan representatif. Dengan menggunakan komputer dan mesin virtual ini, penelitian

dapat dilakukan dengan lebih efektif dan efisien, serta menghasilkan hasil yang dapat diandalkan. Berikut merupakan spesifikasi komputer yang di gunakan:

Tabel 3.1 Tabel spesifikasi *hardware*

Komponen	Spesifikasi Asli	Spesifikasi Mesin Virtual
Processor	AMD Ryzen 9 4900HS (8 Core)	AMD Ryzen 9 4900HS (2 Core)
RAM	16 GB DDR4	4 GB DDR4
Storage	1 TB SSD	40 GB SSD
GPU	NVIDIA GeForce RTX 2060 Max-Q	AMD Radeon Graphics
Sistem operasi	Windows 11	Ubuntu 22.4

Tabel 3.1 menjelaskan spesifikasi komputer yang digunakan dalam penelitian ini. Terdapat dua komponen yang dijelaskan, yaitu spesifikasi asli untuk komputer utama yang digunakan untuk melatih model MLP, dan spesifikasi mesin virtual untuk mesin virtual yang digunakan untuk menjalankan simulasi jaringan. *Hardware* tersebut digunakan untuk menjalankan proses-proses komputasi yang membutuhkan daya komputasi yang tinggi. Dengan menggunakan komputer utama yang kuat dan mesin virtual yang memadai, penelitian ini dapat dilakukan dengan efektif dan efisien. Kombinasi spesifikasi yang tepat memungkinkan peneliti untuk melatih model MLP dengan cepat dan menjalankan simulasi jaringan dengan akurasi yang tinggi.

3.3. Metode Penelitian

Penelitian ini memiliki beberapa tahapan proses untuk mencapai hasil yang diharapkan, diantaranya adalah sebagai berikut.

3.3.1 Studi Literatur

Studi literatur merupakan tahap awal dari metodologi penelitian yang dilakukan. Pada tahap ini, peneliti melakukan pencarian informasi mengenai serangan DDoS pada jaringan SDN, serta metode *deep learning* dengan algoritma MLP. Proses ini mencakup pembacaan dan pengumpulan informasi dari berbagai sumber seperti jurnal ilmiah, buku, dan artikel online. Setelah informasi terkumpul, peneliti melakukan analisis dan evaluasi terhadap informasi tersebut untuk memastikan bahwa sumbernya terpercaya dan relevan dengan topik penelitian. Selain itu, peneliti juga dapat mengidentifikasi kekurangan atau kesenjangan dalam penelitian terdahulu dan menemukan potensi untuk penelitian lebih lanjut pada

