TUGAS AKHIR

PERANCANGAN PREPROCESSOR PADA SISTEM PENDETEKSI INTRUSI SNORT BERBASIS CONVOLUTIONAL NEURAL NETWORK DAN LONG SHORT TERM MEMORY

Oleh:
Didik Hadumi Setiaji
F1B014020

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK
UNIVERSITAS MATARAM

Oktober 2019

LEMBAR PENGESAHAN

TUGAS AKHIR
PERANCANGAN PREPROCESSOR PADA SISTEM PENDETEKSI INTRUSI SNORT BERBASIS CONVOLUTIONAL NEURAL NETWORK DAN LONG SHORT TERM MEMORY

Oleh:
Didik Hadumi Setiaji
F1B 014 020

Telah diperiksa dan disetujui oleh Tim Pembimbing:

Pembimbing Utama

Cahyo Mustiko O. M. ST., M.Sc., Ph.D.
NIP. 19691028 199802 1 001 Tanggal:

Pembimbing Pendamping

Lalu A. Syamsul Irfan Akbar, ST., M.Eng.
NIP. 19830310 200912 1 004 Tanggal:

Mengetahui
Ketua Jurusan Teknik Elektro
Fakultas Teknik
Universitas Mataram

Muhamad Syamsu Iqbal, ST., MT., Ph.D.
NIP. 19720222 199903 1 002

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Sistem keamanan jaringan bertujuan untuk memberikan keamanan pada sistem komunikasi data. Sistem keamanan jaringan ini dapat berupa sistem enkripsi data, maupun filter data pada sisi penerima. Saat terjadinya transmisi data, maka data yang sebelumnya bersifat lokal akan bersifat publik. Sehingga data publik dapat memiliki 2 sifat yakni *malicious* atau mengandung gangguan dan *benign* atau normal. Kedua perbedaan data ini dapat diatasi dengan adanya Sistem Pendeteksi Intrusi atau *Intrusion Detection System*, dapat disingkat IDS.

Seiring dengan perkembangan teknologi, semakin banyaknya *software* yang dikembangkan, sehingga banyak pula celah keamanan yang timbul. Oleh karena itu, dibutuhkan teknologi sistem keamanan yang dapat mengatasi permasalahan tersebut. Gangguan terhadap keamanan memiliki pola gangguan yang bervariasi. Sifat keterbukaan internet menyebabkan persebaran dan variasi gangguan menjadi sangat besar. Pada umumnya Sistem Pendeteksi Intrusi melakukan pendeteksian intrusi berdasarkan pola yang identik dengan pola yang telah dikenali sebagai gangguan. Salah satu tipe gangguan yang ada adalah *virus* atau *malware* yang utuh maupun yang diselipkan ke dalam aplikasi lain.

Metode *Deep Learning* dapat menjadi solusi bagi sistem keamanan jaringan IDS. Kelebihan dari *Deep Learning* dibandingkan dengan metode kecerdasan lainnya yakni *Deep Learning* dapat mendeteksi pola berdasarkan fitur yang ada pada sebuah objek baik itu gambar maupun bentuk pola lain.

Berdasarkan sifat data *Virus* dan *Malware* yang memiliki pola yang terstruktur dan berada di dalam jaringan yang memiliki data terpotong-potong saat pengirimannya atau disebut *truncated data*. Metode *Deep Learning* yang akan digunakan adalah *Convolutional Neural Network* atau CNN, dan *Long Short Term Memory* atau LSTM. Implementasi CNN digunakan untuk mengekstraksi fitur pada sebuah data menggunakan matriks konvolusi. Sedangkan LSTM di implementasikan untuk

memprediksi fitur berdasarkan waktu atau deret.

1.2 Rumusan Masalah

SnortIDS merupakan IDS yang memiliki bagian decoder dan preprocessor yang hanya dapat menganalisis paket secara eksak dengan memanipulasi pola dan mendeteksi pola tersebut. Snort belum mengimplementasikan metode *Deep Learning* pada sistemnya. Pada penelitian ini dirancang sebuah preprocessor yang dapat memanfaatkan metode ini.

Rumusan masalah pada penelitian ini adalah mengenai rancangan preprocessor pada snort yang dapat menggunakan metode *Deep Learning* yakni CNN dan LSTM sebagai metode pendeteksian intrusinya.

1.3 Batasan Masalah

Batasan masalah yang diterapkan pada penelitian ini adalah sebagai berikut :

1. Sistem yang dirancang berbasis *service daemon* atau berjalan di belakang
2. Sistem diujikan pada jaringan kabel *ethernet* dengan skala jaringan LAN
3. Sistem diujikan dengan menonaktifkan seluruh rule yang ada pada Snort IDS
4. Pengujian sistem bersifat aktif atau disengaja
5. Pendeteksian diujikan pada file malware yang sudah di *training*.

1.4 Tujuan

Berdasarkan latar belakang dan rumusan masalah, tujuan dari penelitian ini adalah sebagai berikut :

1. Dapat meningkatkan kapabilitas sistem pendeteksi intrusi snort dengan mengintegrasikan sistem *deep learning* berupa CNN dan LSTM dalam pada sistem pendeteksi intrusi malware
2. Menemukan arsitektur dan karakteristik model *Deep Learning* yang optimal dijadikan sebagai metode pendeteksi intrusi malware

1.5 Manfaat

1. Memberikan kemudahan dalam pendeteksian intrusi *malware* dengan snort IDS

2. Menjadi referensi untuk mengimplementasikan metode *Deep Learning* pada Snort IDS. Dapat menjadi acuan dari penelitian tentang bagaimana pengaruh dan manfaat penerapan metode *deep learning*

1.6 Sistematika Penulisan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam tugas akhir ini dibagi menjadi 5 bab sebagai berikut :

- Bab I. Pendahuluan

  Bab ini membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan

- Bab II. Tinjauan Pustaka dan Landasan Teori

  Bab ini memuat tentang tinjauan pustaka yang menjabarkan hasil penelitian yang berkaitan dengan penelitian ini dan landasan teori yang menjabarkan teori-teori penunjang yang berhubungan dengan penelitian ini.

- Bab III. Metedologi Penelitian

  Memuat tentang metode penelitian, mulai dari pelaksanaan penelitian, diagram alir penelitian, menentukan alat dan bahan, lokasi penelitian, dan langkah-langkah penelitian.

- Bab IV. Hasil Penelitian

- Bab V. Penutup

BAB 2

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Beberapa penelitian tentang metode sistem pendeteksian intrusi telah dilakukan oleh peneliti-peneliti di bidang keamanan jaringan. Beberapa diantaranya menggunakan metode matematis statistik dan metode *machine learning*.

Pada Penelitian Zhang et al. (2001), tentang pendeteksian menggunakan metode statistik dan *Neural Network* pada Intrusion Detection System untuk pendeteksian intrusi pada sisi serangan protokol UDP. Pengujian dilakukan dengan menggunakan beberapa jenis neural network antara lain Perceptron, Backpropagation, PBH, Fuzzy ARTMAP, dan Radial-based Function.

Akurasi tertinggi dicapai oleh Backpropagation, dan PBH (Perceptron Backpropagation Hybrid), dimana MSR errornya berkurang seiring dengan jumlah data yang dimasukkan yang berkisar di 5% sampai 10% dari intensitas data background.

Pada penelitian Wirawan & Eksistyanto (2015), tentang penerapan kecerdasan buatan pada IDS.Penelitian ini menggunakan salah satu metode Machine Learning yakni Naive Bayes memperoleh akurasi sebesar 89% dengan running time. Penelitian ini menerapkan *Naive Bayes Classifier* dengan memilah atribut berdasarkan korelasi, dan menggunakan mean/standar deviasi untuk atribut kontinyu pada 3-interval dan 5-interval. Metode ini memperoleh akurasi sebesar 89% dengan running time rata-rata 31 detik.

Pada Penelitian Jacobus & Winarko (2014), tentang penerapan metode Support Vector Machine pada Intrusion Detection System secara Realtime menerapkan 3 kelas untuk proses pendeteksian jenis intrusi yakni normal, probe, dan DoS. Untuk beberapa tipe intrusi dapat terdeteksi dengan akurasi diatas 90%, dengan memanfaatkan kluster pada beberapa jenis faktor parameter dari header packet yang diperoleh dari hasil data mining.

Penelitian-penelitian sebelumnya banyak menggunakan metode matematis tanpa diintegrasikan dengan sistem pendeteksi intrusi yang umum dipakai contohnya

seperti *snort IDS*. Hal ini menyebabkan kurangnya implementasi praktis yang terjadi pada sistem pendeteksi intrusi yang harusnya diimplementasikan metode-metode pada penelitian tersebut.

Untuk mengatasi masalah ini. Pada penelitian ini akan diterapkan metode *Deep Learning* yang akan dijadikan prototip integrasi sistem pendeteksi intrusi dengan aplikasi modern. Metode *Deep Learning* yang akan digunakan adalah CNN dan LSTM. Metode ini menggabungkan *time series regression* pada LSTM dan memanfaatkan fungsi CNN yang dapat mendeteksi pola berdasarkan jarak relatif piksel.

Dataset yang akan digunakan pada penelitian ini berasal dari dataset CTU yang terdiri dari beberapa botnet, yang baik secara trafik penyebaran data maupun konten data memiliki ciri khusus *malware*.

Metode CNN digunakan karena dataset *payload* memiliki keacakan data tinggi namun memiliki pola yang serupa secara relatif. Metode LSTM digunakan karena keacakan data dalam bentuk *time series*. Penelitian ini diharapkan mampu mengintegrasikan sistem pendeteksi intrusi dengan metode *filter* jaringan berbasis CNN dan *profiler* jaringan berbasis LSTM.

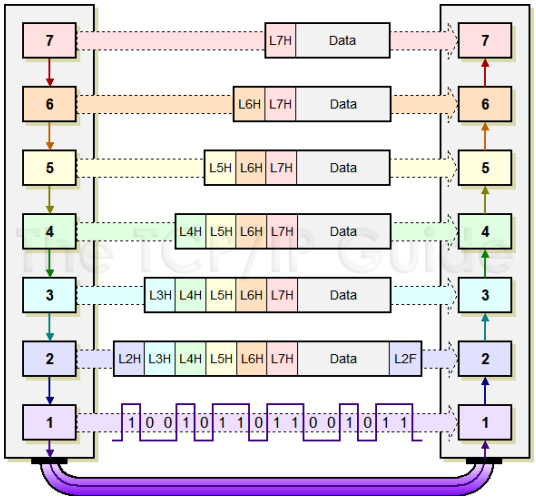
2.2 Landasan Teori

2.2.1 Protocol Data Unit

Protocol Data Unit atau dapat disingkat PDU merupakan satuan unit yang ditransmisikan antar *node* pada jaringan. PDU terdiri dari *control information* dan *user data*. Dalam arsitektur standarisasi ISO untuk protokol komunikasi, setiap layer mengimplementasikan protokol yang disesuaikan dengan tipe atau mode pertukaran data tertentu.

Protokol TCP atau *Transmission Control Protocol* menerapkan mode transfer berorientasi koneksi, dan PDU pada protokol ini disebut segmen, sedangkan pada protokol UDP atau *Unit Datagram Protocol* memiliki PDU yang disebut datagram untuk transfer data secara *connectionless*. Pada layer *network*, PDU disebut paket, terlepas dari jenis muatannya.

Berdasarkan gambar 2.1, setiap protokol memiliki bentuk PDU nya masing-masing. Diagram diatas menunjukkan layer 7 PDU yang terdiri dari L7H dan



Gambar 2.1: Packet Data Unit pada OSI Layer

data. Ketika data di teruskan ke layer 6, data ditambahkan dengan L6H sehingga menjadi layer 6 PDU, dan seterusnya sampai di layer 1 data dengan seluruh header di transmisikan. Data yang dapat diperoleh dari MTU

Data pada jaringan berbentuk *ethernet frame* dengan ukuran *Maximum Transmission Unit* (MTU) tiap frame sebesar 1500 bytes. Untuk transmisi data lebih besar dari 1500 bytes akan dilakukan pemecahan per frame.

2.2.2 IDS

Sistem deteksi intrusi (IDS) adalah perangkat atau aplikasi perangkat lunak yang memantau jaringan atau sistem untuk aktivitas jahat atau pelanggaran kebijakan. Setiap aktivitas atau pelanggaran berbahaya biasanya dilaporkan kepada administrator atau dikumpulkan secara terpusat menggunakan sistem informasi keamanan dan manajemen kejadian (SIEM). Sistem SIEM menggabungkan output dari berbagai sumber, dan menggunakan teknik penyaringan alarm untuk membedakan aktivitas berbahaya dari alarm palsu. Martellini & Malizia (2017)

Tipe IDS berkisar dalam ruang lingkup dari satu komputer ke jaringan besar. Axelsson (2000) Klasifikasi yang paling umum adalah sistem deteksi intrusi jaringan (NIDS) dan sistem deteksi intrusi berbasis host (HIDS). Sistem yang memantau file sistem operasi penting adalah contoh dari HIDS, sedangkan sistem yang menganalisis lalu lintas jaringan yang masuk adalah contoh dari NIDS. Dimungkinkan juga untuk mengklasifikasikan IDS dengan pendekatan deteksi: varian yang paling

terkenal adalah deteksi berbasis tanda tangan (mengenali pola-pola buruk, seperti malware); dan deteksi berbasis anomali (mendeteksi penyimpangan dari model lalu lintas "baik", yang sering bergantung pada pembelajaran mesin), yang lain adalah deteksi berbasis reputasi (mengenali potensi ancaman sesuai dengan skor reputasi). Beberapa produk IDS memiliki kemampuan untuk menanggapi intrusi yang terdeteksi. Sistem dengan kemampuan respons biasanya disebut sebagai sistem pencegahan intrusi. Newman (2009) Sistem deteksi intrusi juga dapat melayani tujuan tertentu dengan menambahkannya dengan alat khusus, seperti menggunakan honeypot untuk menarik dan mengkarakterisasi lalu lintas berbahaya. Liao et al. (2013)

2.2.3 Snort IDS

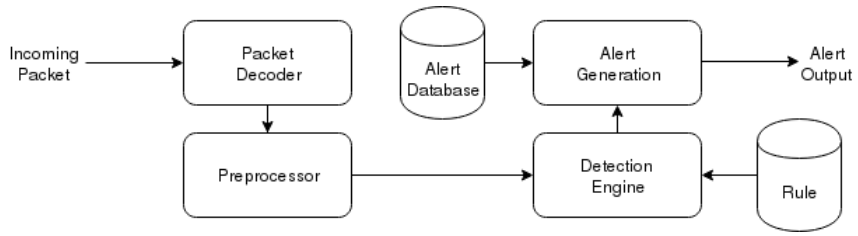
Snort adalah sistem deteksi intrusi jaringan open source (IDS) dan sistem pencegahan intrusi (IPS) Carr (2007) yang dibuat pada tahun 1998 oleh Martin Roesch, pendiri dan mantan CTO Sourcefire. Greenemeier (2006) Snort sekarang dikembangkan oleh Cisco, yang membeli Sourcefire pada 2013.

2.2.3.1 Cara kerja Snort IDS

Sistem deteksi / pencegahan intrusi (IDS / IPS) berbasis jaringan sumber terbuka milik Snort memiliki kemampuan untuk melakukan analisis lalu lintas waktu-nyata dan pendataan paket pada jaringan Internet Protocol (IP). Snort melakukan analisis protokol, pencarian konten, dan pencocokan.

SnortIDS dapat digunakan untuk mendeteksi probe atau serangan. Baik intrusi berupa *fingerprinting* sistem operasi, serangan URL, buffer overflows, probe SMB, dan *port scanning*. Karena bersifat *open source* snortIDS banyak dikembangkan dan dijadikan *engine* dasar untuk membangun IDS. Stanger (2011)

Snort dapat dikonfigurasi dalam tiga mode utama: sniffer, packet logger, dan deteksi intrusi jaringan. Dalam mode sniffer, program akan membaca paket jaringan dan menampilkannya di konsol. Dalam mode packet logger, program akan mencatat paket-paket ke disk. Dalam mode deteksi intrusi, program akan memonitor lalu lintas jaringan dan menganalisanya terhadap aturan yang ditetapkan oleh pengguna. Program kemudian akan melakukan tindakan spesifik berdasarkan apa yang telah



Gambar 2.2: Sistem kerja Snort IDS

diidentifikasi.

Berdasarkan gambar sistem kerja 2.2, dapat diketahui bahwa sistem *preprocessor* berada setelah *packet decoder* dan sebelum *detection engine*. *Preprocessor* pada snort berperan dapat sekaligus sebagai *packet decoder* dan *detection engine*. *Preprocessor* memiliki argumen yang mengacu kepada *detection engine* dan membaca *packet* yang sudah di decode seperti pada URL agar dapat teratur ketika di deteksi pada *detection engine*.

2.2.4 Deep Learning

Banyak masalah dalam penerapan kecerdasan buatan adalah bahwa banyak faktor variasi yang dapat memengaruhi setiap bagian dari data yang dapat kita amati. Contohnya, masing-masing piksel dalam gambar sebuah mobil merah mungkin menyerupai dengan mobil hitam saat di malam hari. Bentuk bayangan mobil tergantung pada sudut pandang. Sebagian besar penerapan kecerdasan buatan mengharuskan kita untuk memisahkan faktor-faktor variasi dan membuang faktor-faktor yang dapat diabaikan.

Ekstraksi fitur abstrak sangat sulit dilakukan. Banyak faktor yang bervariasi seperti aksen pembicara pada ekstraksi fitur suara yang secara langsung mudah di kenali oleh manusia sedangkan jika diterapkan secara matematis justru sangat sulit dikenali.

Deep Learning memecahkan masalah ini pada saat proses merubah bentuk dengan mengekspresikan data dalam bentuk lain yang lebih sederhana dan mudah diamati secara matematis. Deep Learning memungkinkan komputer untuk membangun konsep kompleks yang dapat memprediksi hal yang dapat dikenali manusia.

Deep learning merupakan salah satu penerapan dari Machine Learning yang

digunakan untuk memahami data yang memiliki makna. Contohnya seperti gambar, suara, maupun bentuk data lain yang dapat diinterpretasikan tidak secara matematis. Deep Learning memanfaatkan konsep matematis pada Machine Learning untuk mempelajari data yang lebih kompleks. nyata.Goodfellow et al. (2016)

2.2.5 Jaringan Syaraf Tiruan

Jaringan saraf tiruan atau JST dipandang di sini sebagai model komputasi paralel, dengan berbagai tingkat kompleksitas, terdiri dari unit pemrosesan adaptif yang saling berhubungan. Jaringan ini adalah implementasi paralel halus dari sistem statis atau dinamis nonlinear.

Fitur yang sangat penting dari jaringan ini adalah sifat adaptifnya, di mana "belajar dengan contoh" menggantikan "pemrograman" tradisional dalam menyelesaikan masalah. Fitur ini membuat model komputasi sangat cocok pada sistem yang memiliki sedikit atau pemahaman tidak lengkap atas masalah yang harus dipecahkan tetapi di mana data pelatihan tersedia.

Fitur utama lainnya adalah paralelisme intrinsik yang memungkinkan perhitungan solusi yang cepat ketika jaringan ini diimplementasikan pada komputer digital paralel atau, pada akhirnya, ketika diimplementasikan dalam perangkat keras khusus. Hassoun et al. (1995)

2.2.6 Botnet

Botnet merupakan sekumpulan program yang saling terhubung dan biasanya mengirimkan pesan spam atau berpartisipasi dalam melakukan DDoS. Botnet menyerang protokol yang menyediakan kanal komunikasi seperti protokol HTTP, IRC, dan Email.

2.2.7 CNN

CNN merupakan salah satu metode artificial neural network yang biasa diterapkan untuk data dalam bentuk gambar. Pada CNN data gambar melalui proses ekstraksi fitur. Pada proses ekstraksi fitur setiap piksel dari gambar akan diubah dalam bentuk data matriks angka. Fitur yang sudah diekstraksi kemudian akan di alirkan melalui dua layer yaitu *convolutional layer* dan *pooling layer*.

Convolutional layer terdiri dari neuron yang tersusun sehingga membentuk sebuah filter yang memiliki ukuran lebar, tinggi, dan tebal piksel. Contohnya ukuran *convolutional layer* 5x5x3, memiliki panjang 5 piksel, tinggi 5 piksel dan tebal 3 yang merupakan channel dari gambar dalam bentuk representasi RGB.

Filter ini kemudian akan digeser ke seluruh bagian dari gambar yang kemudian akan di lakukan operasi perkalian antara input dan nilai dari filter sehingga menghasilkan output yang disebut *feature map*. Parameter pada convolutional layer, salah satunya adalah Stride. Stride adalah parameter yang menentukan jumlah pergeseran filter. Jika nilai stride 1, maka filter akan bergeser sebanyak 1 piksel.