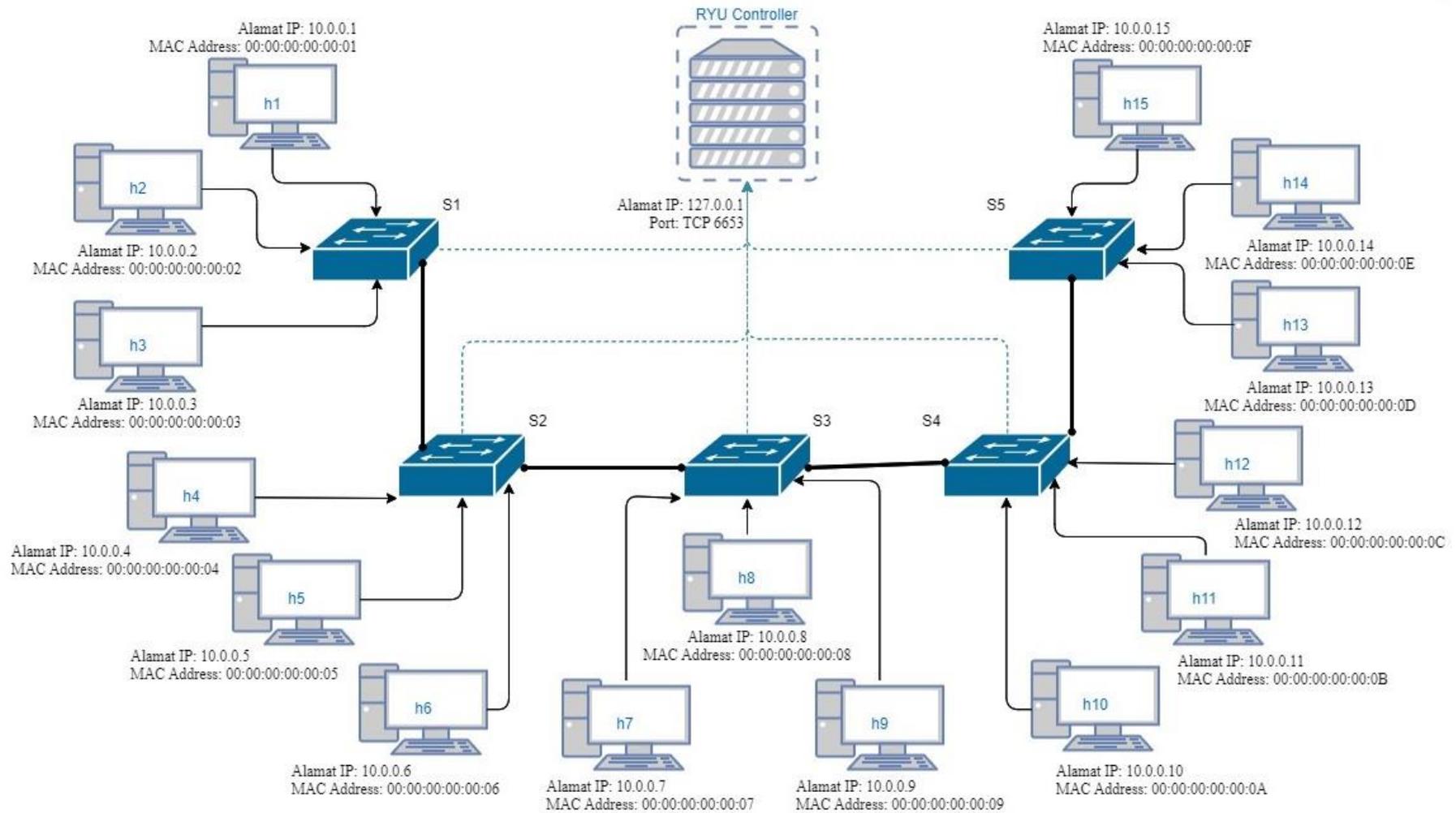
topik yang sama. Dalam hal ini, peneliti dapat mengeksplorasi faktor-faktor lain yang mempengaruhi serangan DDoS, mengevaluasi metode *deep learning* lainnya, atau mempelajari aplikasi lain dari metode SDN.

3.3.2 Perancangan Topologi *Software Defined Network* (SDN)

Perancangan topologi SDN dengan Mininet dan Ryu merujuk pada proses merancang atau membuat struktur jaringan menggunakan konsep *Software-Defined Network* (SDN). Dengan melakukan perancangan topologi SDN, peneliti dapat memastikan bahwa jaringan SDN yang diimplementasikan dapat beroperasi dengan efisien, fleksibel, dan mudah untuk di analisa. Perancangan topologi SDN melibatkan merancang dan mengatur struktur jaringan yang memisahkan lapisan kontrol dari lapisan penerusan.

Mininet berfungsi sebagai emulator jaringan untuk membuat topologi jaringan virtual, sementara Ryu digunakan sebagai kerangka kerja perangkat lunak untuk mengembangkan aplikasi kontroler SDN. Topologi terdiri dari 5 *switch* yang dihubungkan secara linear, di mana setiap *switch* terhubung ke 3 *host*, menghasilkan total 15 *host* dalam jaringan. Masing-masing *switch* dihubungkan satu sama lain, membentuk rantai linear yang memungkinkan pengujian efektivitas routing dan pengelolaan aliran data di jaringan. Setiap *host* memiliki alamat IP dan *MAC address* yang berurutan, mulai dari h1 hingga h15.

Pengaturan ini memudahkan manajemen dan pemetaan *host* di dalam jaringan serta membantu dalam proses *troubleshooting*. Setiap *switch* terhubung dengan controller SDN melalui koneksi virtual, dan koneksi ini memanfaatkan protokol *OpenFlow* 1.3, yang memberikan fleksibilitas dalam pengaturan aliran data serta mendukung berbagai fitur canggih. Pemantauan lalu lintas jaringan dilakukan melalui controller yang beroperasi pada *port* 6653, memungkinkan peneliti untuk melihat dan menganalisis aliran data secara *real-time*. Topologi ini dirancang untuk mendukung eksperimen dan pengujian berbagai skenario jaringan, dengan fokus pada pengujian performa jaringan SDN, analisis serangan DDoS, dan pengembangan metode mitigasi yang efektif. Berikut adalah topologi yang digunakan dalam penelitian ini.



Gambar 3.2 Topologi jaringan SDN

Gambar 3.2 menunjukkan topologi jaringan SDN yang digunakan untuk pembuatan *dataset* dan simulasi deteksi serangan DDoS. Lingkungan simulasi jaringan ini dibuat di dalam, sistem operasi Ubuntu 22.4 yang berperan sebagai platform penyedia infrastruktur dan sumber daya untuk menjalankan simulasi. Berikut merupakan tabel konfigurasi jaringan yang di gunakan dalam penelitian.

Tabel 3.2 Tabel konfigurasi SDN

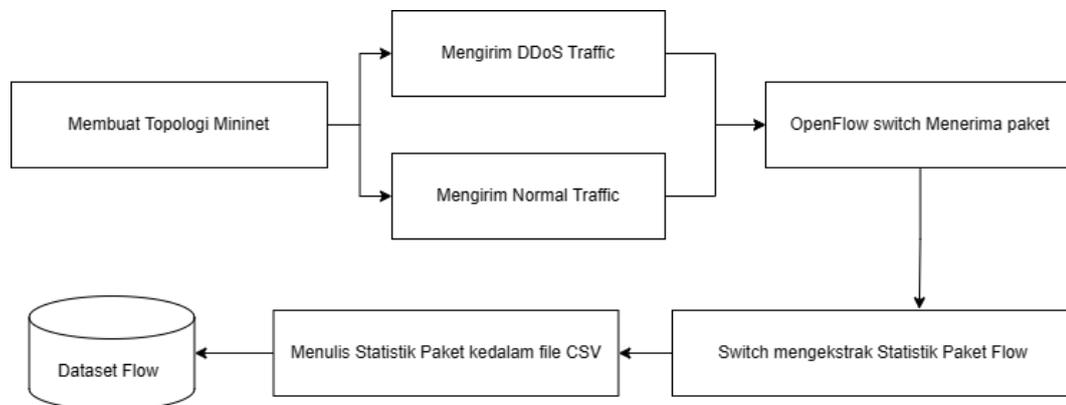
Detail Konfigurasi Node		
<i>Switches:</i>		
1.	s1: Terhubung ke	h1, h2, h3, s2
2.	s2: Terhubung ke	h4, h5, h6, s1, s3
3.	s3: Terhubung ke	h7, h8, h9, s2, s4
4.	s4: Terhubung ke	h10, h11, h12, s3, s5
5.	s5: Terhubung ke	h13, h14, h15, s4
<i>Hosts:</i>		
1.	h1	- IP: 10.0.0.1/24, - MAC: 00:00:00:00:00:01, - Terhubung ke: s1
2.	h2	- IP: 10.0.0.2/24, - MAC: 00:00:00:00:00:02, - Terhubung ke: s1
3.	h3	- IP: 10.0.0.3/24, - MAC: 00:00:00:00:00:03, - Terhubung ke: s1
4.	h4	- IP: 10.0.0.4/24, - MAC: 00:00:00:00:00:04, - Terhubung ke: s2
5.	h5	- IP: 10.0.0.5/24, - MAC: 00:00:00:00:00:05, - Terhubung ke: s2
6.	h6	- IP: 10.0.0.6/24, - MAC: 00:00:00:00:00:06, - Terhubung ke: s2
7.	h7	- IP: 10.0.0.7/24, - MAC: 00:00:00:00:00:07, - Terhubung ke: s3
8.	h8	- IP: 10.0.0.8/24, - MAC: 00:00:00:00:00:08, - Terhubung ke: s3
9.	h9	- IP: 10.0.0.9/24, - MAC: 00:00:00:00:00:09, - Terhubung ke: s3
10.	h10	- IP: 10.0.0.10/24, - MAC: 00:00:00:00:00:0A, - Terhubung ke: s4
11.	h11	- IP: 10.0.0.11/24, - MAC: 00:00:00:00:00:0B, - Terhubung ke: s4
12.	h12	- IP: 10.0.0.12/24,