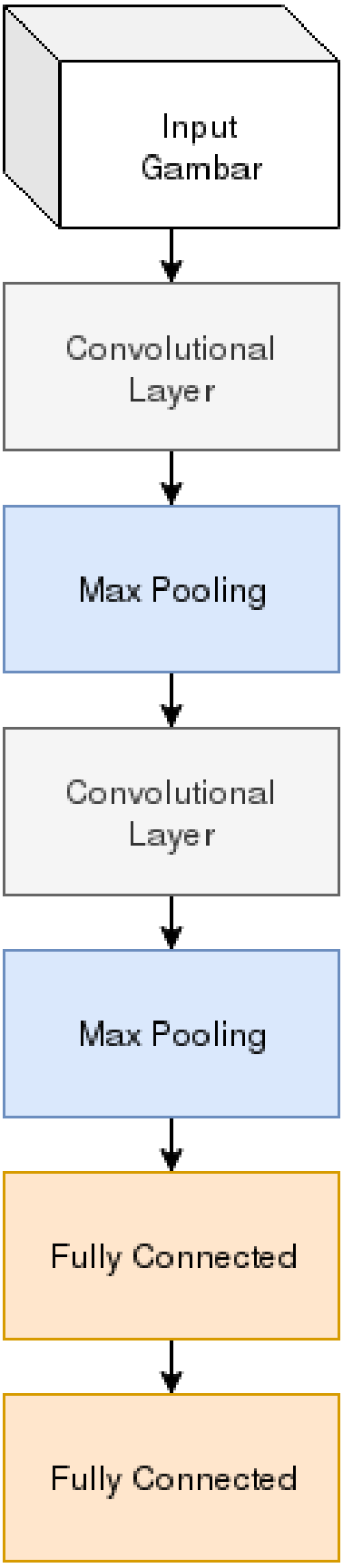
Semakin kecil stride maka akan semakin detil informasi yang diperoleh dari input. Padding merupakan parameter yang menentukan jumlah piksel (bernilai 0) yang akan ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi output dari feature map. Pooling Layer merupakan layer CNN yang berada setelah Convolutional Layer. Pooling layer merupakan sebuah filter yang bergeser juga pada seluruh area feature map.

Pada setiap pergeseran filter, nilai maksimum akan diambil sebesar ukuran dari filter. Contoh apabila menggunakan max pooling 2x2 dengan stride 2 maka setiap pergeseran filter, nilai maksimum pada area 2x2 akan dipilih, sedangkan Average Pooling memilih nilai rata-rata. Berikut ini adalah contoh sederhana bentuk *Convolutional Neural Network*

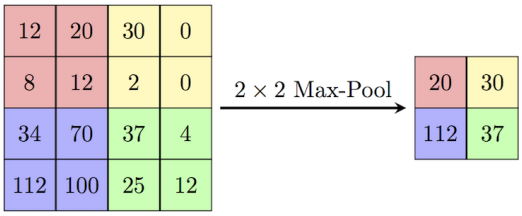
Berdasarkan gambar 2.3 dapat diketahui sederhananya model CNN memiliki komponen dasar yakni *Convolutional Layer*, *Max Pooling*, dan *Fully Connected Layer*. Pada *Convolutional Layer* volume akan diperkecil seiring dengan dimasukannya input pada Layer ini. Misal dijabarkan persamaan sebagai berikut.

$$V_1 = W_1 \times H_1 \times D_1 \tag{2.1}$$

Berdasarkan persamaan 2.1, diketahui W_1 merupakan lebar gambar H_1 merupakan tinggi gambar, dan D_1 merupakan *deepthness* gambar yang menunjukan komponen RGB dari gambar. Agar dapat dimasukkan pada CNN harus memiliki beberapa *hyperparameter* yang dibutuhkan antara lain.



Gambar 2.3: Model CNN sederhana



Gambar 2.4: Max Pooling Layer

K = jumlah filter, F = *spatial extents*, S = *stride*, dan P = jumlah *zero padding*. Maka akan dihasilkan volume V_2

$$V_2 = W_2 \times H_2 \times D_1 \tag{2.2}$$

Dengan parameter persamaan 2.2 diperoleh dari :

$$W_2 = (W_1 - F + 2P)/S + 1 \tag{2.3}$$

$$H_2 = (H_1 - F + 2P)/S + 1 \tag{2.4}$$

$$D_2 = K \tag{2.5}$$

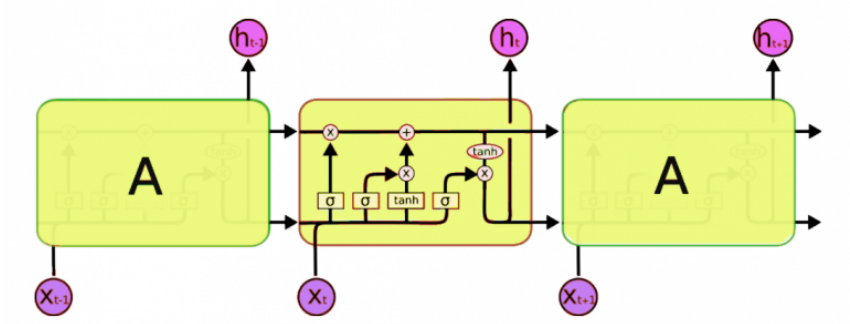
Dapat dilihat dari persamaan 2.3 dan 2.4 bahwa ukuran W_2 dan H_2 akan tetap simetris ketika dimasukkan dalam CNN.

Berdasarkan gambar 2.4, pada layer *max pooling* ukuran akan direduksi kembali dan sesuai dengan ketentuan *stride* dan ukuran kernel yan akan diambil nilai terbesarnya.

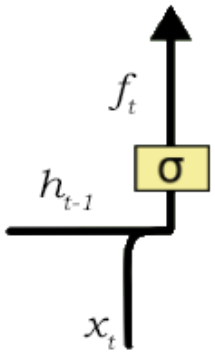
Kemudian pada layer aktivasi akan diaktivasi dapat menggunakan aktivasi *sigmoid*, aktivasi *relu*, atau aktivasi lainnya.

2.2.8 LSTM

LSTM (Long short-term memory) memiliki kemampuan untuk melupakan informasi yang tidak relevan pada jaringan RNN (recurrent neural network). LSTM merupakan RNN yang terdiri dari 4 bagian yakni, cell state, input gate, output gate, dan forget gate. Cell state merupakan bobot dari hidden layer. Input gate merupakan jalur nilai input datang. Forget gate merupakan jalur yang berfungsi untuk



Gambar 2.5: Layer LSTM



Gambar 2.6: Forget Gate

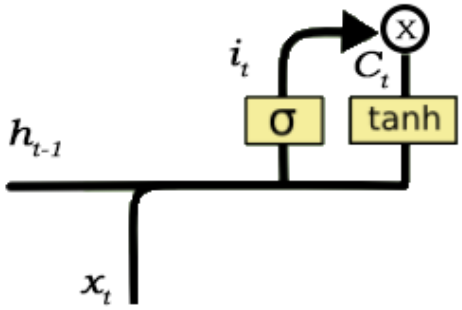
meniadakan atau melupakan informasi yang datang. Output gate merupakan jalur keluaran berdasarkan cell state setelah forget gate dilewati.

Berdasarkan gambar 2.5, sebuah sel *Long Short Term Memory* terdiri dari empat komponen yakni komponen gate *input gate*, *output gate*, dan *forget gate*, dan juga komponen *cell state*. Keempat komponen ini masing-masing memiliki perannya sendiri. Setiap peran yang ada pada komponen ini yang membuat LSTM dapat lebih di andalkan dalam proses pendeteksian data sekuensial.

2.2.8.1 Forget Gate

Forget Gate bertanggung jawab untuk menghapus informasi dari status sel. Informasi yang tidak lagi diperlukan untuk LSTM atau informasi yang kurang penting dihapus melalui gerbang ini. Hal Ini diperlukan untuk mengoptimalkan kinerja jaringan LSTM.

Dapat dilihat dari gambar 2.6, bahwa *forget gate* menerima dua input h_{t-1} dan x_t. h_{t-1} adalah keadaan tersembunyi dari sel sebelumnya atau output dari sel sebelumnya



Gambar 2.7: Input Gate

dan x_t adalah input sesaat. gambar 2.6 diatas dapat dijabarkan oleh persamaan *forget gate* berikut ini :

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2.6}$$

Dengan f_t = *forget gate*, σ = fungsi sigmoid, w_f = bobot dari forget gate, h_{t-1} = output dari sel lstm sebelumnya saat $t - 1$, x_t = input dari waktu saat ini, b_x = bias dari gate.

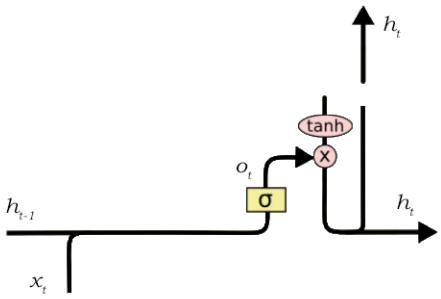
Berdasarkan persamaan 2.6, input yang diberikan dikalikan dengan matriks bobot dan kemudian ditambahkan bias kemudian di masukkan pada fungsi sigmoid. Fungsi sigmoid menghasilkan vektor, dengan nilai mulai dari 0 hingga 1, sesuai dengan setiap angka dalam keadaan sel.

Pada dasarnya, fungsi sigmoid bertanggung jawab untuk memutuskan nilai mana yang harus disimpan dan mana yang harus dibuang. Jika output dari *forget gate* bernilai '0' maka *forget gate* akan melupakan keadaan sel melupakan informasi itu sepenuhnya. Sebaliknya, apabila output dari *forget gate* bernilai '1' maka *forget gate* akan mengingat seluruh informasi itu. Output vektor ini dari fungsi sigmoid dikalikan ke *cell state*.

2.2.8.2 Input Gate

Input Gate bertanggung jawab untuk penambahan informasi ke status sel. Penambahan informasi ini pada dasarnya adalah proses tiga langkah seperti yang terlihat dari diagram di atas.

Berdasarkan Gambar 2.7, *input gate* memiliki parameter h_{t-1} yang merupakan



Gambar 2.8: Output Gate

nilai *hidden cell* dari waktu sebelumnya, dan x_t yang merupakan nilai *hidden cell* dari waktu sekarang, dan σ yang merupakan fungsi aktivasi, sehingga dapat dijabarkan persamaan *input gate* berikut ini :

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{2.7}$$

dengan i_t = input gate, σ = fungsi sigmoid, w_i bobot dari input gate, h_{t-1} = output dari sel lstm sebelumnya $t-1$, x_t = input dari waktu saat ini, b_i bias dari input gate.

1. Mengatur nilai apa yang perlu ditambahkan ke keadaan sel dengan melibatkan fungsi sigmoid. Ini pada dasarnya sangat mirip dengan gerbang lupa dan bertindak sebagai filter untuk semua informasi dari h_{t-1} dan x_t.

2. Membuat vektor yang berisi semua nilai yang mungkin yang dapat ditambahkan (seperti yang dirasakan dari h_{t-1} dan x_t. Hal ini dilakukan dengan menggunakan fungsi tanh, yang menampilkan nilai dari -1 hingga +1.

3. Mengalikan nilai filter pengaturan (gerbang sigmoid) ke vektor yang dibuat (fungsi tanh) dan kemudian menambahkan informasi berguna ini ke *cell state* melalui operasi penambahan.

4. Setelah proses tiga langkah ini selesai, layer memastikan bahwa hanya informasi yang ditambahkan ke keadaan sel yang relevan.

2.2.8.3 Output Gate

Berdasarkan gambar 2.8, *output gate* menerima input dari h_{t-1} dan x_t, yang langsung diaktivasi dengna fungsi sigmoid. Sehingga dapat dijabarkan persamaan untuk *output gate* adalah sebagai berikut :

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{2.8}$$

Dengan o_t = output gate, σ = fungsi sigmoid, w_o merupakan bobot dari *output gate*, h_{t-1} = output dari sel lstm sebelumnya $t-1$, x_t = input dari waktu saat ini, b_o = bias dari output gate. Berdasarkan persamaan 2.8, fungsi dari *output gate* terbagi menjadi tiga langkah:

1. Membuat vektor setelah menerapkan fungsi tanh ke status sel, sehingga menskalakan nilai ke rentang -1 hingga +1.

2. Membuat filter menggunakan nilai-nilai h_{t-1} dan x_t, sehingga dapat mengatur nilai-nilai yang perlu dikeluarkan dari vektor yang dibuat di atas. Filter ini lagi menggunakan fungsi sigmoid.

3. Mengalikan nilai filter pengaturan ini ke vektor yang dibuat pada langkah 1, dan mengirimkannya sebagai output dan juga ke keadaan tersembunyi sel berikutnya.

Filter pada contoh di atas memastikan bahwa output mengurangi semua nilai lain kecuali nilai cell state. Oleh karena itu filter juga perlu dibuat pada input dan nilai-nilai *hidden state* dan diterapkan pada vektor *cell state*. Gers et al. (1999)

2.2.9 Python

Python merupakan bahasa pemrograman yang memiliki banyak *library* yang memiliki banyak fungsi dan penerapan. Beberapa *library* python antara lain seperti *tensorflow* dan *keras* yang merupakan framework yang digunakan untuk membuat model jaringan syaraf tiruan maupun *machine learning*. Banyaknya *library* python seperti tensorflow dan keras menyebabkan python sebagai salah satu bahasa pemrograman yang paling umum digunakan pada bidang data sains.

Python banyak digunakan oleh *Data scientist* untuk melakukan penelitian model kecerdasan buatan. Contohnya kegunaan *library* keras yang biasa diterapkan pada jaringan syaraf tiruan.

2.2.10 Flask

Flask adalah framework web yang dibuat dengan Python. framework ini termasuk dalam kategori *microservice* karena tidak memerlukan *tools* atau *library* tertentu atau dapat juga disebut dengan *microframework*.

Flask tidak memiliki lapisan abstraksi untuk basis data, validasi formulir, dan juga tidak memiliki *third-party library* yang menyediakan fungsi umum. Flask mendukung ekstensi yang dapat menambahkan fitur aplikasi sehingga dapat diimplementasikan dalam Flask itu sendiri. Ekstensi yang ada seperti *Object Relational Mappers*, *Form Validation*, dan *upload handling*. Ekstensi flask lebih sering diperbaharui dari *core* flask sendiri.

2.2.11 Keras

Keras adalah salah satu library python untuk jaringan syaraf tiruan. Keras merupakan *framework* dari tensorflow, CNTK, atau Theano. Keras dikembangkan sebagai solusi untuk memungkinkan eksperimen jaringan syaraf tiruan dengan cepat. Keras memungkinkan peneliti lebih cepat mengaplikasikan idenya langsung dalam bentuk hasil yang di inginkan.

BAB 3

METODE PENELITIAN

3.1 Metode Penelitian

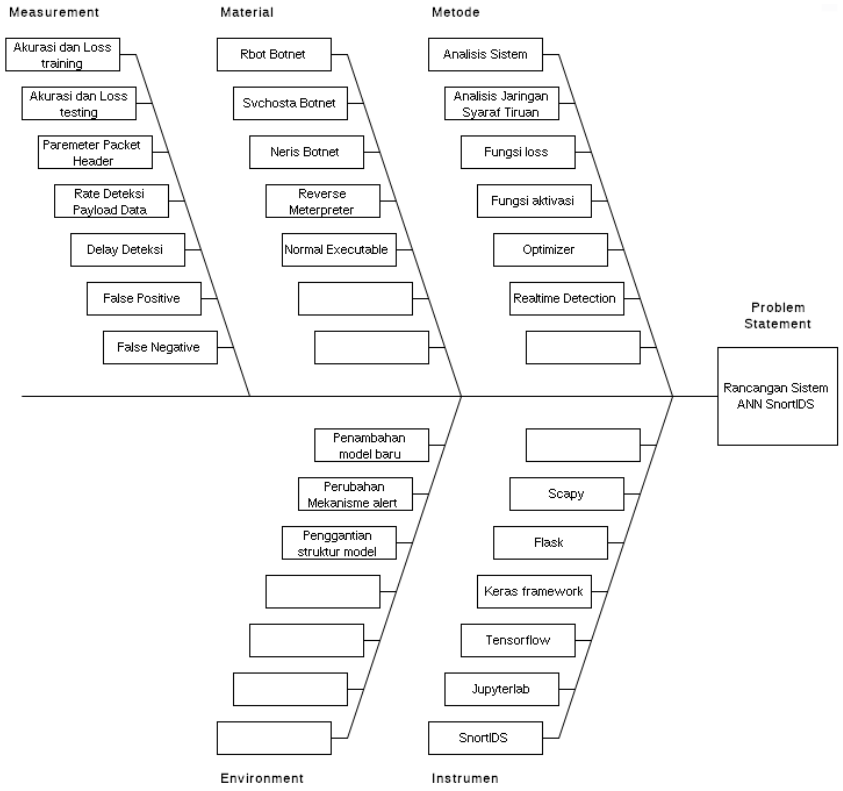
Metode penelitian yang digunakan pada penelitian ini adalah metode penelitian kuantitatif deskriptif. Berikut adalah gambaran penelitian yang diusulkan. Berdasarkan grafik metode penelitian 3.1, dapat dijabarkan beberapa komponen pada proses penelitian sehingga menghasilkan sebuah produk sistem yang diinginkan.

Pada bagian *measurement* menjelaskan apa saja yang dijadikan ukuran yang akan dianalisis pada penelitian ini. Parameter yang dijadikan ukuran yang akan dianalisis adalah akurasi dan loss training dan testing. Pada sistem *profiling* diinputkan parameter *packet header* yang dijadikan input pada LSTM. Hasil dari sistem profiling ini adalah bagaimana bentuk trafik jaringan yang akan dipelajari dan dianggap normal. Pada sistem *filtering* yang dijadikan input adalah payload data, dan akan diukur hasil akurasi training dan testing pada data normal dan malicious. Pada snortIDS akan dikukur Delay Deteksi yang terjadi.

Pada bagian *material* berisi beberapa dataset yang digunakan. Dataset yang digunakan adalah file executable windows dari botne rbot, neris, dan svchosta. Dan beberapa file tambahan seperti executable dari file exe biasa, dan virus reverse meterpreter yang digunakan untuk dijadikan tambahan pada saat testing.

Pada bagian *environment* berisi tentang parameter kerja yang akan diubah pada sistem. Parameter ini adalah dengan menambah model pada sistem profiling yang berbeda-beda. Pada snort IDS parameter yang diubah adalah mekanisme alert pada *preprocessor* nya. Dan pada proses *filtering* dengan mengganti-ganti struktur model baik urutan maupun ukuran filter.

Pada bagian instrumen berisi tentang apa saja instrumen yang digunakan untuk memperoleh hasil penelitian. Instrumen ini antara lain *scapy*, *flask*, *keras framework*, *tensorflow*, *jupyterlab*, *snortIDS*. Scapy adalah *library* snort yang digunakan untuk mengekstraksi data dari trafik jaringan. Flask adalah *microframework* yang digunakan untuk melakukan binding model Jaringan Syaraf Tiruan. Keras sebagai



Gambar 3.1: Metode Penelitian

framework standar dari *tensorflow* digunakan untuk merancang model CNN dan LSTM.

Pada bagian metode berisi metode penelitian yang digunakan. Analisis sistem dilakukan untuk menjabarkan bagaiman sistem yang digunakan pada sistem *filtering* dan *profiling*. Analisis jaringan syaraf tiruan merupakan langkah yang dilakukan untuk menjabarkan bagaimana bentuk model jaringan syaraf tiruan yang cocok digunakan pada dataset penelitian ini. Fungsi loss, aktivasi dan optimizer merupakan parameter yang diterapkan pada model jaringan syaraf tiruan. *Realtime detection* adalah metode yang digunakan untuk memperoleh delay sistem dan *detection rate* pada snortIDS.

3.2 Waktu dan Tempat Pelaksanaan

Waktu dan tempat pelaksanaan penelitian ini adalah sebagai berikut :

3.2.1 Waktu Pelaksanaan

Waktu pelaksanaan penelitian dilakukan dari tanggal 1 Maret 2019 sampai dengan 31 Juli 2019.

3.2.2 Tempat Pelaksanaan

Pelaksanaan penelitian dilakukan di Laboratorium 3 UPT Pusat Teknologi Informasi dan Komunikasi Universitas Mataram

3.3 Populasi dan Sampel

Populasi dan sampel penelitian ini adalah sebagai berikut :

3.3.1 Populasi Penelitian

Populasi pada penelitian ini mencakup intrusi dalam bentuk Executable-based Virus dan Payload-based Virus

3.3.2 Sampel Penelitian

Sampel pada penelitian ini antara lain :

1. Neris Botnet executable untuk CNN
   Merupakan salah satu botnet yang diperoleh dari CTU-42 dataset. Botnet ini mengirimkan data post request pada sistem C&C channel. Payload dan executable dari botnet ini akan dijadikan sampel pada sistem *filtering*

2. Neris Botnet Traffic Data pcap untuk LSTM
   Berkas pcap atau *packet capture* dari botnet ini dianalisis pada sistem *profiling*.