		- MAC: 00:00:00:00:00:0C, - Terhubung ke: s4
13.	h13	- IP: 10.0.0.13/24, - MAC: 00:00:00:00:00:0D, - Terhubung ke: s5
14.	h14	- IP: 10.0.0.14/24, - MAC: 00:00:00:00:00:0E, - Terhubung ke: s5
15.	h15	- IP: 10.0.0.15/24, - MAC: 00:00:00:00:00:0F, - Terhubung ke: s5
Monitoring dan <i>controller</i> :		
1.	sFlow:	Fungsi <i>wrapper</i> dari <i>sflow.py</i> digunakan untuk monitoring jaringan.
2.	<i>Controller</i>	- IP: 127.0.0.1 - Port: 6653 - Protokol: OpenFlow13 - Menggunakan <i>remote controller</i> di Mininet

Secara keseluruhan, topologi SDN ini menciptakan lingkungan jaringan yang terpusat dan terkontrol dengan fleksibilitas, efisiensi, dan kemampuan adaptasi yang tinggi dalam pengaturan dan manajemen jaringan, serta memberikan fondasi yang kuat untuk penelitian dan eksperimen dalam bidang keamanan jaringan dan deteksi intrusi.

3.3.3 Akuisisi Data

Pembuatan *dataset* adalah proses pengumpulan, pengolahan, dan penyusunan data dalam sebuah format yang dapat digunakan untuk analisis. *Dataset* yang baik harus memiliki keragaman data yang mencakup berbagai variabel dan kategori. Berikut merupakan proses pembuatan *dataset*.



Gambar 3.3 Blok diagram pembuatan *dataset*

Gambar 3.3 menggambarkan blok diagram proses yang diterapkan dalam langkah-langkah pembuatan *dataset*. Untuk membuat *dataset* pada jaringan SDN, langkah-langkah berikut perlu dilakukan. Pertama, kita perlu membuat topologi jaringan yang diinginkan menggunakan Mininet, di mana jaringan virtual dibuat sesuai dengan kebutuhan. Setelah itu, pada langkah kedua, dilakukan pengiriman paket data normal dan DDoS ke jaringan SDN dengan menggunakan alat seperti, *ping*, *iperf*, dan *hping*. Langkah ketiga melibatkan *openflow switch* sebagai penerima paket data dalam jaringan tersebut. Selanjutnya, pada langkah keempat, *switch* melakukan ekstraksi statistik dari aliran paket yang diterima. Hasil ekstraksi statistik tersebut, disimpan oleh sistem ke dalam file berformat CSV, membentuk *datasetflow.csv* yang nantinya dapat digunakan untuk analisis lebih lanjut terkait aliran data dalam jaringan. Cara diatas juga di gunakan sebagai bahan pengumpulan *dataset* yang akan di gunakan sebagai bahan prediksi untuk sistem pendeteksi serangan DDoS.

Penelitian ini menggunakan *dataset* yang terdiri dari 22 fitur yang umumnya digunakan untuk menggambarkan aliran data dalam jaringan yang menggunakan protokol *OpenFlow*. *Dataset* ini memberikan informasi yang cukup lengkap dan mendalam tentang karakteristik aliran data yang berjalan melalui jaringan. Dengan menggunakan *dataset* ini, penelitian dapat memberikan pemahaman yang lebih komprehensif tentang penggunaan protokol *OpenFlow* dalam mengelola aliran data dalam jaringan. Penjelasan dari masing - masing fitur di gambarkan pada tabel 3.2 berikut ini.