3. RBot Botnet executable untuk CNN Merupakan salah satu botnet yang diperoleh dari CTU-92 dataset. Botnet ini merupakan botnet lama yang menyerang sistem IRC dengan melakukan DDOS dan Spam. Payload dan executable dari botnet ini dijadikan sampel pada sistem *filtering*

4. RBot Botnet Traffic Data pcap untuk LSTM Berkas pcap atau *packet capture* dari botnet ini dianalisis pada sistem *profiling*.

5. Svchosta Botnet executable untuk CNN

   Merupakan salah satu botnet yang diperoleh dari CtU-47 dataset. Botnet ini merupakan botnet yang menyerang menginfeksi file svchosta pada windows dan menyebabkan email spam, nama asli botnet ini adalah *DonBot* botnet. Payload dan executable dari botnet ini akan dijadikan sampel pada sistem *filtering*

6. Svchosta Botnet Traffic Data pcap untuk LSTM Berkas pcap atau *packet capture* dari botnet ini dianalisis pada sistem *profiling*.

7. Payload Reverse Meterpreter dalam file PE32

   Beberapa file executable malicious yang diselipkan dengan sengaja menggunakan *metasploit framework*. Payload dan pcap dari file ini akan dijadikan bahan untuk melakukan testing pada sistem *filtering* dan *profiling*.

8. File executable normal untuk CNN

   Beberapa file executable normal seperti firefox dan executable dari game lama windows. Payload dan pcap dari file ini akan dijadikan bahan untuk melakukan testing pada sistem *filtering* dan *profiling*.

3.4 Instrumen Penelitian

Instrumen penelitian yang digunakan antara lain :

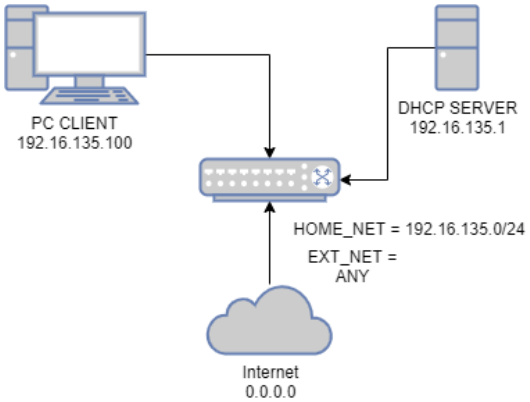
1. SnortIDS sebagai Intrusion Detection System

2. Python sebagai bahasa pemrograman yang digunakan untuk pemodelan LSTM dan CNN

3. Library python seperti : tensorflow, keras, numpy, matplotlib untuk perlengkapan pada saat training data dan testing data dengan Convolutional Neural Network dan Long Short Term Memory.

4. Python flask sebagai library untuk webservice filter dan profiler

5. PC Server Linux sebagai host yang menjalankan SnortIDS, dan Python



Gambar 3.2: Topologi jaringan

3.4.1 Proses Pengembangan Instrumen

Proses pengembangan instrumen berawal dari perancangan instrumen, uji validitas instrumen, dan uji reliabilitas instrumen.

3.4.2 Uji Validitas Instrumen

Proses pengujian validitas instrumen dilakukan dengan menguji IDS dengan data input yang sama, lalu mengamati hasilnya yang kemudian akan disesuaikan.

3.4.3 Uji Reliabilitas Instrumen

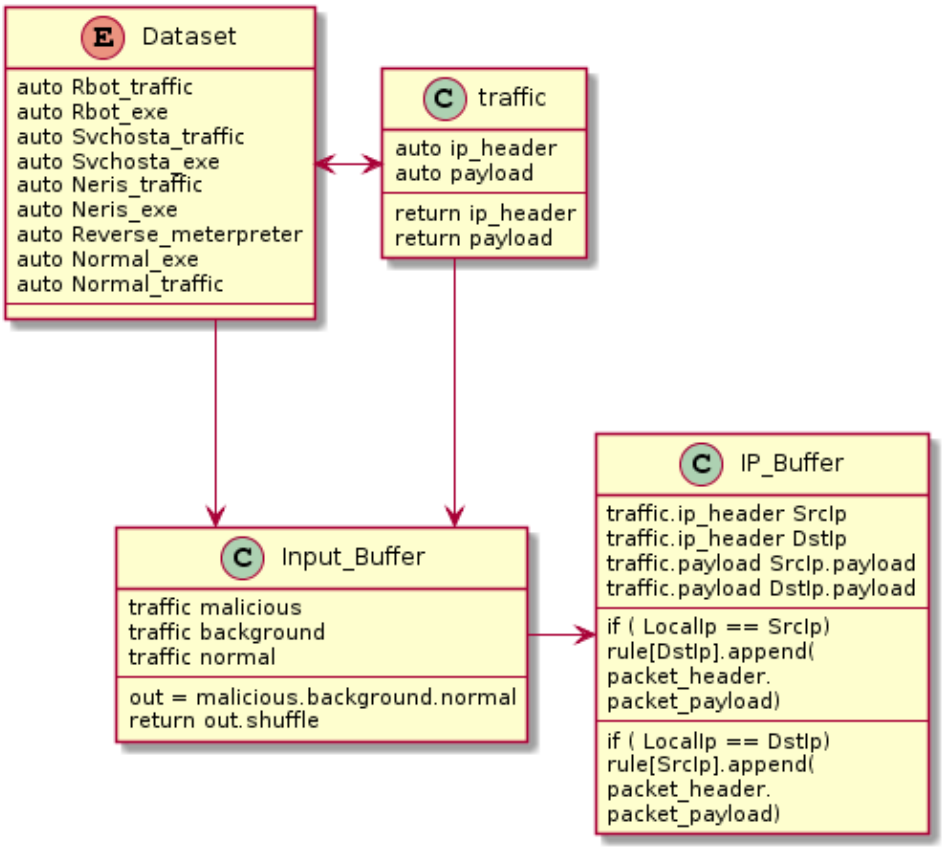
Proses pengujian reliabilitas instrumen dilakukan dengan menguji IDS dengan data input yang bervariasi ukuran datanya, dan dimuat dalam bentuk tabel *detection rate*.

3.5 Prosedur Penelitian

Prosedur penelitian terdiri dari perancangan sistem training dan perancangan sistem filtering dan profiling untuk testing. Rancangan sistem training dibuat dengan menggunakan jupyter notebook. Sedangkan rancangan sistem filter menggunakan snortIDS dan flask framework untuk menghubungkan keluaran data dari snortIDS ke dalam model CNN dan LSTM.

3.5.1 Perancangan Sistem

Sistem preprocessor diterapkan pada topologi jaringan seperti pada gambar 3.2. Seluruh proses dilakukan pada sisi PC Client. Ketika PC Client mendownload data



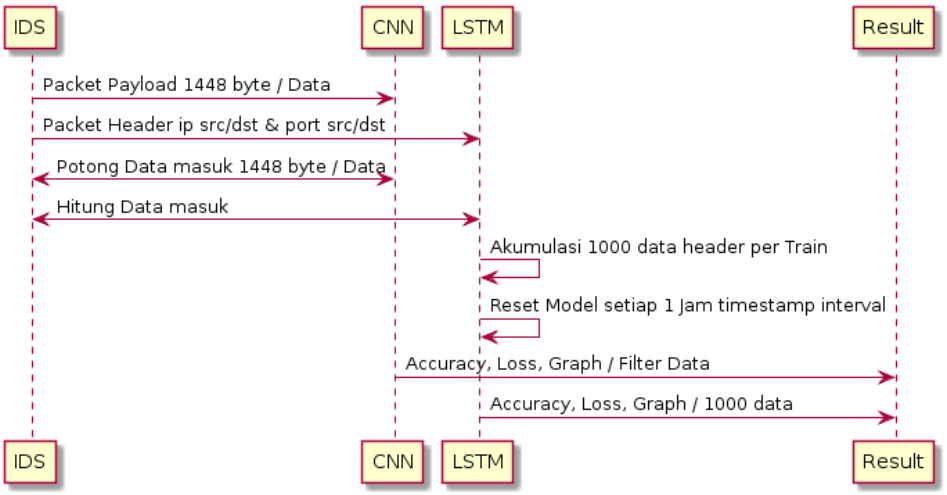
Gambar 3.3: Model Rancangan Sistem

dari internet atau PC lain mengupload data ke PC Client, PC Client akan melakukan proses filter dan profiling, kemudian menentukan bahwa data yang dikirim aman atau tidak.

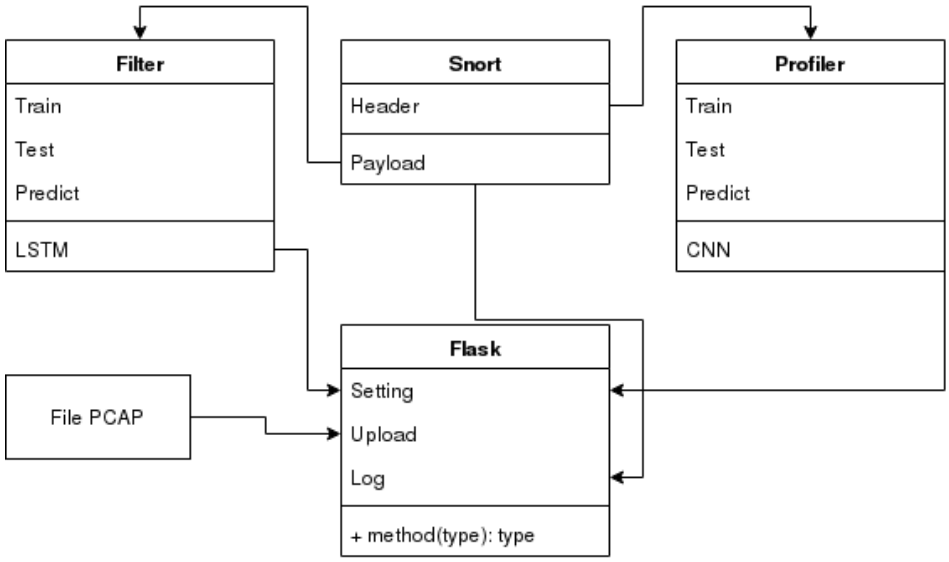
Rancangan sistem filter terdiri dari Dataset yang dijadikan data untuk proses testing. Pada sisi filter juga terdapat IP_buffer yang digunakan sebagai input yang akan mengumpulkan packet yang bersumber dari alamat IP yang sama.

Berdasarkan gambar 3.3, dapat dijabarkan rancangan sistem yang akan dibuat untuk *preprocessor* terdiri dari sistem *profiling* dengan metode LSTM dan sistem *filtering* dengan metode CNN.

Dari gambar 3.3 juga dapat dijelaskan bahwa dataset yang digunakan adalah beberapa *botnet*, *reverse meterpreter*, dan *file executable normal*. Kemudian dari beberapa file ini akan diambil bagian-bagian yang akan dijadikan analisa yakni bagian *header* dan bagian *payload* nya. Sebelum dapat diproses oleh sistem, semua isi packet ini di masukkan ke dalam input buffer untuk kemudian dilanjutkan. Untuk



Gambar 3.4: Proses kerja sistem training



Gambar 3.5: Proses Kerja sistem

di proses oleh sistem.

Berdasarkan gambar 3.4, dapat dijabarkan proses kerja sistem yang akan dirancang. Rancangan sistem untuk mendapatkan hasil diperoleh dari IDS sebagai sistem pendeteksi intrusi, CNN sebagai sistem *filtering*, dan LSTM sebagai sistem *filtering*

Berdasarkan gambar 3.5 dapat dijabarkan rancangan sistem inti terdiri dari snort, filter, dan profiler, dengan tampilan menggunakan flask. Pemroresan data dan analisis data terjadi pada sistem snort, LSTM dan CNN, yang kemudian hasilnya akan diteruskan menjadi grafik dan ditampilkan dengan flask.

3.6 Analisis Data

Proses analisis data meliputi analisis tabel dan analisis grafis dari seluruh hasil pengujian validitas dan reliabilitas instrumen

3.6.1 Prosedur Pengolahan Data

Sebelum data dimasukkan, data yang masih mentah diubah menjadi data angka. Pada sistem *profiling*, data angka yang dijadikan input adalah data header yang berisi *IP Address*, *Port* baik sumber maupun tujuan dan beberapa parameter header lainnya. Pada sistem *filtering*, data angka yang dijadikan input diperoleh dari data 1448 byte per frame packet yang diperoleh dari *packet payload*.

Setelah diperoleh data angka pada sistem *profiling* di inputkan data angka header, lalu proses *training* dilakukan untuk memperoleh hasil regresi dari trafik jaringan. Sedangkan pada sistem *filtering* di inputkan data payload yang kemudian akan di lakukan training untuk mengklasifikasi data yang dimasukkan aman atau intrusi. Data hasil training dan testing dari proses ini akan dianalisis untuk menentukan model yang akan digunakan pada sistem pendeteksi intrusi SnortIDS.

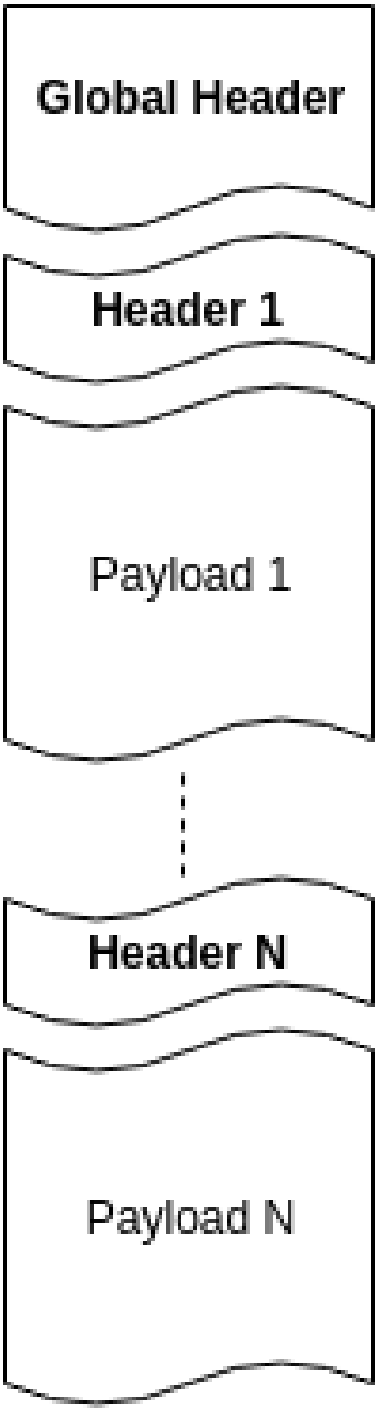
Setelah diperoleh model yang sesuai diperoleh. Model yang sudah jadi dibinding oleh *flask* dan dapat diakses dengan HTTP Request. Data hasil keluaran snortIDS akan dikirim ke *flask* ini untuk dilakukan analisis dan diperoleh hasil prediksinya. Dari proses ini diperoleh delay sistem dan *detection rate* dari kinerja snortIDS.

3.6.1.1 Proses Parsing Paket Jaringan

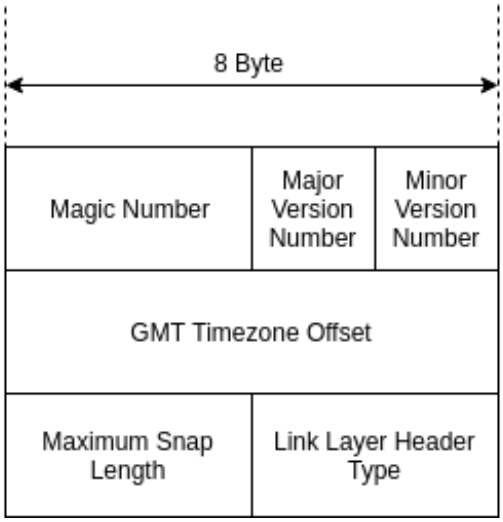
Sebelum data dapat diolah, sebelumnya data mentah pcap harus melalui ekstraksi header dan payload. Diketahui struktur data PCAP adalah sebagai berikut.

Dapat dilihat pada gambar 3.6, file pcap terdiri dari sebuah Global Header dan beberapa *Packet Header* dan *Packet Payload*.

Berdasarkan gambar 3.7 dapat diamati *Global Header* terdiri dari 4 byte *Magic Number* yakni D4 C3 B2 A1 yang menandakan jenis file PCAP. Kemudian diikuti dengan 4 byte *Version Number* yang merupakan versi dari file PCAP. Kemudian 8 byte berikutnya berisi timezone offset berdasarkan lokasi dari file PCAP diambil. Kemudian 4 byte selanjutnya berisi *Maximum Snap Length* yang berisi tentang



Gambar 3.6: Struktur Data File PCAP



Gambar 3.7: Global Header

Ethernet Header
u_char ether_dhost[6]
u_char ether_shost[6]
u_short ether_type

Gambar 3.8: Ethernet Header

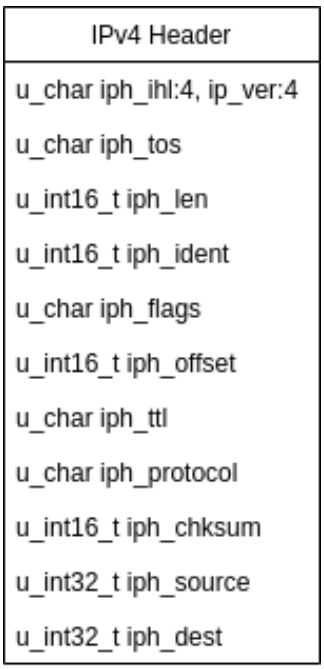
ukuran maksimum frame setiap packet yang diambil. Kemudian 4 byte selanjutnya berisi *Link Layer Header Type* yang berisi informasi tentang link layer yang digunakan.

Beberapa header yang umum pada *Packet Header* antara lain :

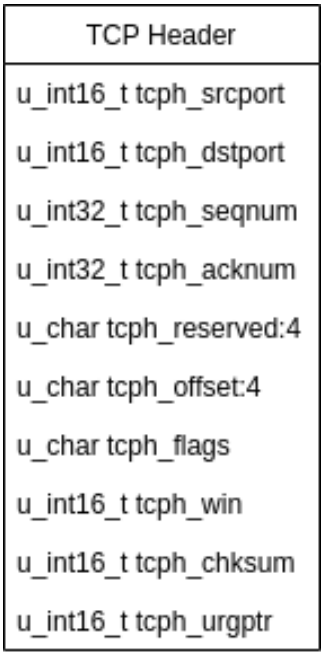
Ethernet Header dijelaskan pada gambar 3.8, header ini berisi informasi MAC Address pengirim dan penerima. Header ini berukuran 14 byte yang terdiri dari ether_dhost sebesar 6 byte, ether_shost sebesar 6 byte, dan ether_type sebesar 2 byte.

IPv4 Header dijelaskan pada gambar 3.9, header ini berisi informasi tentang alamat IP sumber dan tujuan. Header ini berukuran 20 byte yang terdiri dari 1 byte iph_ihl dan iph_ver, 1 byte iph_tos, 2 byte iph_len, 2 byte iph_indent, 1 byte iph_flags, 2 byte iph_offset, 1 byte iph_ttl, 1 byte iph_protocol, 2 byte iph_chksum, 4 byte iph_source, 4 byte iph_dhost.

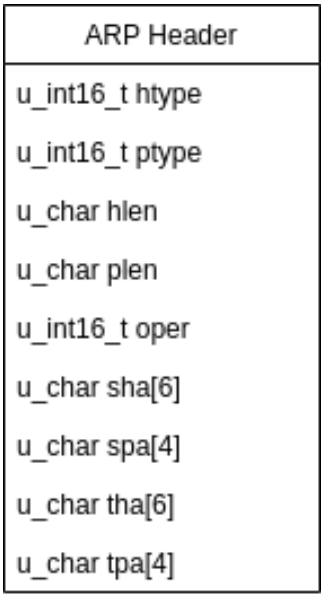
TCP Header memiliki struktur sesuai dengan gambar 3.10, header ini berisi ten-



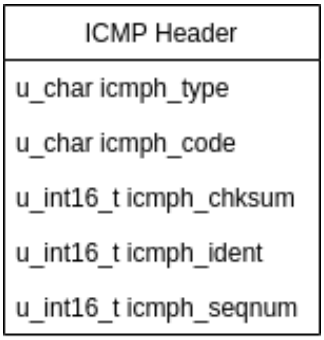
Gambar 3.9: IPV4 Header



Gambar 3.10: TCP Header



Gambar 3.11: ARP Header

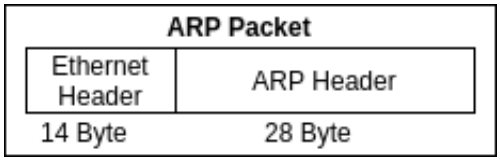


Gambar 3.12: ICMP Header

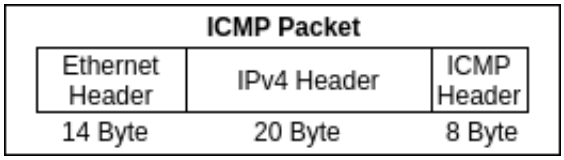
tang informasi port sumber dan tujuan. Header ini berukuran 20 byte yang terdiri dari 2 byte tcph_srcport, 2 byte tcph_dstport, 4 byte tcph_seqnum, 4 byte tcph_acknum, 1 byte tcph_reserved dan tcph_offset, 1 byte tcph_flags, 2 byte tcph_win, 2 byte tcph_chksum, 2 byte tcph_urgptr.

ARP Header memiliki struktur data sesuai dengan gambar 3.11, header ini berisi tentang informasi tentang *Address Resolution Protocol*. Header ini berukuran 28 byte yang terdiri dari 2 byte htype, 2 byte ptype, 1 byte hlen, 1 byte plen, 2 byte operation, 6 byte sha, 4 byte spa, 6 byte tha, 4 byte tpa.

ICMP (*Internet Control Message Protocol*) Header memiliki struktur data pada gambar 3.12, header ini berisi tentang informasi *Control Message* yang dikeluarkan oleh program seperti Ping ataupun Traceroute. Header ini berukuran 8 byte ter-



Gambar 3.13: ARP Packet



Gambar 3.14: ICMP Packet

diri dari 1 byte icmph_type, 1 byte icmph_code, 2 byte icmph_chksum, 2 byte icmph_ident, 2 byte icmph_seqnum.

Beberapa jenis packet terdiri dari beberapa header, contohnya adalah ARP Packet, ICMP Packet, UDP Packet, TCP Packet, dan HTTP Packet yang merupakan bahan analisis pada penelitian ini.

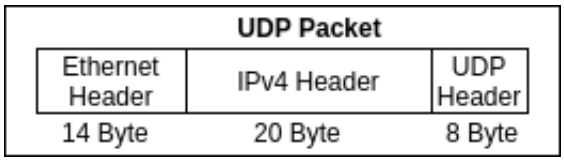
Gambar 3.13 menunujukkan bahwa ARP Packet hanya terdiri dari Ethernet Header yang berukuran 14 byte, dan ARP Header 28 byte.

Dari gambar 3.14 dapat diamati bahwa ICMP Packet terdiri dari Ethernet Header yang berukuran 14 byte, IPV4 Header yang berukuran 20 byte, dan ICMP Header yang berukuran 8 byte.

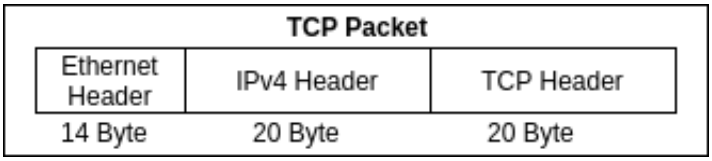
Dapat diamati pada gambar 3.15, UDP Packet terdiri atas 14 byte Ethernet Header, 20 byte IPv4 Header, dan 8 byte UDP Header.

Gambar 3.16 menjabarkan bahwa TCP Packet terdiri atas 14 byte Ethernet Header, 20 byte IPv4 Header, dan 20 byte TCP Header.

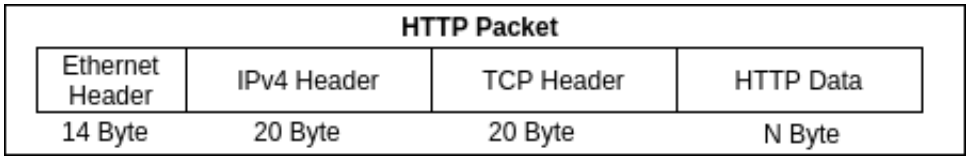
Dapat diamati dari gambar 3.17 bahwa HTTP Paket merupakan TCP Paket yang memiliki data HTTP di dalamnya. Data ini yang kemudian akan dijadikan bahan analisa pada penelitian ini sebagai payload.



Gambar 3.15: UDP Packet



Gambar 3.16: TCP Packet



Gambar 3.17: HTTP Packet

3.6.2 Teknik Analisis Data

Analisis data dilakukan berdasarkan dua jenis data yang diperoleh yakni Sistem *Profiling* oleh LSTM, dan Sisitem *Filtering* oleh CNN. Sistem Profiling LSTM menggunakan tiga metode yakni *Sentiment Analysis*, *Multivariate Prediction*, dan *4 Directional Header Prediction*. Ketiga metode ini kemudian masing-masing parameternya akan di analisis agar memperoleh Sistem *Profiling* yang optimal. Sistem Filtering CNN dianalisis dengan mengubah parameter *learning rate*-nya. Dimana *learning rate* yang dipakai adalah 0.1, 0.01, 0.001, dan 0.0001. Hal ini dilakukan untuk mempertimbangkan learning rate yang optimal yang akan digunakan untuk melakukan training model filter.

Untuk menguji analisis payload CNN pada jenis data yang berbeda maka diperlukan beberapa tambahan data lain. Pada analisis data CNN diberikan payload tambahan yakni file executable yang berisi reverse meterpreter, dan file executable sebelum diisikan reverse meterpreter. Berikut ini adalah daftar file executable yang digunakan sebagai tambahan :

Dapat diamati pada tabel 3.1, sebagian besar file executable adalah file dengan jenis executable windows. Hal ini dilakukan untuk mempermudah memasukkan data reverse meterpreter dalam file tersebut.

3.6.3 Membuat klasifikasi data

Data kelas terbagi menjadi 2 yakni *Malicious* dan *Benign*. Untuk membedakan kedua kelas ini dilakukan training pada data *malicious* dengan dua kelas. Pada sistem

Nama Program	**Nama File**
Calculator	calc.exe
Cruel game	cruel.exe
Freecell game	freecell.exe
Golf game	golf.exe
MS Paint	mspaint.exe
Pegged	pegged.exe
Realterm	realterm.exe
Reversi game	reversi.exe
Snake game	snake.exe
Solitaire game	sol.exe
Taipei game	taipei.exe
Winmine game	winmine.exe

Tabel 3.1: Dataset tambahan

filtering klasifikasi terjadi berdasarkan sifat dari payload dari *packet*, sedangkan pada sistem profiling klasifikasi terjadi secara tidak langsung, dengan memperhitungkan anomali trafik jaringan dengan regresi.

BAB 4

HASIL PENELITIAN

4.1 Deskripsi Hasil Penelitian

Hasil penelitian terdiri dari data hasil training CNN, LSTM, dan data hasil testing pada snortIDS. Data Hasil dari LSTM dan CNN merupakan data hasil Training, Testing, dan hasil prediksi dari model *Neural Network*.

Data hasil yang diperoleh dari proses *filtering* dengan metode CNN adalah akurasi dan loss data per paket yang dihitung berdasarkan konvolusi dari nilai *raw packet*. Sedangkan data hasil yang diperoleh dari proses *profiling* dengan metode LSTM adalah data keseluruhan paket header yang diperhitungkan secara sekuensial atau regresi. Data hasil pada model LSTM merupakan nilai pendekatan dari trafik asli.

Sebelum mengolah data menjadi input *neural network*, perlu dilakukan konversi data dari data mentah menjadi data yang dapat dibaca. Dataset yang digunakan diambil file pcap dan file executable nya.

Data yang dijadikan input pada sisi CNN adalah seluruh data yang masuk pada host. Sedangkan data yang dijadikan input pada sisi LSTM adalah seluruh data header baik keluar maupun masuk dari host. Proses yang dilakukan untuk memperoleh data hasil adalah dengan menggunakan SnortIDS mengirimkan 10 parameter header dengan payloadnya.

Pada sisi *webservice* dengan menggunakan *flask* akan didefinisikan buffer pada setiap data yang masuk berdasarkan sumber alamat IP nya. Buffer untuk IP ini terbagi menjadi 3 yakni, buffer transmit, receive, dan transceive. Buffer transmit, dan receive yang berkaitan akan digabungkan untuk mendeteksi intrusi pada sisi packet header dengan LSTM. Dan buffer transmit pada setiap IP akan digunakan untuk mendeteksi intrusi pada sisi packet payload dengan CNN.

4.2 Hasil Training Data Neural Network

4.2.1 Model CNN

Berdasarkan gambar 4.1, pada model CNN terdapat layer Embedding digunakan untuk mengekstraksi fitur dari data input dengan ukuran 2xMTU atau 3000 Byte. Layer selanjutnya adalah layer Reshape yang mengubah ukuran input dari 3000x50 menjadi 3000x50x1 dengan kata lain menginklusi matriks dari matriks 2 dimensi menjadi matriks 3 dimensi dengan ketebalan 1.

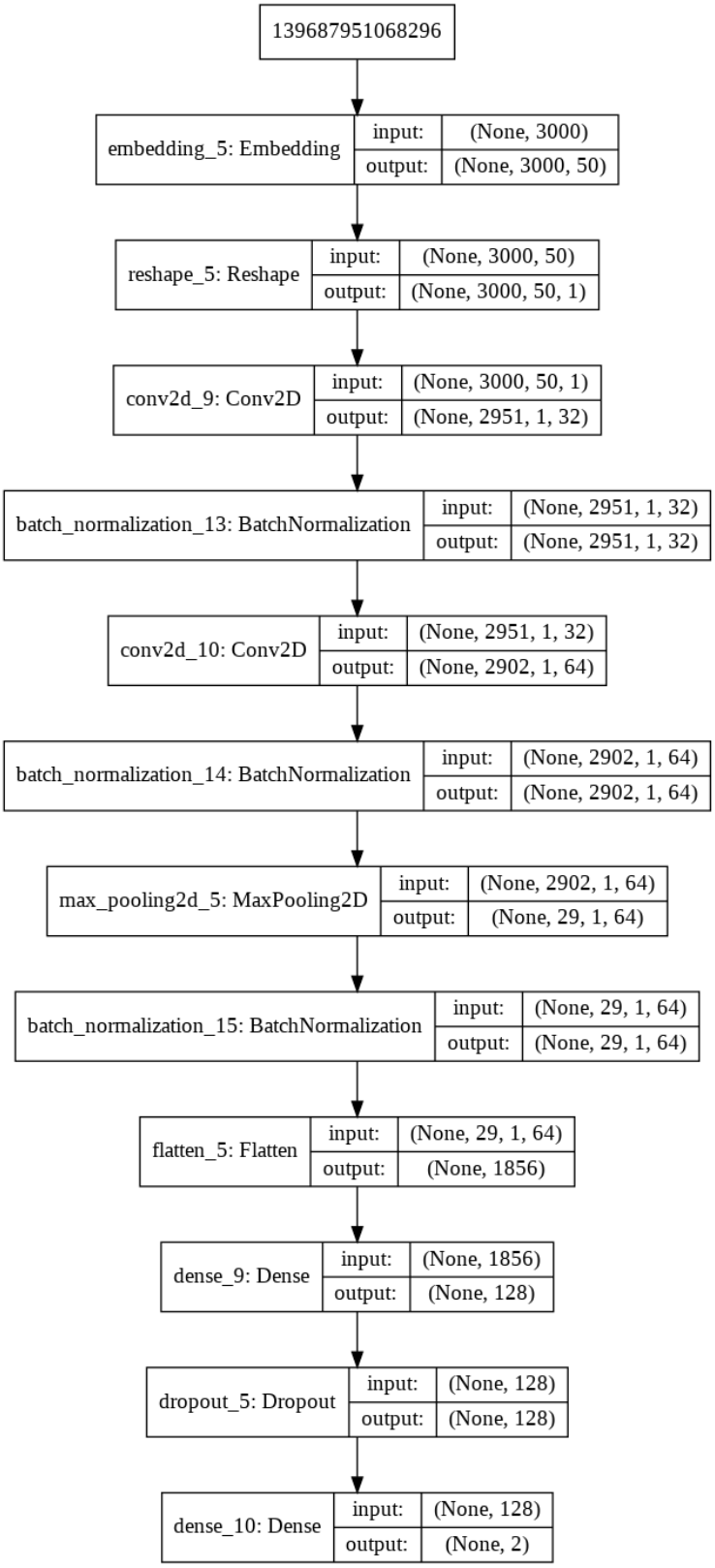
Layer selanjutnya merupakan layer konvolusi, pada konvolusi pertama dan kedua outputnya di normalisasi dengan BatchNormalization, dan pada Layers konvolusi terakhir dilakukan MaxPooling untuk meniadakan interpretasi yang akan mengurangi rate pendeteksian. Setelah semua konvolusi selesai matriks dinormalisasi lagi sebelum masuk ke layer selanjutnya.

Layer selanjutnya merupakan layer untuk deteksi. Pada layer Flatten semua matriks pada layer konvolusi terakhir diubah menjadi vektor. Kemudian setelah itu masuk ke layer Dense atau untuk aktivasi pertama lalu masuk ke Dropout untuk mengurangi neuron irrelevan dari Dense sebelumnya. Kemudian keluaran dari Dropout Layer menjadi input bagi Layer Dense terakhir yang berukuran vektor 2 bit, atau memiliki 2 kelas. Dari kedua kelas ini hanya kategori [0 1] yang dilakukan training data.

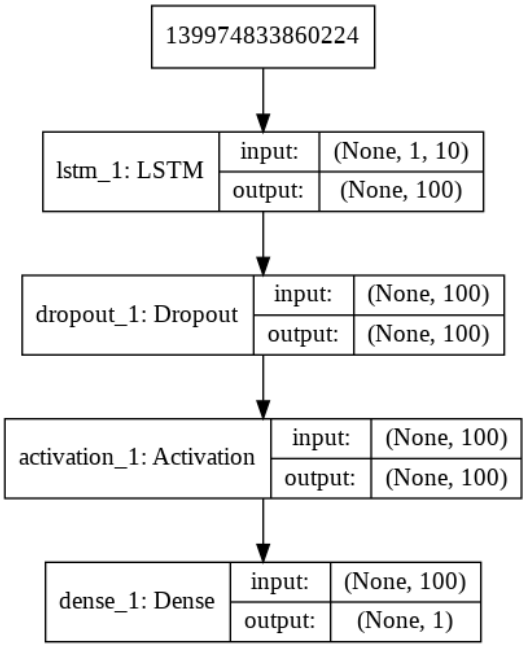
4.2.2 Model LSTM

Pada Gambar 4.2 model LSTM hanya terdapat layer LSTM sebagai pembaca vektor header dari packet. Dropout untuk menghilangkan neuron yang irrelevan. Layer Activation dimana hasil keluaran Layer sebelumnya diaktivasi dengan menggunakan fungsi aktivasi *softmax* dan Dense dengan ukuran klaster 1, yakni memiliki skalar untuk pendeteksiannya.

Untuk proses pendeteksian dengan sentiment analysis neuron Dense berjumlah 1, sedangkan untuk prediksi multivariable neuron Dense berjumlah 10, dimana neuron ini merupakan jumlah parameter pada header packet.



Gambar 4.1: Model CNN



Gambar 4.2: Model LSTM

4.2.3 Hasil Training Data LSTM

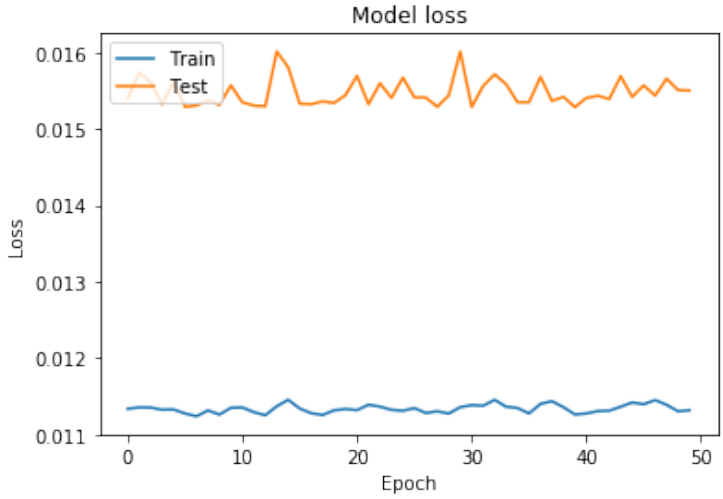
Sistem training data LSTM menggunakan 2 metode yang berbeda yakni *Sentiment Analysis* dan *Multivariate Prediction*. Hasil *training* kedua metode ini memiliki perbedaan dimana pada *Sentiment Analysis* dapat memprediksi apabila variasi dan distribusi variasi pada targetnya merata. Sedangkan pada *Multivariate Prediction* hasil training memiliki karakteristik konvergen yang tinggi karena memperhitungkan dan memprediksi data secara keseluruhan.

4.2.3.1 Hasil training Svchosta Botnet dengan Sentiment Analysis LSTM

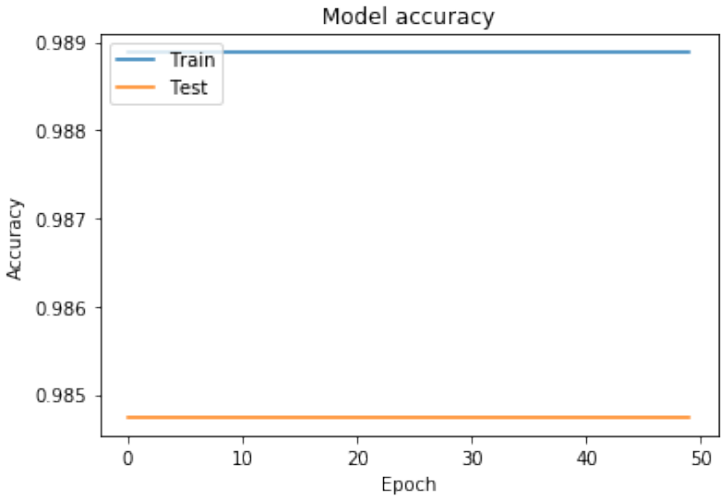
Untuk Hasil training svchosta dengan sentiment analysis memiliki data sebagai berikut :

Dapat diamati pada grafik kisaran akurasi Loss dari model svchosta test ada di antara 0.016 dan 0.015, sedangkan kisaran akurasi Loss dari model svchosta train ada di antara 0.011 dan 0.012. Hasil akurasi diperoleh dapat diamati juga kisaran train berada di sekitar 0.989, dan kisaran test berada di sekitar 0.985.