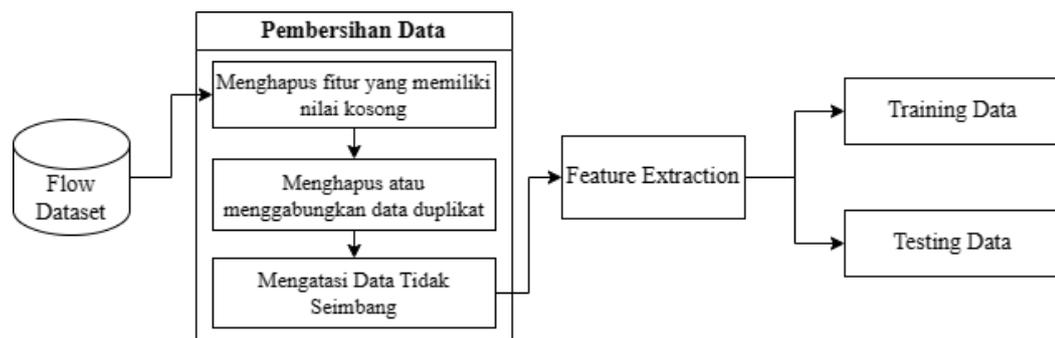
Tabel 3.3 Tabel parameter lingkungan simulasi

No	Fitur pada <i>dataset</i>	Deskripsi
1	Timestamp	Waktu saat aliran data dimulai atau diakhiri
2	<i>datapath_id</i>	Pengenal unik dari <i>switch OpenFlow</i> yang terlibat dalam aliran data
3	<i>flow_id</i>	Pengenal unik dari aliran data itu sendiri
4	<i>ip_src</i>	Alamat IP Sumber
5	<i>tp_src</i>	<i>Port</i> sumber
6	<i>ip_dst</i>	Alamat IP tujuan
7	<i>tp_dst</i>	<i>Port</i> tujuan
8	<i>ip_proto</i>	Protokol IP (TCP, UDP, ICMP)
9	<i>icmp_code</i>	Kode ICMP yang digunakan oleh paket data dalam aliran
10	<i>icmp_type</i>	Tipe ICMP yang digunakan

11	<i>flow_duration_sec</i>	Durasi aliran data dalam detik
12	<i>flow_duration_nsec</i>	Durasi aliran data dalam <i>nanodetik</i>
13	<i>idle_timeout</i>	Waktu maksimum aliran data tidak aktif sebelum dihapus
14	<i>hard_timeout</i>	Waktu maksimum aliran data tetap ada sebelum dihapus
15	<i>flags</i>	Tanda status atau sifat aliran data
16	<i>packet_count</i>	Jumlah paket data dalam aliran
17	<i>byte_count</i>	Jumlah byte data dalam aliran
18	<i>packet count per second</i>	Rata-rata jumlah paket data per detik
19	<i>packet count per nsecond</i>	Rata-rata jumlah paket data per <i>nanodetik</i>
20	<i>byte_count per second</i>	Rata-rata jumlah byte data per detik
21	<i>byte_count per nsecond</i>	Rata-rata jumlah byte data per <i>nanodetik</i>

3.3.4 Pra-pemrosesan data

Tujuan dari pra-pemrosesan data adalah untuk mengubah data mentah menjadi bentuk yang berguna dan mudah dalam analisis. Berikut adalah gambaran alur dari pra-pemrosesan data.



Gambar 3.4 Alur pra-pemrosesan data

Gambar 3.4 merupakan *flowchart* prosedur pra-pemrosesan dataset dalam konteks jaringan lalu lintas Software-Defined Network (SDN). Langkah pertama adalah menghilangkan fitur yang memiliki nilai kosong, yang penting untuk menghindari bias dalam analisis karena informasi yang tidak tercatat atau tidak tersedia. Selanjutnya, data duplikat dihapus atau digabungkan untuk mencegah distorsi dalam pola yang dianalisis, mengingat bahwa dalam lalu lintas jaringan, data duplikat mungkin terjadi akibat pengumpulan informasi yang sama berulang kali. Langkah ketiga adalah menangani data yang tidak seimbang, ini sering terjadi ketika sampel dari kelas berbeda tidak merata, seperti lebih banyak contoh lalu lintas normal dibandingkan serangan jaringan, yang dapat mengganggu model pembelajaran mesin dalam mendeteksi serangan. Untuk mengatasi ini, diterapkan