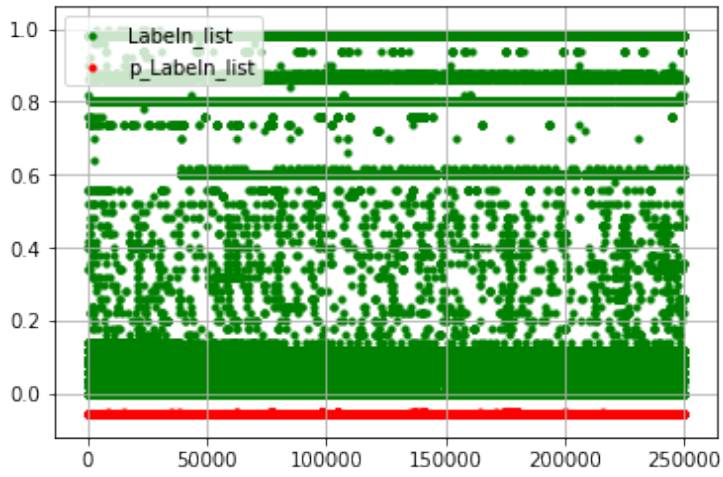
Dari grafik 4.5 dapat diamati bentuk prediksi *LSTM Sentiment Analysis*. Tampak variasi trafik label yang diperoleh di prediksi hanya bernilai 0, sedangkan beberapa label lain tidak terdeteksi sama sekali. Hal ini dikarenakan jarangnya trafik malware



Gambar 4.3: Loss model LSTM svchosta Sentiment Analysis



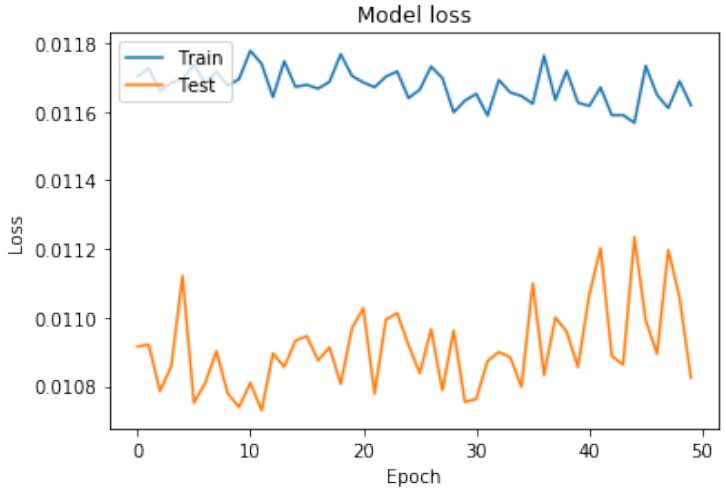
Gambar 4.4: Akurasi model LSTM svchosta Sentiment Analysis



Gambar 4.5: Prediksi model LSTM svchosta Sentiment Analysis

Tabel 4.1: Tabel Hasil LSTMS Svchosta

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.015	0.985	0.015	1.000	0.011	0.989	0.011	1.000
5	0.015	0.985	0.015	1.000	0.011	0.989	0.011	1.000
10	0.015	0.985	0.015	1.000	0.011	0.989	0.011	1.000
15	0.015	0.985	0.015	1.000	0.011	0.989	0.011	1.000
20	0.016	0.985	0.016	1.000	0.011	0.989	0.011	1.000
25	0.015	0.985	0.015	1.000	0.011	0.989	0.011	1.000
30	0.015	0.985	0.015	1.000	0.011	0.989	0.011	1.000
35	0.015	0.985	0.015	1.000	0.011	0.989	0.011	1.000
40	0.015	0.985	0.015	1.000	0.011	0.989	0.011	1.000
45	0.016	0.985	0.016	1.000	0.011	0.989	0.011	1.000



Gambar 4.6: Loss model LSTM Neris Sentiment Analysis

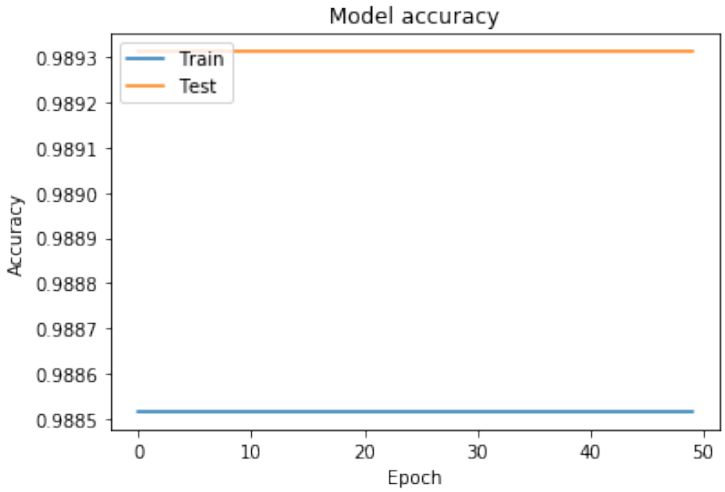
yang ada pada jaringan. Pemberian angka dilakukan pada setiap label yang berbeda. Nilai 0 diperoleh dikarenakan label yang dianalisis bukan merupakan label dari botnet.

Berdasarkan tabel 4.1 dapat dilihat tidak terjadi konvergensi pada (Value Loss) VL, (Loss) L, maupun parameter (A) Akurasi. Nilai yang diperoleh di sisi Loss bernilai 0.011 dan akurasi 0.989. Tidak terjadi perubahan setiap epoch. Beberapa parameter terjadi sedikit fluktuasi pada sisi prediksi.

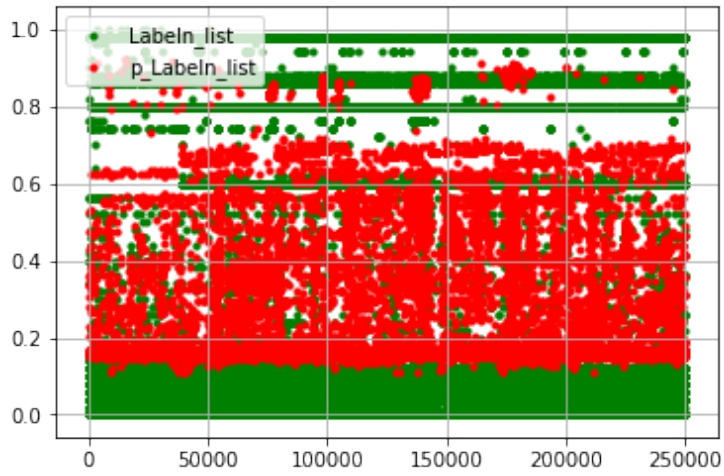
4.2.3.2 Hasil Training Neris Botnet dengan Sentiment Analysis LSTM

Untuk Hasil training Neris dengan sentiment analysis memiliki data sebagai berikut :

Berdasarkan grafik 4.7 dan **??** dapat dilihat bahwa pola grafik tidak menunjukkan perubahan, hal ini berarti tidak terjadi proses pembelajaran. Tidak adanya proses pembelajaran ini disebabkan karena jumlah data yang bernilai malicious pada



Gambar 4.7: Akurasi model LSTM neris Sentiment Analysis



Gambar 4.8: Prediksi model LSTM neris Sentiment Analysis

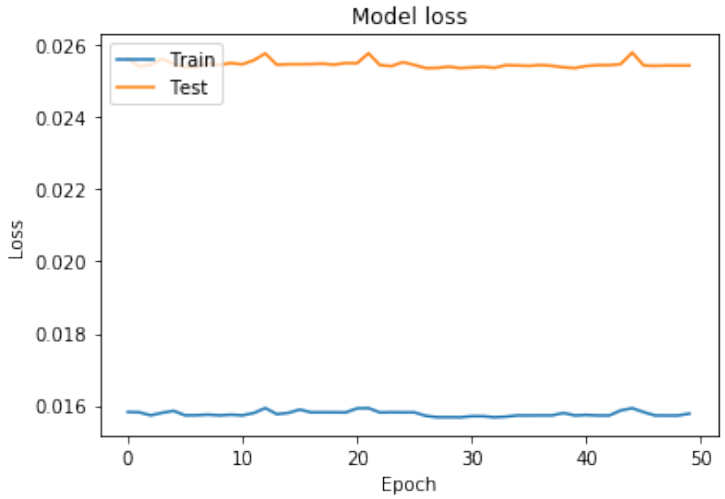
header sangat kecil atau tidak seimbang dengan data yang bernilai normal. Hal ini menyebabkan data dapat diabaikan.

Pada grafik 4.8 dapat damati bentuk prediksi *LSTM Sentiment Analysis*. Dimana pada sisi loss model dari hasil testing berada di kisaran 0.0108 sampai 0.0112, sedangkan loss di hasil training berada di kisaran 0.0118, dan 0.0116. Pada sisi akurasi model dari hasil testing berada di kisaran 0.9893, sedangkan dari hasil training berada di kisaran 0.9885. Tidak terjadi perubahan pada kedua grafik ini dikarenakan tidak terjadiny a pembelajaran pada model ini.

Dapat dilihat pada tabel 4.2 tidak terjadi konvergensi pada (Value Loss) VL, (Loss) L, maupun parameter (A) Akurasi. Hal ini disebabkan karena tidak terjadi

Tabel 4.2: Tabel Hasil LSTMS Neris

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
5	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
10	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
15	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
20	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
25	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
30	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
35	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
40	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000
45	0.011	0.989	0.011	1.000	0.012	0.989	0.012	1.000



Gambar 4.9: Loss model LSTM Rbot Sentiment Analysis

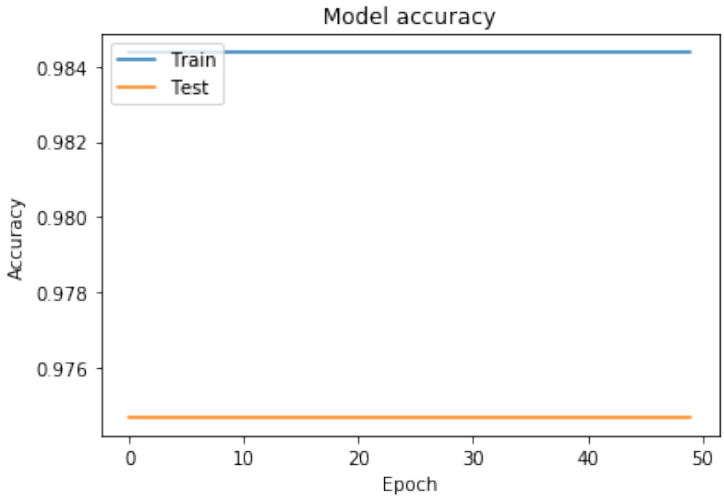
pembelajaran data dikarenakan dataset memiliki label yang tidak seimbang.

4.2.3.3 Hasil Training Rbot Botnet dengan Sentiment Analysis LSTM

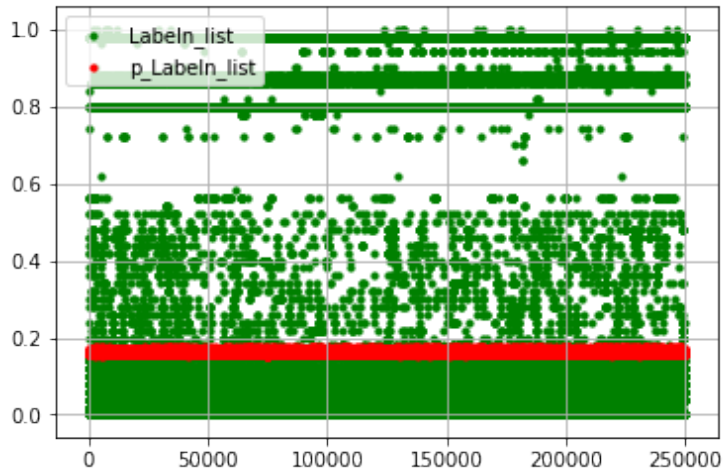
Untuk Hasil training Rbot dengan sentiment analysis memiliki data sebagai berikut :

Berdasarkan dari analisis grafik 4.9 dan 4.10 dapat diamati bahwa tidak adanya pembelajaran. Dapat diamati bahwa tidak ada perubahan nilai baik pada akurasi maupun pada loss dari model. Hal ini disebabkan karena data yang bernilai malicious atau intrusi pada *header* sangat kecil sehingga dapat diabaikan.

Dapat diamati bentuk prediksi dari grafik 4.11 yang menunjukkan hasil prediksi *LSTM Sentiment Analysis*. Karena saat training tidak terjadi konvergensi, sehingga data prediksi yang diperoleh tidak menunjukkan kecocokan antara hasil prediksi dengan data yang seharusnya.



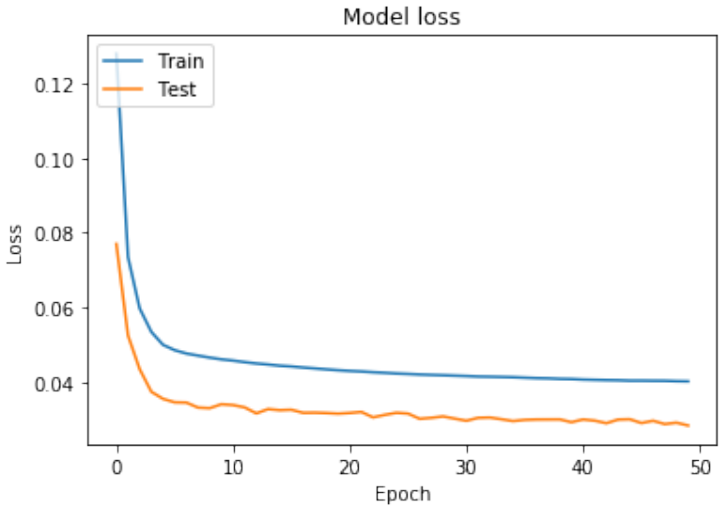
Gambar 4.10: Akurasi model LSTM rbot Sentiment Analysis



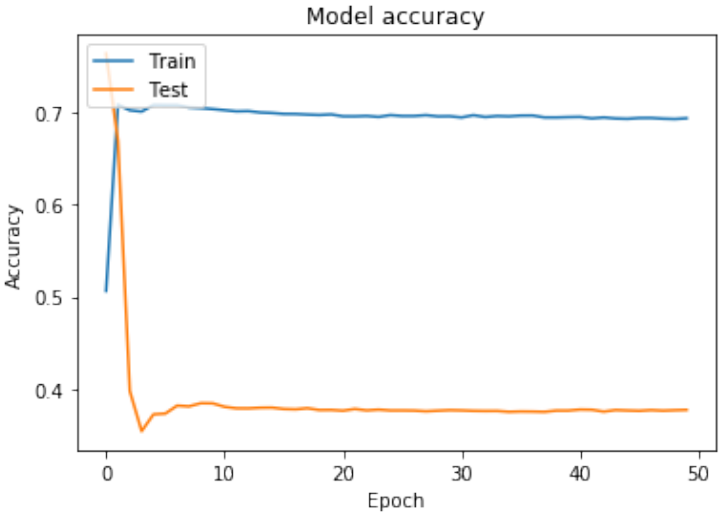
Gambar 4.11: Prediksi model LSTM rbot Sentiment Analysis

Tabel 4.3: Tabel Hasil LSTMS RBot

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.026	0.975	0.026	1.000	0.016	0.984	0.016	1.000
5	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000
10	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000
15	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000
20	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000
25	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000
30	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000
35	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000
40	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000
45	0.025	0.975	0.025	1.000	0.016	0.984	0.016	1.000



Gambar 4.12: Loss model LSTM svchosta Sentiment Analysis



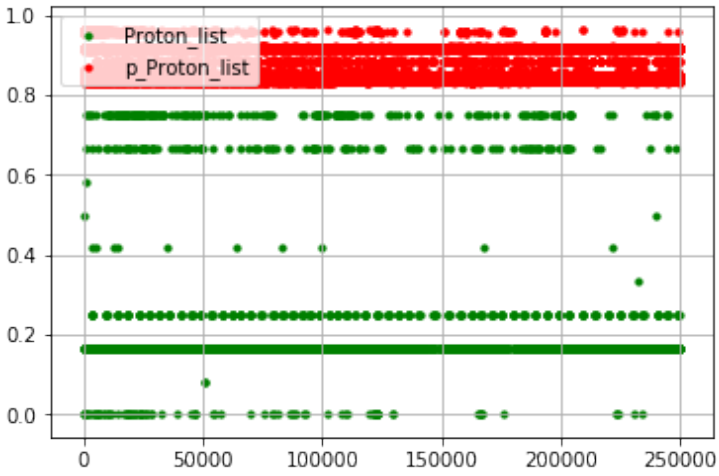
Gambar 4.13: Akurasi model LSTM svchosta Sentiment Analysis

Tabel 4.3 menunjukkan tidak adanya konvergensi pada (Value Loss) VL, (Loss) L, maupun parameter (A) Akurasi. Hal ini disebabkan karena tidak terjadi pembelajaran data dikarenakan dataset memiliki label yang tidak seimbang atau data malicious yang sangat jarang.

4.2.3.4 Hasil Training svchosta Botnet dengan Multivariate Prediction LSTM

Untuk Hasil training svchosta dengan multivariate prediction memiliki data sebagai berikut :

Dapat dilihat bahwa grafik menjelaskan bahwa hasil training konvergen dan



Gambar 4.14: Grafik Prediksi Proto Svchosta

memiliki loss terendah sebesar 0.03 pada sisi testing dan 0.05 pada sisi training. Dapat dilihat juga dari grafik yang mengalami pemusatan pada satu sumbu y baik pada grafik *Loss* maupun pada grafik *Accuracy* berbeda dari metode sebelumnya.

Parameter yang diukur memiliki karakteristik prediksi sebagai berikut :

1. Proto

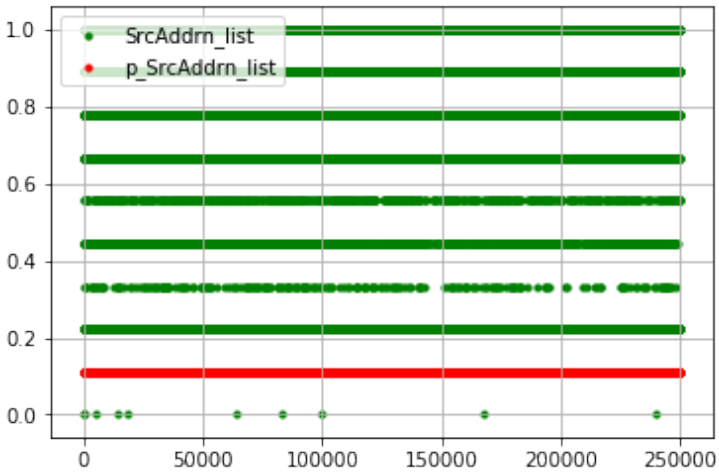
   Bagian 4.14 ini merupakan salah satu bagian dari paket jaringan yang mendefinisikan jenis protokol jaringan yang digunakan. Dapat dilihat dari grafik bahwa sebagian besar hasil prediksi berada di atas tapi memiliki pola yang mirip dengan trafik asli.

   Titik hijau merupakan data asli sedangkan titik merah merupakan data hasil prediksi model. Proses perubahan data proto menjadi grafik dilakukan dengan metode tokenisasi, dimana setiap data berbeda dilabeli dengan angka dan di sesuaikan dengan jumlah variasi datanya.

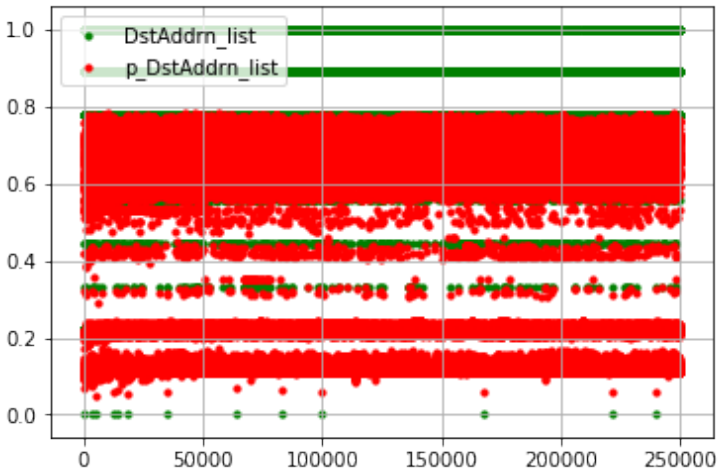
2. SrcAddr

   Gambar 4.15 merupakan bagian IP header dari paket jaringan yang mendefinisikan alamat IP sumber. Dapat dilihat dari grafik bahwa sebagian besar hasil prediksi hanya mencakup nilai rendah saja, sedangkan trafik asli tersebar.

   Dapat diamati pada gambar bahwa titik hijau menunjukkan data asli sedangkan titik merah menunjukkan data prediksi. Proses pemetaan ip ke grafik juga



Gambar 4.15: Grafik Prediksi SrcAddr Svchosta



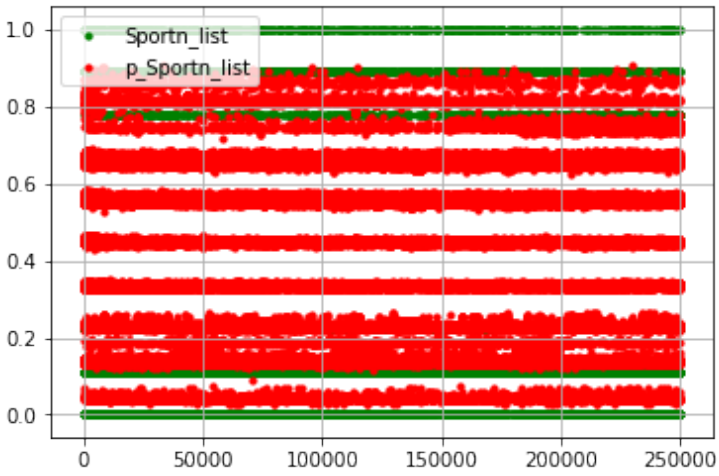
Gambar 4.16: Grafik Prediksi DstAddr Svchosta

sama dengan pada proto, menggunakan metode tokenisasi dimana data unik akan diberi label nilai dari rentang 0 sampai 1 berdasarkan jumlah variasinya.

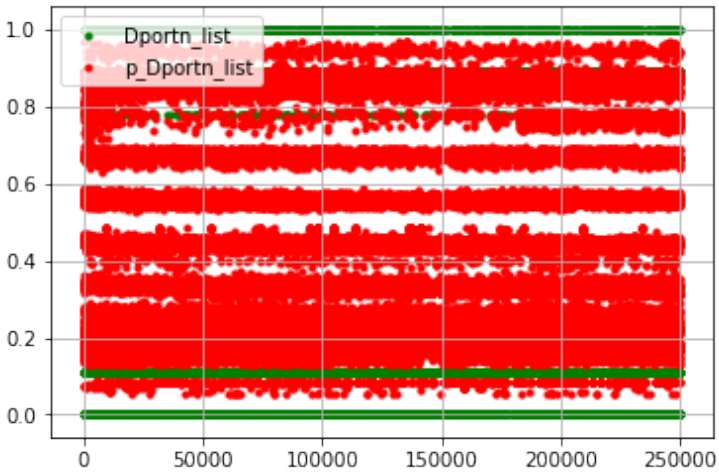
3. DstAddr

   Pada gambar 4.16 merupakan bagian IP header dari paket jaringan yang mendefinsikan alamat IP tujuan. Dapat dilihat dari grafik bahwa sebagian besar hasil prediksi dapat mencakup keseluruhan trafik asli.

   Titik merah menunjukkan data prediksi, sedangkan titik hijau menunjukkan data trafik asli. Proses pemetaan juga sama seperti sebelumnya dengan metode tokenisasi.



Gambar 4.17: Grafik Prediksi Sport Svchosta



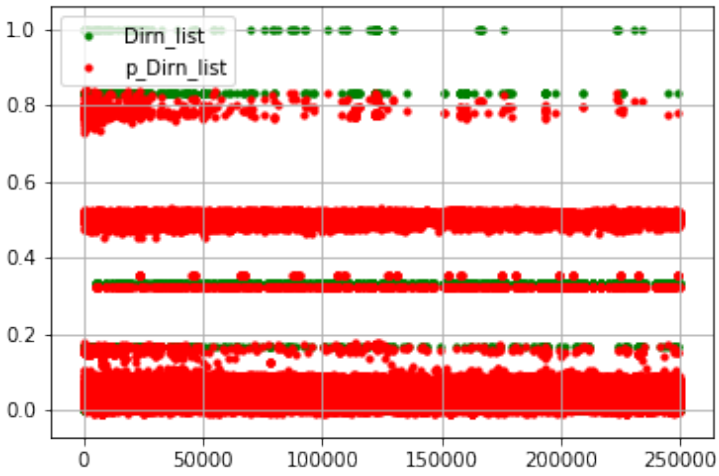
Gambar 4.18: Grafik Prediksi Dport Svchosta

4. Sport

   Pada gambar 4.17 menjelaskan persebaran data dari IP header yang mendefinisikan port sumber. Dapat dilihat dari grafik hasil pendeteksian port sumber mencakup keseluruhan, walaupun ada beberapa bagian yang tidak dicakup terutama dibagian nilai tertinggi.

   Titik hijau menunjukkan data trafik asli sedangkan titik merah menunjukkan data trafik hasil prediksi. Proses pemetaan menjadi grafik juga menggunakan metode tokenisasi.

5. Dport



Gambar 4.19: Grafik Prediksi Dir Svchosta

Gambar 4.18 ini merupakan bagian dari IP header yang mendefinisikan port tujuan. Dapat dilihat dari grafik hasil pendeteksian port tujuan bahwa hasil pendeteksian mencakup keseluruhan trafik port tujuan, sementara beberapa bagian sekitar angka port tertinggi dan terendah tidak ada di hasil prediksi.

Titik hijau dan titik merah menunjukkan data asli dan hasil prediksi sama seperti sebelumnya. Proses pemetaan juga menggunakan metode tokenisasi.

6. Dir

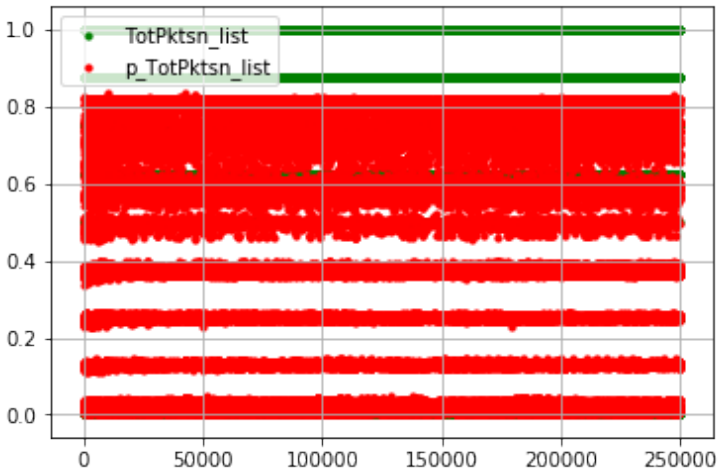
   Bagian 4.19 ini merupakan arah yang menunjukan arah transmisi dari paket baik itu multicast, maupun unicast. Dapat dilihat dari grafik hasil pendeteksian dapat mencakup hampir keseluruhan trafik kecuali kategori 1.0.

   Titik merah menunjukkan prediksi dan titik hijau menunjukkan data asli. Pemetaan pada data ini juga menggunakan metode tokenisasi.

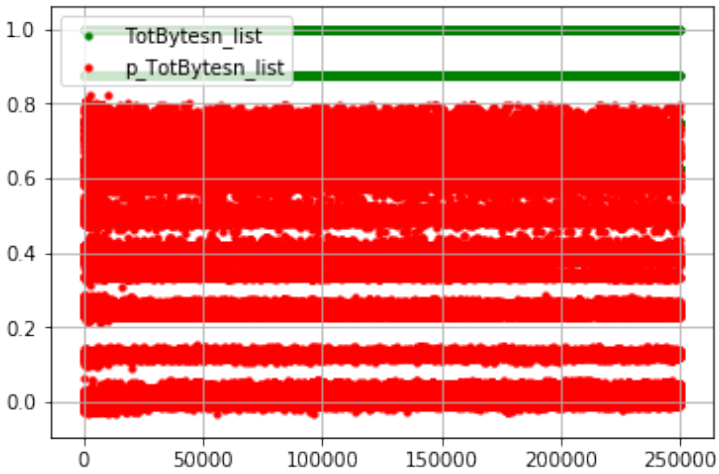
7. TotPkts

   Grafik 4.20 ini menjelaskan bagian dari paket header yang menjelaskan jumlah paket dalam suatu ethernet frame yang dikirim per paket. Dari grafik dapat dilihat variasi total paket hasil prediksi sebagian besar dapat terdeteksi sementara bagian variasi kelas ukuran disekitar 1.0 dan 0.9 tidak terdeteksi.

   Berdasarkan grafik 4.20 dapat diamati titik merah yang merupakan data hasil prediksi dan titik hijau yang merupakan data asli. Proses pemetaan juga



Gambar 4.20: Grafik Prediksi TotPkts Svchosta



Gambar 4.21: Grafik Prediksi TotBytes Svchosta

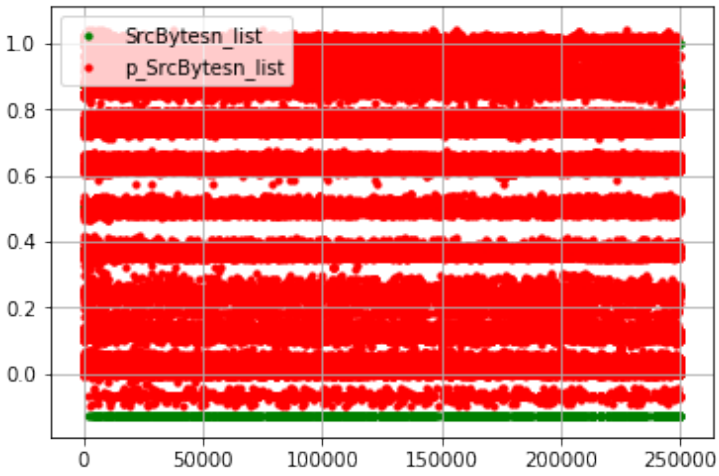
melalui metode tokenisasi.

8. TotBytes

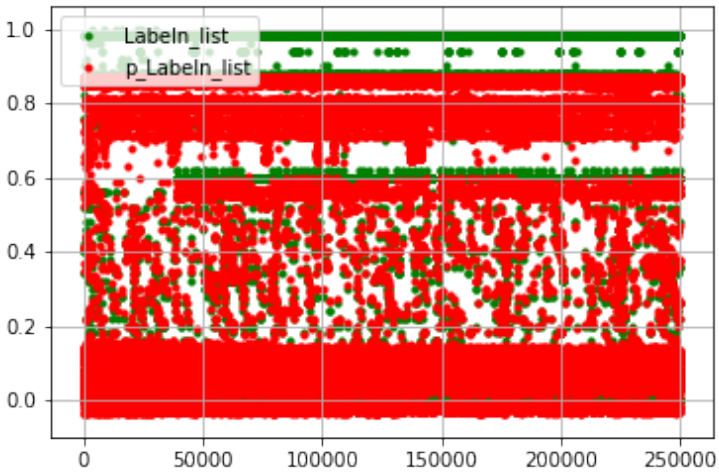
   Bagian ini merupakan bagian dari paket header yang menjelaskan jumlah bytes dalam satu ethernet frame yang dikirim per paket. Dari grafik dapat dilihat variasinya menyerupai totpkts, sedangkan hasil prediksi memiliki cakupan yang sangat besar menutupi sebagian besar kelas total byte rendah.

9. SrcBytes

   Pada bagian **??** ini merupakan bagian dari paket header yang menjelaskan ukuran byte source. Dapat dilihat trafik yang terdeteksi mencakup sebagian



Gambar 4.22: Grafik Prediksi SrcBytes Svchosta



Gambar 4.23: Grafik Prediksi Label Svchosta

besar kecuali sisi kelas terendah atau bawah yang bernilai 0.1.

Titik hijau menjelaskan data asli dan titik merah data prediksi. Proses pemetaan sama seperti sebelumnya yakni dengan metode tokenisasi.

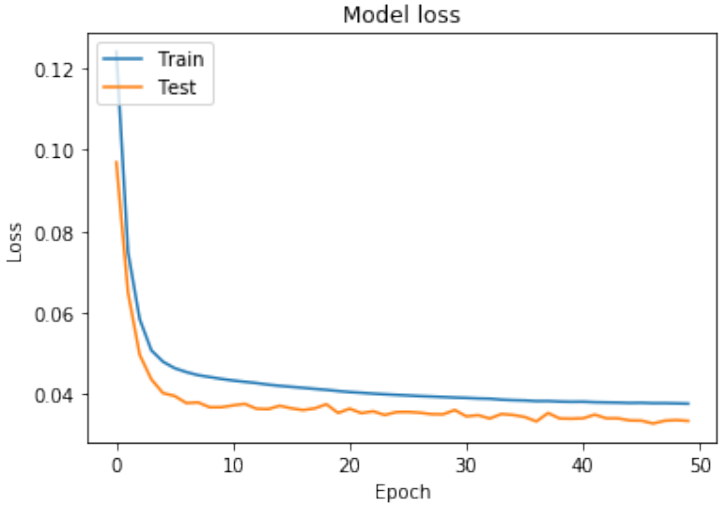
10. Label

    Bagian ini merupakan label dari dataset yang sudah di angka kan. Dapat dilihat dari grafik bahwa sebagian besar terdeteksi sedangkan kelas label tertinggi atau sekitar 0.9 sampai 1.0 tidak terdeteksi.

    Titik merah menunjukkan data prediksi sedangkan titik hijau menunjukkan data asli. Proses pemetaan juga melalui proses tokenisasi

Tabel 4.4: Tabel Hasil LSTMM Svc

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.077	0.764	0.077	0.764	0.128	0.506	0.128	0.506
5	0.035	0.373	0.035	0.373	0.048	0.708	0.048	0.708
10	0.034	0.381	0.034	0.381	0.046	0.703	0.046	0.703
15	0.033	0.378	0.033	0.378	0.044	0.699	0.044	0.699
20	0.032	0.376	0.032	0.376	0.043	0.696	0.043	0.696
25	0.031	0.377	0.031	0.377	0.042	0.696	0.042	0.696
30	0.030	0.377	0.030	0.377	0.042	0.695	0.042	0.695
35	0.030	0.376	0.030	0.376	0.041	0.697	0.041	0.697
40	0.030	0.378	0.030	0.378	0.041	0.695	0.041	0.695
45	0.029	0.376	0.029	0.376	0.040	0.694	0.040	0.694

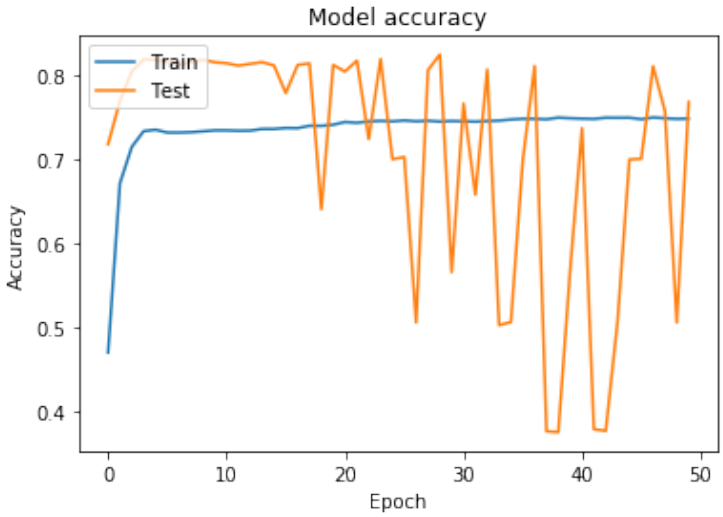


Gambar 4.24: Loss model LSTM neris Multivariate Prediction

Berdasarakan grafik keseluruhan dari hasil prediksi kesepuluh parameter *LSTM Multivariate Prediction*, dapat dilihat bahwa pada beberapa parameter prediksi terdapat beberapa parmaeter yang berhasil di prediksi dan beberapa parameter pula yang tidak dapat terdeteksi.

Berdasarkan tabel 4.4 diatas dapat diamati bahwa penurunan Loss terjadi dan peningkatan terjadi. Pada sisi akurasi dapat dilihat bahwa akurasi tertinggi yang dicapai model adalah 0.7. Hal ini dikarenakan beberapa parameter tidak mempengaruhi satu sama lain, namun berdasarkan keseluruhan hasil prediksi yang diperoleh dapat digunakan untuk melakukan profiling.

Metode pemetaan dan keterangan titik juga berlaku pada data hasil training neris botnet dan rbot botnet.



Gambar 4.25: Akurasi model LSTM neris Multivariate Prediction

4.2.3.5 Hasil Training Neris Botnet dengan Multivariate Prediction LSTM

Berdasarkan dari grafik diatas dapat diamati bahwa hasil training konvergen. Loss terendah yang dicapai pada sisi testing berada disekitar 0.03, sedangkan loss terendah dicapai pada sisi training adalah sekitar 0.05.

Dapat dilihat dari grafik yang mengalami pemusatan pada satu sumbu y baik pada grafik *Loss* maupun pada grafik *Accuracy*. Dapat dilihat juga dari grafik akurasi pada hasil testing memiliki fluktuasi akurasi tapi berada disekitar garis konvergensi training yang berada di antara nilai 0.7 dan 0.8.

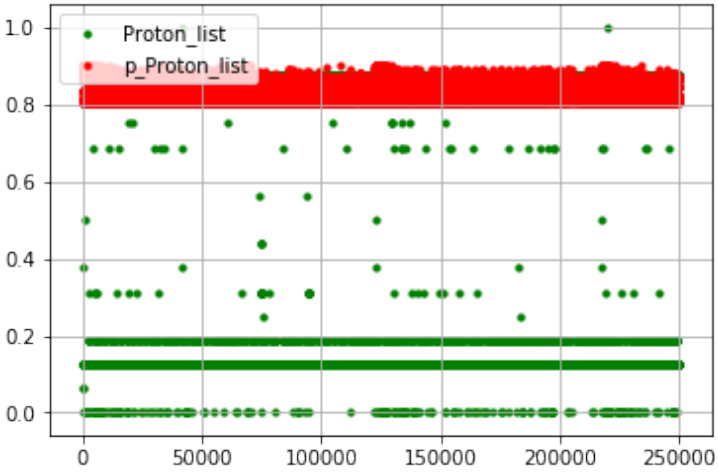
Parameter yang diukur memiliki karakteristik prediksi sebagai berikut :

1. Proto

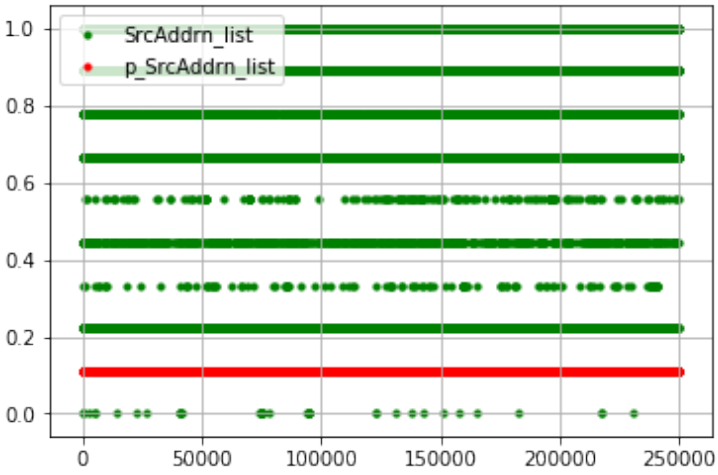
   Header Proto merupakan salah satu bagian dari paket jaringan yang mendefinisikan jenis protokol jaringan yang digunakan. Dapat dilihat dari grafik prediksi mencakup sebagaian besar nilai tertinggi diantara 0.8dan 1.0, sedangkan beberapa titik dibawah tidak tedeteksi.

2. SrcAddr

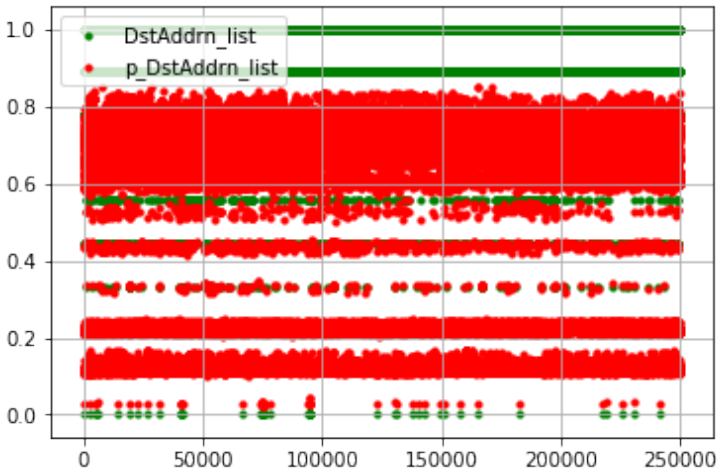
   Header SrcAddr merupakan bagian IP header dari paket jaringan yang mendefinisikan alamat IP sumber. Dapat dilihat dari grafik bahwa sebagian besar hasil prediksi hanya mencakup nilai rendah disekitar 0.1 saja, sedangkan trafik asli



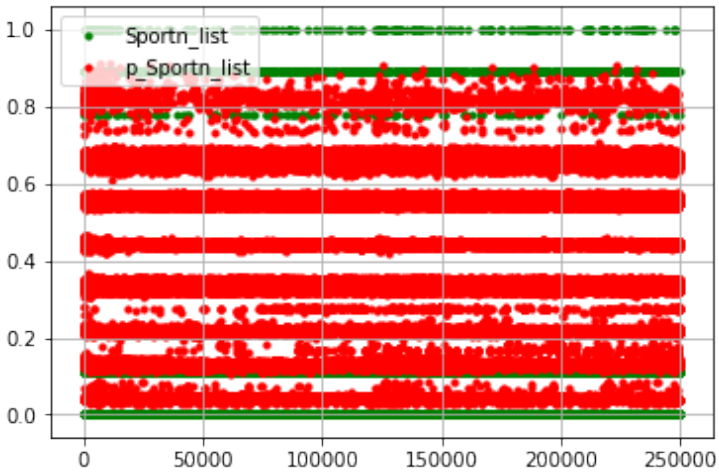
Gambar 4.26: Grafik Prediksi Proto neris



Gambar 4.27: Grafik Prediksi SrcAddr neris



Gambar 4.28: Grafik Prediksi DstAddr neris



Gambar 4.29: Grafik Prediksi Sport neris

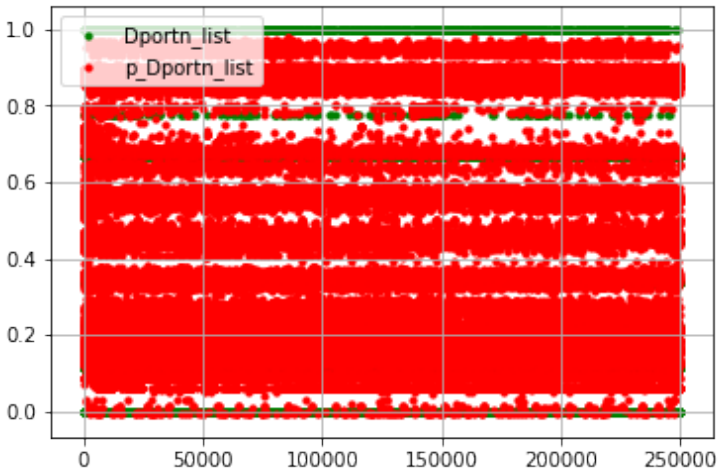
tersebar merata dengan interval 0.1.