teknik seperti *oversampling* atau *undersampling*. Berikutnya, fitur diekstraksi dari data mentah untuk mengidentifikasi pola lalu lintas normal dan anomali, seperti durasi aliran rata-rata atau ukuran paket. Setelah itu, 80% dari dataset yang telah diproses digunakan sebagai data latihan untuk model pembelajaran mesin, sedangkan 20% sisanya disimpan sebagai data pengujian untuk mengevaluasi kinerja model terhadap data baru. Seluruh proses ini esensial untuk memastikan bahwa model yang dilatih memiliki kemampuan untuk memprediksi dan menggeneralisasi dengan baik saat diterapkan dalam skenario nyata.

3.3.5 Perancangan Algoritma

Perancangan *deep learning* dengan arsitektur *Multi Layer Perceptron* (MLP) untuk mengklasifikasikan serangan DDoS merupakan langkah awal dalam merancang suatu sistem yang dapat secara efektif membedakan dan mengidentifikasi pola serangan yang mungkin terjadi pada jaringan. Perancangan algoritma atau arsitektur MLP ini memainkan peran kunci dalam memberikan kecerdasan buatan yang dapat mengenali tanda-tanda dan perilaku karakteristik serangan DDoS sehingga memungkinkan sistem untuk memberikan respon yang cepat dan efisien dalam melindungi integritas dan ketersediaan layanan jaringan.

Tabel 3.4 Tabel arsitektur MLP

Models	Hyperparameters
MLP	Layer : 2 layer (256 dan 128 neuron)
	Activation : relu
	Regulasi L2 : 0.0001
	Random state : 42
	Solver : adam
	Learning rate : 0.001

Tabel 3.3 merupakan rancangan arsitektur MLP yang digunakan dalam penelitian ini. Dalam rancangan arsitektur MLP ini, terdapat dua lapisan tersembunyi dengan jumlah *neuron* yang ditentukan. Lapisan tersembunyi bertugas untuk mengekstraksi fitur dari data masukan, sedangkan lapisan keluaran bertugas untuk menghasilkan prediksi atau klasifikasi dari data. Pada setiap *neuron* dalam MLP, digunakan *rectified linear unit* untuk memperoleh hasil yang lebih baik.

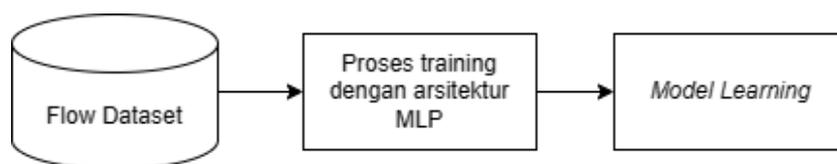
Untuk menghindari *overfitting*, digunakan regulasi L2 dengan nilai 0.0001. Regulasi L2 adalah teknik yang digunakan untuk mengurangi kompleksitas model

dengan menghukum bobot parameter yang besar dan kompleks pada model. Selain itu, rancangan ini menggunakan algoritma optimasi adam dengan laju pembelajaran sebesar 0.001. Algoritma optimasi adam adalah algoritma yang adaptif dan dapat meningkatkan kinerja MLP dengan mengatasi masalah gradien yang meledak atau stagnan.

Rancangan arsitektur MLP ini memiliki spesifikasi yang dapat diimplementasikan dalam penelitian untuk mengklasifikasikan serangan DDoS. Dengan menggunakan rancangan ini, diharapkan dapat memperoleh hasil yang akurat dan efektif dalam melindungi integritas dan ketersediaan layanan jaringan.

3.3.6 Training Dataset

Berikut adalah gambaran alur pelatihan yang bertujuan untuk melatih model agar mampu membedakan antara lalu lintas jaringan normal dan serangan DDoS.