3. DstAddr

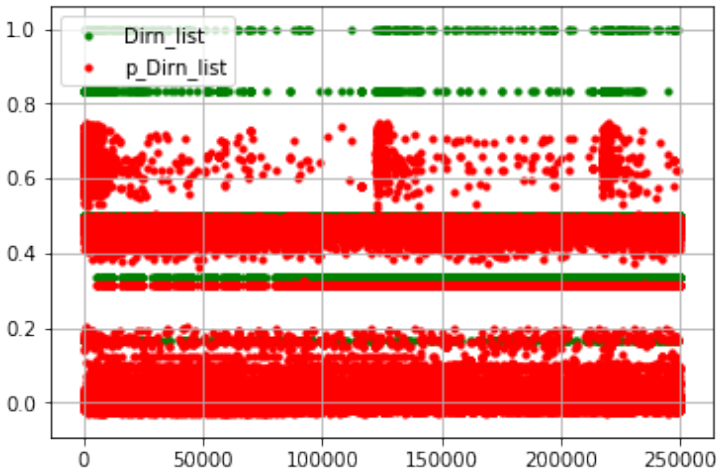
   Header DstAddr ini merupakan bagian IP header dari paket jaringan yang mendefinsikan alamat IP tujuan. Dapat dilihat dari grafik bahwa sebagian besar hasil prediksi dapat mencakup keseluruhan trafik asli dan dibagian antara 1.0 dan 0.8 tidak terdeteksi.

4. Sport

   Header Sport ini merupakan bagian dari IP header yang mendefinisikan port sumber. Dapat dilihat dari grafik hasil pendeteksian port sumber mencakup



Gambar 4.30: Grafik Prediksi Dport neris



Gambar 4.31: Grafik Prediksi Dir neris

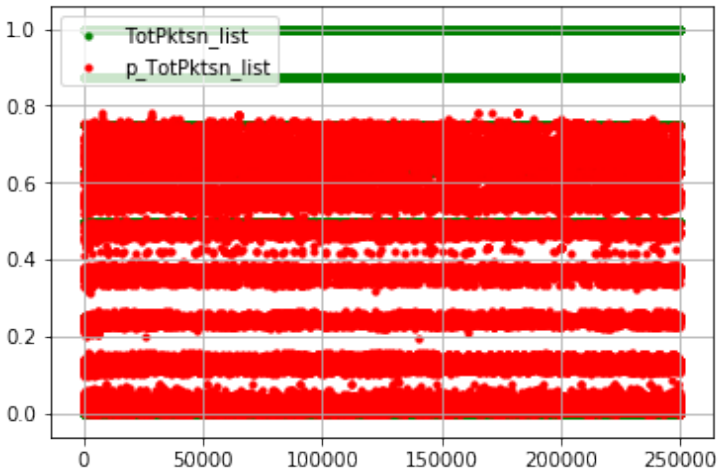
keseluruhan, walaupun ada beberapa bagian yang tidak dicakup terutama dibagian nilai tertinggi yakni disekitar 0.0 dan sekitar 1.0.

5. Dport

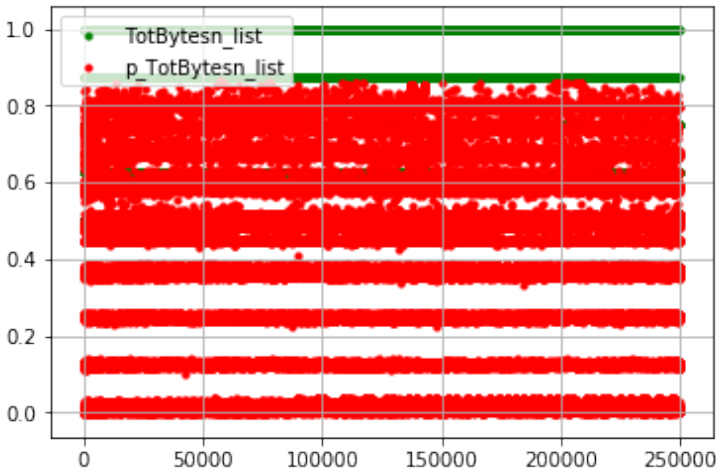
   Header Dport ini merupakan bagian dari IP header yang mendefinisikan port tujuan. Dapat dilihat dari grafik prediksi bahwa cakupan data prediksi mencakup keseluruhan sedangkan beberapa bagian yakni sekitar 1.0 dan 0.8 sebagian besar tidak terdeteksi

6. Dir

   Header Dir merupakan arah yang menunjukan arah transmisi dari paket baik



Gambar 4.32: Grafik Prediksi TotPkts neris



Gambar 4.33: Grafik Prediksi TotBytes neris

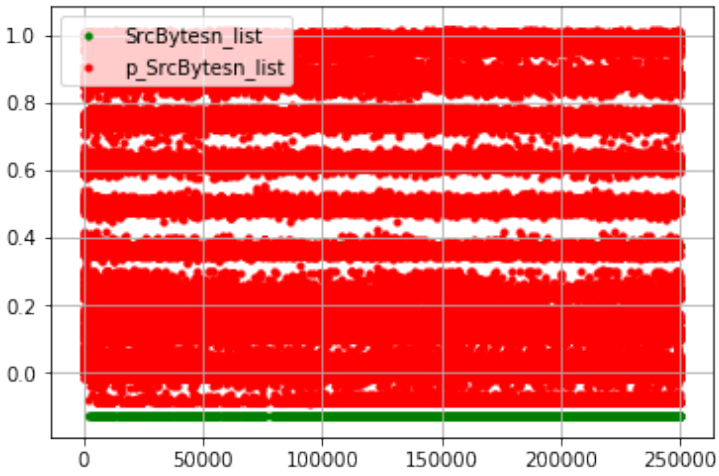
itu multicast, maupun unicast. Dapat dilihat dari grafik hasil prediksi bahwa keseluruhan trafik dicakup kecuali disekitar kelas 0.8 dan 1.0.

7. TotPkts

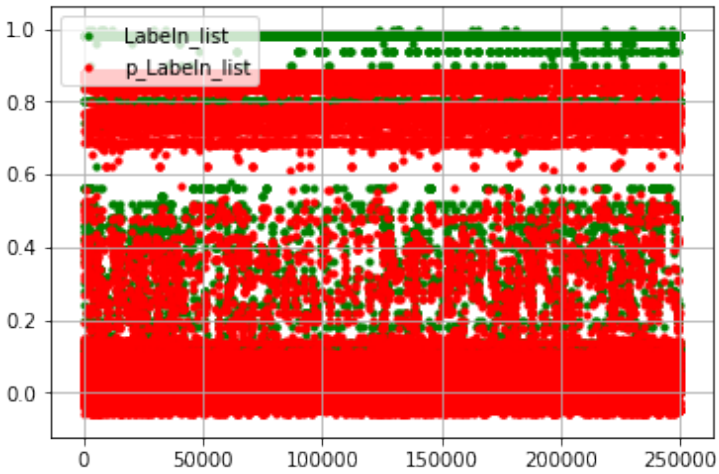
   Header TotPkts merupakan bagian dari paket header yang menjelaskan jumlah paket dalam suatu ethernet frame yang dikirim per paket. Dari grafik hasil prediksi dapat diamati bahwa keseluruhan variasi totpkts teredeteksi sedangkan diukuran sekitar 0.8 dan 1.0 tidak terdeteksi.

8. TotBytes

   Header TotBytes merupakan bagian dari paket header yang menjelaskan jum-



Gambar 4.34: Grafik Prediksi SrcBytes neris



Gambar 4.35: Grafik Prediksi Label neris

lah bytes dalam satu ethernet frame yang dikirim per paket. Dapat dilihat dari grafik hasil prediksi diatas bahwa sebagian besar prediksi mencakup semua kecuali dibagian 0.8 dan 1.0.

9. SrcBytes

   Header SrcBytes merupakan bagian dari paket header yang menjelaskan ukuran byte source. Dapat diamati dari grafik diatas bahwa keseluruhan hasil prediksi mencakup semua trafik kecuali dibagian terendah.

10. Label

    Pada bagian ini berisi label dari dataset yang sudah di angka kan. Dapat

Tabel 4.5: Tabel Hasil LSTMM Neris

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.065	0.655	0.065	0.655	0.120	0.450	0.120	0.450
5	0.035	0.734	0.035	0.734	0.046	0.741	0.046	0.741
10	0.031	0.442	0.031	0.442	0.040	0.743	0.040	0.743
15	0.029	0.388	0.029	0.388	0.038	0.742	0.038	0.742
20	0.029	0.376	0.029	0.376	0.036	0.739	0.036	0.739
25	0.028	0.374	0.028	0.374	0.035	0.741	0.035	0.741
30	0.028	0.381	0.028	0.381	0.034	0.741	0.034	0.741
35	0.028	0.385	0.028	0.385	0.034	0.740	0.034	0.740
40	0.028	0.383	0.028	0.383	0.033	0.739	0.033	0.739
45	0.029	0.377	0.029	0.377	0.033	0.739	0.033	0.739

dilhat dari grafik diatas bahwa ukuran yang terdeteksi kecuali dibagian trafik tertinggi diantara 0.9 dan 1.0 dan diantara 0.6.

Berdasarakan grafik keseluruhan dari hasil prediksi kesepuluh parameter *LSTM Multivariate Prediction*, dapat dilihat bahwa pada beberapa parameter prediksi terdapat beberapa parmaeter yang berhasil di prediksi dan beberapa parameter pula yang tidak dapat terdeteksi.

Berdasarakan grafik diatas keseluruhan dari hasil prediksi kesepuluh parameter *LSTM Multivariate Prediction*, dapat dilihat bahwa pada beberapa parameter prediksi terdapat beberapa parmaeter yang berhasil di prediksi dan beberapa parameter pula yang tidak dapat terdeteksi.

Berdasarkan tabel 4.5 diatas dapat diamati bahwa penurunan Loss terjadi dan peningkatan terjadi. Pada sisi akurasi dapat dilihat bahwa akurasi tertinggi yang dicapai model adalah 0.75. Hal ini dikarenakan beberapa parameter tidak mempengaruhi satu sama lain, namun berdasarkan keseluruhan hasil prediksi yang diperoleh dapat digunakan untuk melakukan profiling.

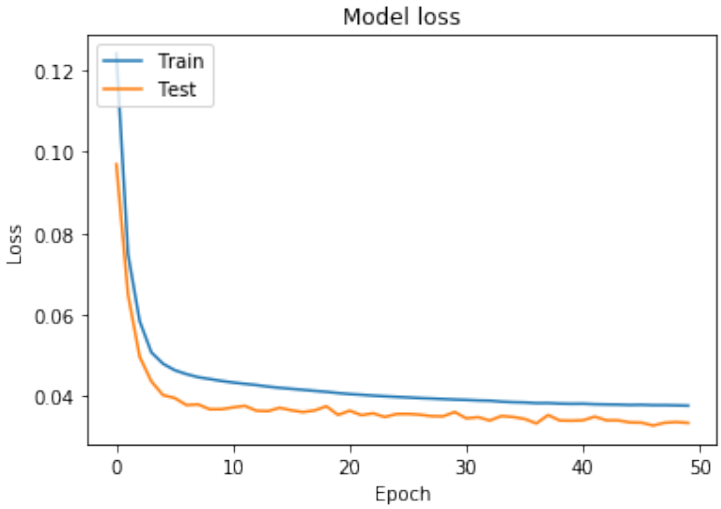
4.2.3.6 Hasil Training Rbot Botnet dengan Multivariate Prediction LSTM

Berdasarkan dari grafik diatas dapat diamati bahwa hasil training konvergen diantara 0.04 pada grafik testing dan 0.05 pada grafik training.

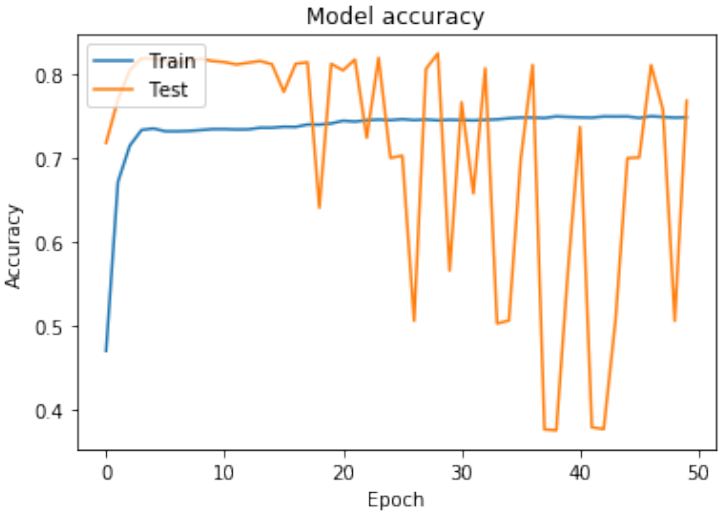
Dapat dilihat dari grafik loss mengalami pemusatan pada satu sumbu y baik pada grafik *Loss* maupun pada grafik *Accuracy*. Pada grafik akurasi terjadi fluktuasi pada hasil testing mulai pada epoch ke 25 sampai ke 50.

Parameter yang diukur memiliki karakteristik prediksi sebagai berikut :

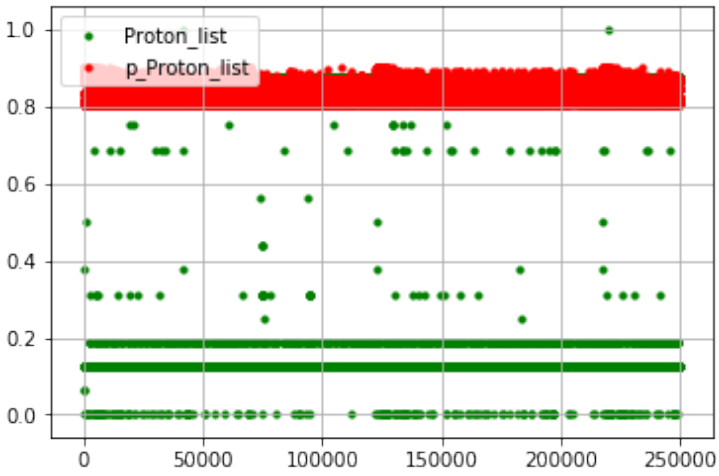
1. Proto



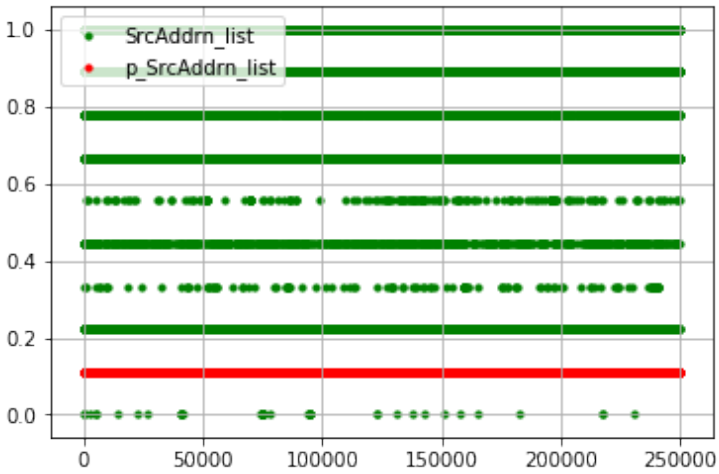
Gambar 4.36: Loss model LSTM Rbot Multivariate Prediction



Gambar 4.37: Akurasi model LSTM Rbot Multivariate Prediction



Gambar 4.38: Grafik Prediksi Proto rbot



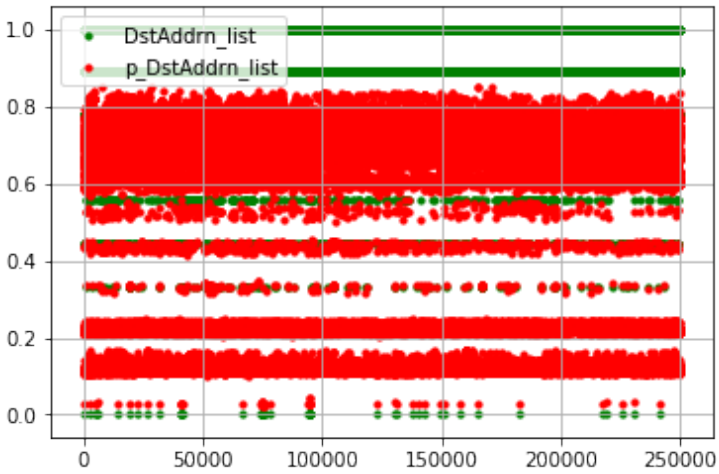
Gambar 4.39: Grafik Prediksi SrcAddr rbot

Bagian ini merupakan salah satu bagian dari paket jaringan yang mendefinisikan jenis protokol jaringan yang digunakan. Dari grafik diatas dapat dilihat cakupan dari prediksi ada disekitar 0.8 sampai 0.9 sedangkan diluar dari itu tidak terdeteksi.

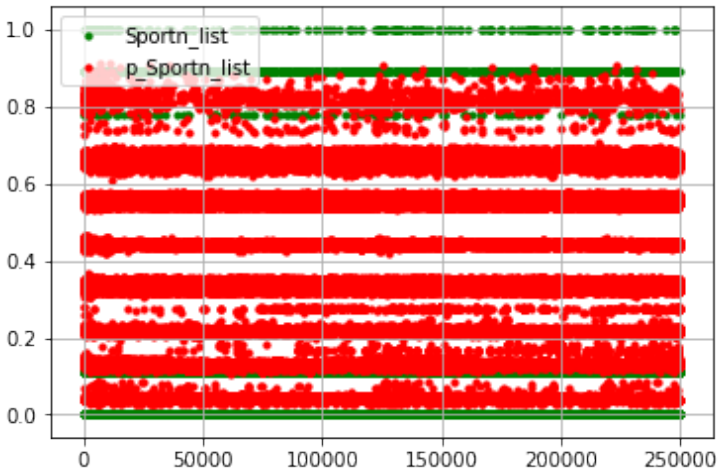
2. SrcAddr

   Bagian ini merupakan bagian IP header dari paket jaringan yang mendefinisikan alamat IP sumber. Dapat dilihat dari grafik diatas hanya satu bagian yang terdeteksi yakni 0.1, sedangkan sumber lain tidak terdeteksi.

3. DstAddr



Gambar 4.40: Grafik Prediksi DstAddr rbot



Gambar 4.41: Grafik Prediksi Sport rbot

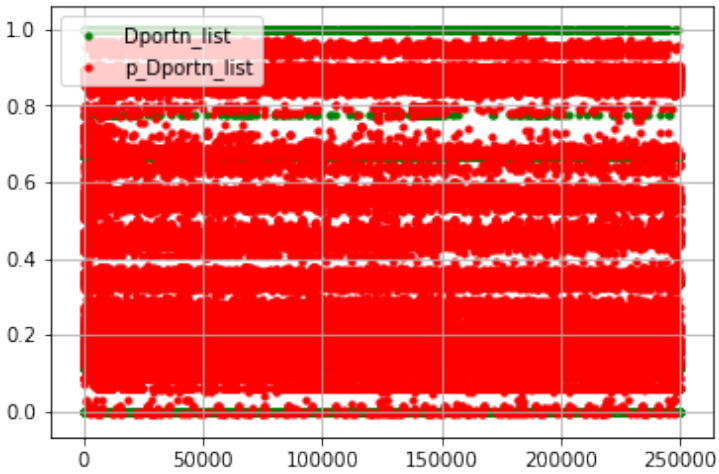
Bagian ini merupakan bagian IP header dari paket jaringan yang mendefinsikan alamat IP tujuan. Dapat dilihat dari grafik diatas bahwa sebagian besar trafik terdeteksi kecuali dibagian antara 0.8 dan 1.0.