Gambar 3.5 Alur proses training data

Gambar 3.5 yang terlampir di bawah ini menjelaskan secara rinci alur dari proses pelatihan model. Pada tahap awal, digunakanlah *dataset* yang berisi data jaringan yang telah diekstrak dari jaringan SDN. *Dataset* ini mencakup berbagai jenis serangan DDoS dan juga lalu lintas normal. Proses pelatihan dimulai dengan melakukan impor *dataset* yang telah disiapkan sebelumnya. Selanjutnya, algoritma *Multilayer Perceptron* (MLP) digunakan untuk melatih model yang telah disiapkan sebelumnya. Proses pelatihan dilakukan dalam beberapa iterasi, dimana setiap iterasi bertujuan untuk meningkatkan akurasi model yang sedang dilatih. Selama proses pelatihan berlangsung, model yang sedang dilatih dievaluasi secara berkala untuk memastikan bahwa kinerjanya sesuai dengan harapan.

3.3.7 Uji Keandalan Model

Setelah model berhasil dilatih, langkah selanjutnya adalah mengevaluasi performa dari model yang telah dibuat. Evaluasi ini dilakukan untuk mengetahui seberapa akurat dan handal model dalam membedakan antara serangan DDoS dan

lalu lintas jaringan normal. Evaluasi model dilakukan dengan data yang belum digunakan pada proses *training*.

Setelah metrik evaluasi dihitung, hasilnya dianalisis untuk mendapatkan kinerja model. Akurasi memberikan gambaran umum tentang seberapa sering model membuat prediksi yang benar. Presisi dan *recall* menunjukkan seberapa banyak prediksi serangan yang benar-benar serangan, sementara *recall* menunjukkan seberapa banyak serangan yang berhasil dideteksi dari semua serangan yang ada. Skor F1, kombinasi dari presisi dan *recall*, memberikan metrik tunggal untuk mengevaluasi model, terutama ketika ada *trade-off* antara presisi dan *recall*.

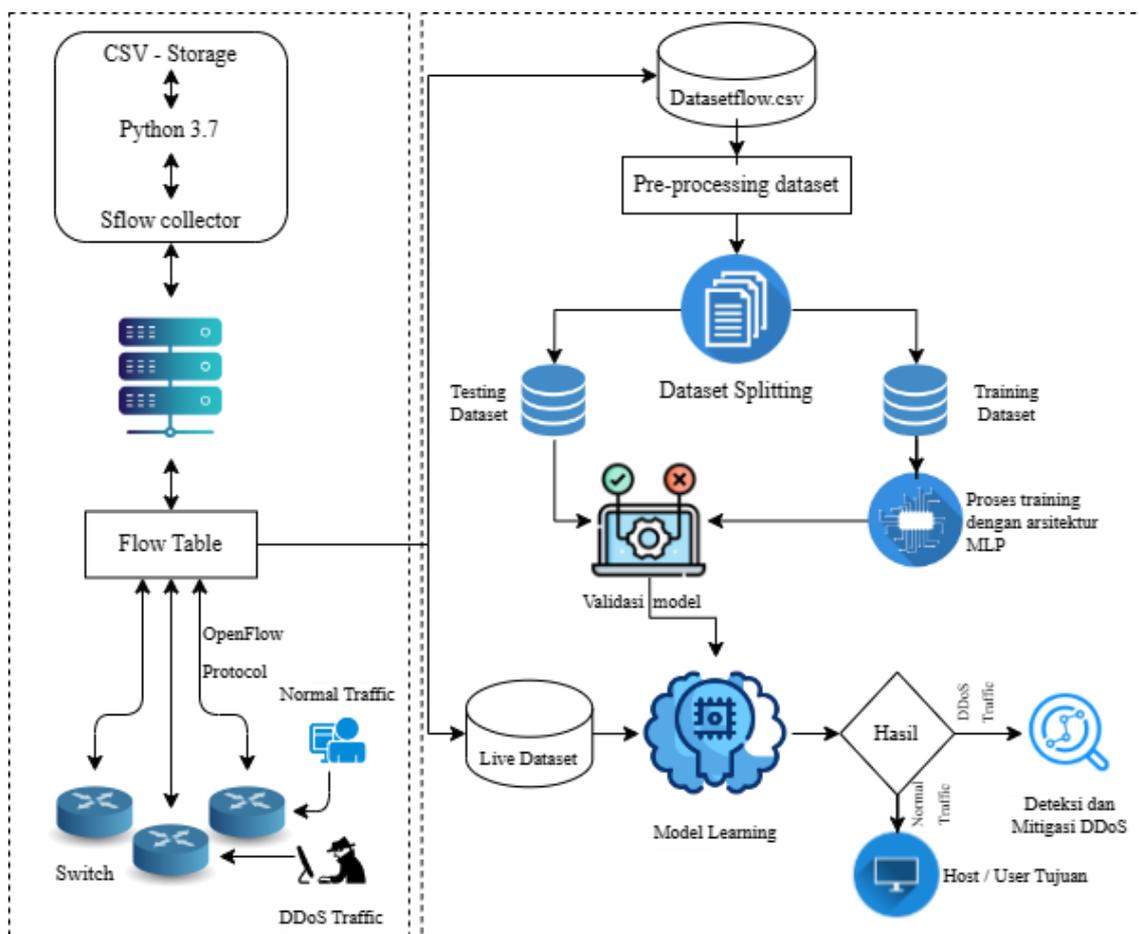
3.3.8 Analisis *Quality of Service* (QoS)

Analisis QoS dilakukan dengan menggunakan standar TIPHON untuk mengukur kualitas jaringan. Parameter yang di gunakan untuk melakukan analisis QoS yaitu *throughput*, *delay*, *jitter*, dan *packet loss*. Parameter ini kemudian akan di ukur dan dibandingkan dengan standar TIPHON untuk menentukan kategori kualitas jaringan, apakah masuk dalam kategori "Sangat Bagus", "Bagus", "Sedang", atau "Buruk". Analisis ini dilakukan baik saat jaringan berjalan normal, selama serangan DDoS, maupun setelah mitigasi. Oleh karena itu analisis ini dapat digunakan untuk membantu mengevaluasi dampak serangan DDoS terhadap performa jaringan dan efektivitas model mitigasi yang diterapkan.

Dalam penelitian ini, *iPerf* digunakan untuk mengukur *throughput* jaringan, *Ping* untuk mengukur *delay* dan *packet loss*, Wireshark untuk menganalisis lalu lintas jaringan secara detail, dan sFlow untuk memonitor lalu lintas serta menghitung metrik *resource utilization*.

Dengan membandingkan hasil pengukuran QoS terhadap standar TIPHON, dapat ditentukan apakah tindakan mitigasi yang dilakukan berhasil meningkatkan kualitas layanan jaringan. Analisis ini tidak hanya memberikan gambaran tentang kondisi jaringan pasca-mitigasi, tetapi juga membantu dalam pengambilan keputusan lebih lanjut untuk meningkatkan performa jaringan, mengurangi latensi, dan meminimalkan kehilangan data. Evaluasi ini sangat penting untuk memastikan bahwa jaringan dapat terus berfungsi secara optimal, bahkan di bawah ancaman serangan DDoS.

3.4. Perancangan Sistem



Gambar 3.6 Perancangan sistem

Sistem deteksi serangan DDoS pada jaringan SDN menggunakan algoritma MLP terdiri dari dua proses utama, yaitu proses pembelajaran dan proses pendeteksian. Proses pembelajaran dilakukan dengan menggunakan *dataset* yang berisi *flow* data jaringan normal dan DDoS yang dikumpulkan dari topologi jaringan SDN. *Dataset* tersebut kemudian digunakan sebagai data masukan untuk algoritma MLP yang digunakan untuk mengenali pola dari *flow* data jaringan.

Setelah proses pembelajaran selesai, model yang dihasilkan oleh algoritma MLP disimpan pada *controller* SDN. Proses pendeteksian dilakukan dengan menggunakan data *realtime* yang berasal dari topologi jaringan SDN yang sedang beroperasi. Data *realtime* tersebut akan diteruskan ke SDN untuk dilakukan klasifikasi oleh model yang telah dibangun sebelumnya. Jika *flow* data yang masuk kedalam *controller* SDN terdeteksi sebagai DDoS, maka *controller* SDN akan melakukan mitigasi dan memberikan peringatan kepada administrator jaringan.