4. Sport

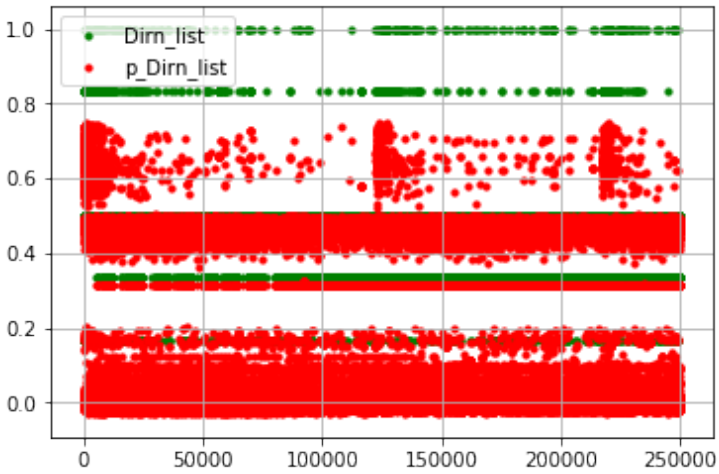
   Bagian ini merupakan bagian dari IP header yang mendefinisikan port sumber. Dapat dilihat dari grafik diatas bahwa sebagian besar trafik terdeteksi kecuali di antara 0.0 dan 1.0.

5. Dport

   Bagian ini merupakan bagian dari IP header yang mendefinisikan port tu-



Gambar 4.42: Grafik Prediksi Dport rbot



Gambar 4.43: Grafik Prediksi Dir rbot

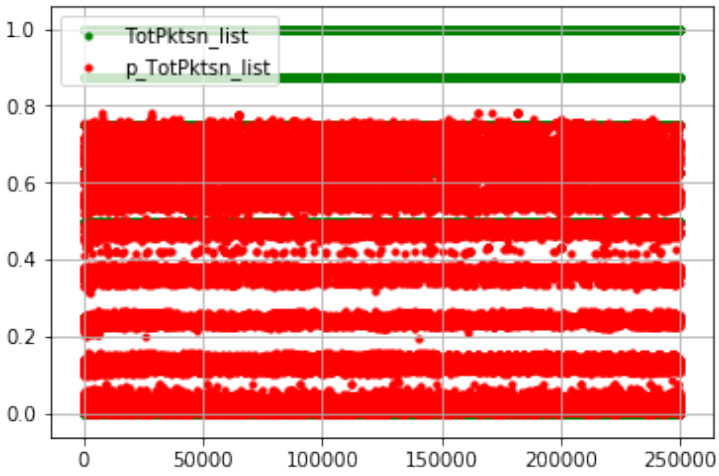
juan. Dapat dilihat dari grafik prediksi diatas hasil perdiksi dapat mencakup keseluruhan trafik kecuali bagian tertinggi 1.0.

6. Dir

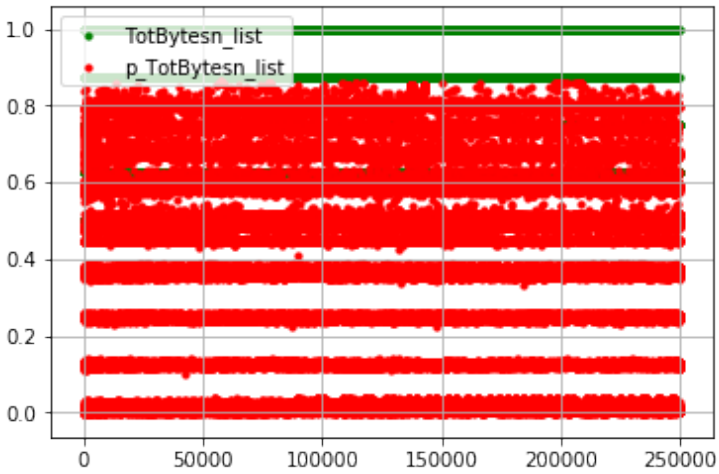
   Bagian ini merupakan arah yang menunjukan arah transmisi dari paket baik itu multicast, maupun unicast. Dapat dilihat dari grafik prediksi diatas keseluruhan hasil prediksi dapat mencakup trafik kecuali 0.8 dan 1.0.

7. TotPkts

   Bagian ini merupakan bagian dari paket header yang menjelaskan jumlah paket dalam suatu ethernet frame yang dikirim per paket. Dari grafik hasil



Gambar 4.44: Grafik Prediksi TotPkts rbot



Gambar 4.45: Grafik Prediksi TotBytes rbot

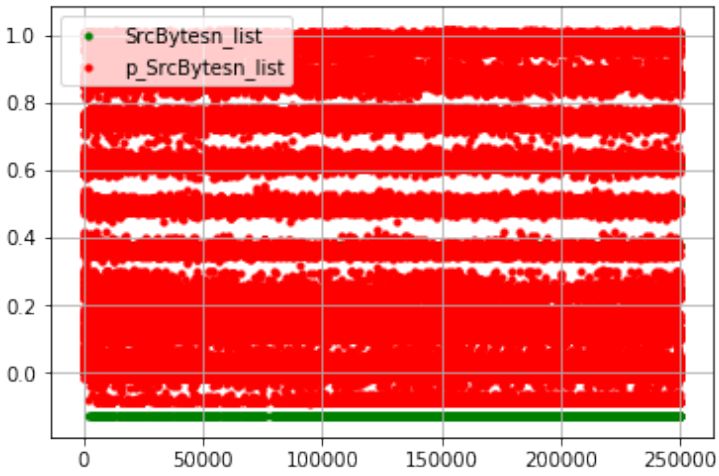
prediksi dapat dilihat seluruh trafik dapat diprediksi kecuali diantar 0.8 dan 1.0

8. TotBytes

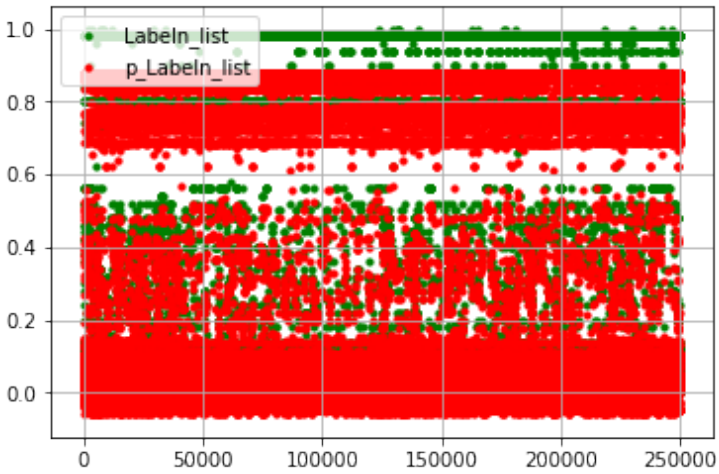
   Bagian ini merupakan bagian dari paket header yang menjelaskan jumlah bytes dalam satu ethernet frame yang dikirim per paket. Dapat dilihat dari grafik prediksi diatas bahwa secara keseluruhan dapat diprediksi kecuali nilai diantara 0.8 dan 1.0.

9. SrcBytes

   Bagian ini merupakan bagian dari paket header yang menjelaskan ukuran byte



Gambar 4.46: Grafik Prediksi SrcBytes rbot



Gambar 4.47: Grafik Prediksi Label rbot

source. Dapat dilihat dari grafik diatas data SrcBytes dapat diprediksi kecuali nilai terendah diantara 0.0.

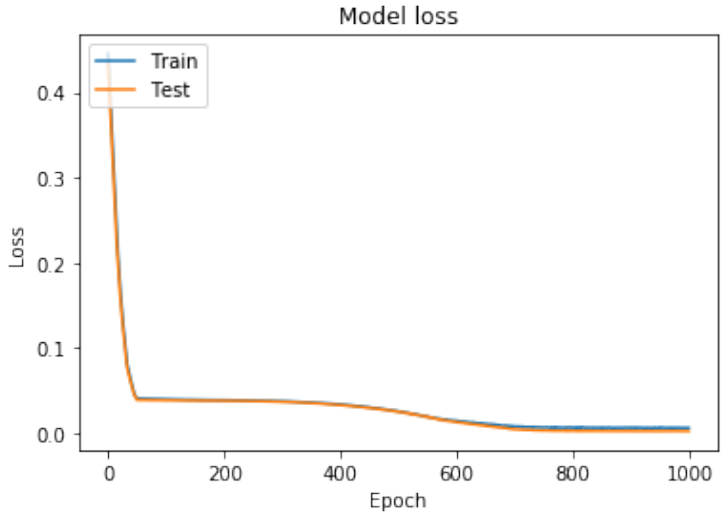
10. Label

    Bagian ini merupakan label dari dataset yang sudah di angka kan. Dapat dilihat dari grafik hasil prediksi diatas, bahwa secara keseluruhan trafik dapat diprediksi kecuali trafik diantara 1.0 dan 0.6.

Dapat diamati grafik keseluruhan dari hasil prediksi kesepuluh parameter *LSTM Multivariate Prediction*, dapat dilihat bahwa pada beberapa parameter prediksi terdapat beberapa parmaeter yang berhasil di prediksi dan beberapa parameter pula

Tabel 4.6: Tabel Hasil LSTMM Rbot

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.097	0.717	0.097	0.717	0.124	0.470	0.124	0.470
5	0.039	0.815	0.039	0.815	0.046	0.731	0.046	0.731
10	0.037	0.814	0.037	0.814	0.043	0.734	0.043	0.734
15	0.036	0.778	0.036	0.778	0.042	0.737	0.042	0.737
20	0.036	0.804	0.036	0.804	0.040	0.744	0.040	0.744
25	0.035	0.702	0.035	0.702	0.040	0.746	0.040	0.746
30	0.034	0.766	0.034	0.766	0.039	0.745	0.039	0.745
35	0.034	0.698	0.034	0.698	0.038	0.748	0.038	0.748
40	0.034	0.736	0.034	0.736	0.038	0.748	0.038	0.748
45	0.033	0.700	0.033	0.700	0.038	0.747	0.038	0.747



Gambar 4.48: Loss model LSTM Svchosta 4 Directional Header

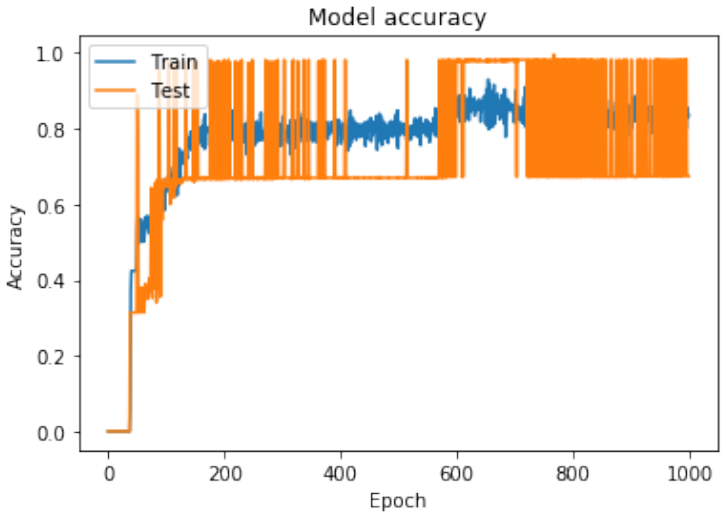
yang tidak dapat terdeteksi.

Tabel 4.6 diatas menunjukkan bahwa penurunan Loss terjadi dan peningkatan terjadi. Pada sisi akurasi dapat dilihat bahwa akurasi tertinggi yang dicapai model adalah 0.75. Hal ini dikarenakan beberapa parameter tidak mempengaruhi satu sama lain, namun berdasarkan keseluruhan hasil prediksi yang diperoleh dapat digunakan untuk melakukan profiling.

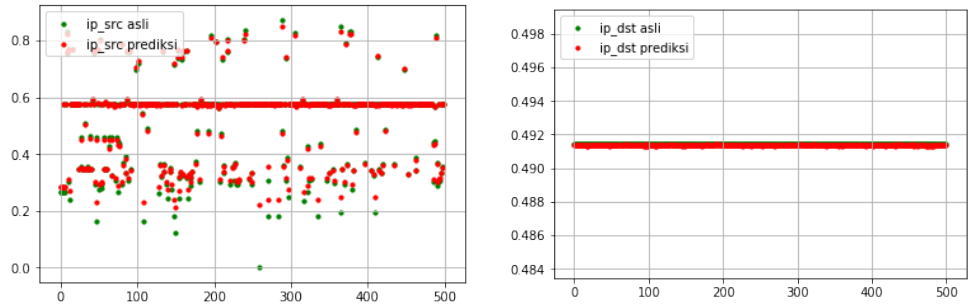
4.2.3.7 Hasil Training LSTM Svchosta 4 Directional Header

Grafik training diatas menunjukkan bahwa pada sisi loss terjadi konvergensi baik pada sisi *Loss* maupun pada sisi *Akurasi*. Semakin banyak *epoch* yang terjadi maka semakin kecil *Loss* yang diperoleh. Semakin banyak *epoch* yang terjadi maka semakin tinggi *Accuracy* yang diperoleh.

Grafik hasil prediksi 4.50 diperoleh dengan mengambil 500 sampel dari dataset,



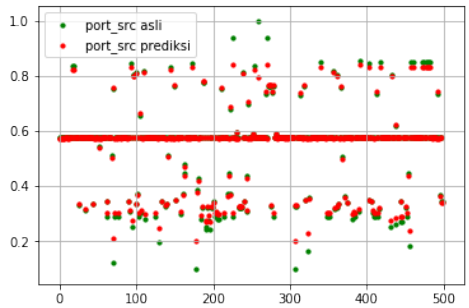
Gambar 4.49: Akurasi model LSTM Svchosta 4 Directional Header



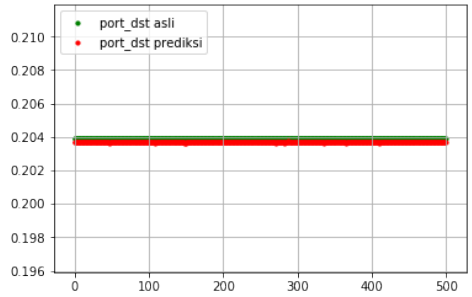
(a) Prediksi IPSrc model LSTM4 svchosta

(b) Prediksi IPDst model LSTM4 svchosta

Gambar 4.50: Prediksi IP Address model LSTM4 svchosta



(a) Prediksi PortSrc model LSTM4 svchosta



(b) Prediksi PortDst model LSTM4 svchosta

Gambar 4.51: Prediksi Port model LSTM4 svchosta

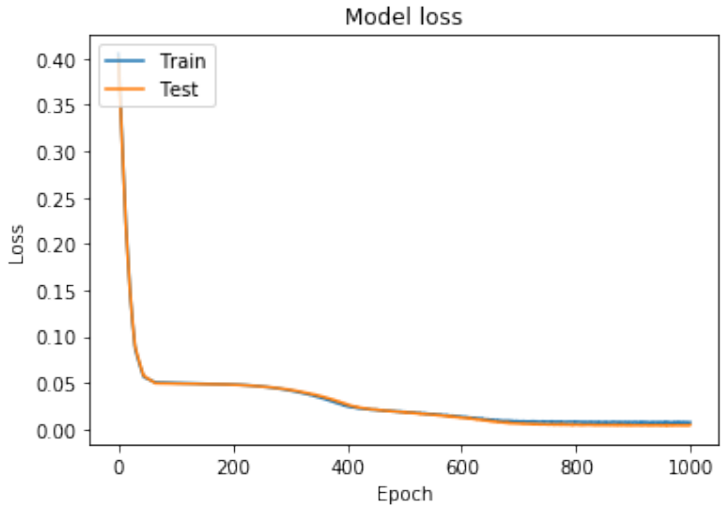
Tabel 4.7: Tabel Hasil LSTM4 Svchosta

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.433	0.000	0.433	0.000	0.446	0.000	0.446	0.000
100	0.038	0.646	0.038	0.646	0.040	0.650	0.040	0.650
200	0.038	0.982	0.038	0.982	0.039	0.814	0.039	0.814
300	0.037	0.670	0.037	0.670	0.037	0.772	0.037	0.772
400	0.033	0.670	0.033	0.670	0.034	0.778	0.034	0.778
500	0.025	0.672	0.025	0.672	0.025	0.810	0.025	0.810
600	0.013	0.982	0.013	0.982	0.014	0.856	0.014	0.856
700	0.004	0.982	0.004	0.982	0.008	0.846	0.008	0.846
800	0.002	0.672	0.002	0.672	0.007	0.868	0.007	0.868
900	0.002	0.982	0.002	0.982	0.006	0.848	0.006	0.848

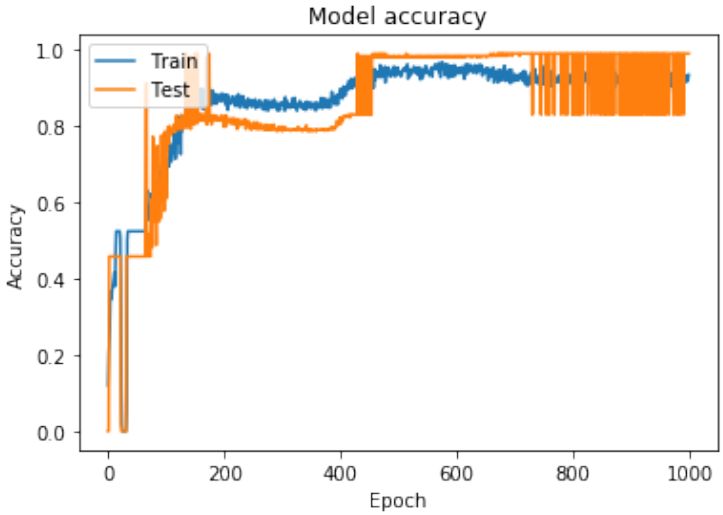
dapat diamati bahwa hampir secara keseluruhan hasil prediksi cocok dengan data yang di inputkan. Grafik prediksi **IPSrc** memiliki divergensi yang tinggi berarti banyak IP yang mengakses Server. Sedangkan Grafik IPDst yang lurus menandakan hanya ada satu alamat IP yang dituju yakni alamat IP Server.

Pada grafik hasil prediksi 4.51 dapat dilihat bahwa secara keseluruhan hasil prediksi cocok dengan data yang di inputkan. Grafik prediksi **PortSrc** memiliki divergensi yang tinggi berarti banyak Port sumber yang mengakses server. Sedangkan Grafik PortDst yang lurus menandakan hanya ada satu Port yang dituju yakni Port HTTP.

Berdasarkan tabel 4.7 diatas dapat diamati bahwa seiring dengan meningkatnya jumlah epoch Nilai *Loss* baik *Value Loss* maupun *Loss* menurun. Begitu pula dengan meningkatnya jumlah epoch Nilai *Accuracy* baik *Value Accuracy* maupun *Accuracy*. Dapat diamati pada tabel diatas akurasi tertinggi yang dicapai adalah 98.2% pada sisi testing dan 84.8%.



Gambar 4.52: Loss model LSTM Neris 4 Directional Header

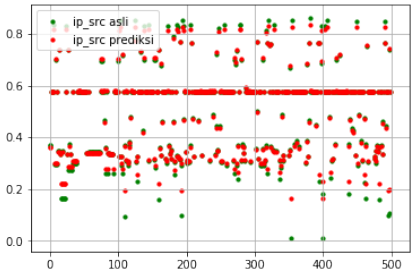


Gambar 4.53: Akuasi model LSTM Neris 4 Directional Header

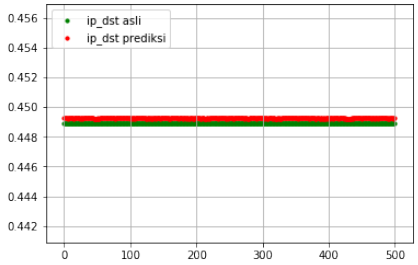
4.2.3.8 Hasil Training LSTM Neris 4 Directional Header

Grafik hasil training 4.53 diatas menunjukkan bahwa pada sisi loss terjadi konvergensi baik pada sisi *Loss* maupun pada sisi *Akurasi*. Semakin banyak *epoch* yang terjadi maka semakin kecil *Loss* yang diperoleh. Semakin banyak *epoch* yang terjadi maka semakin tinggi *Accuracy* yang diperoleh.

Pada grafik hasil prediksi 4.54 dapat diamati bahwa hampir secara keseluruhan hasil prediksi cocok dengan data yang di inputkan. Grafik prediksi **IPSrc** memiliki divergensi yang tinggi berarti banyak IP yang mengakses Server. Sedangkan Grafik IPDst yang lurus menandakan hanya ada satu alamat IP yang dituju yakni alamat IP

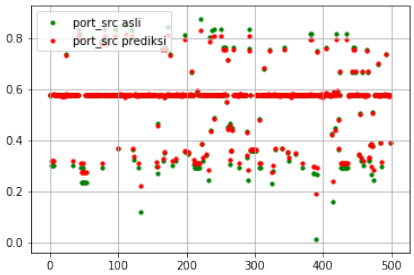


(a) Prediksi IPSrc model LSTM4 Neris

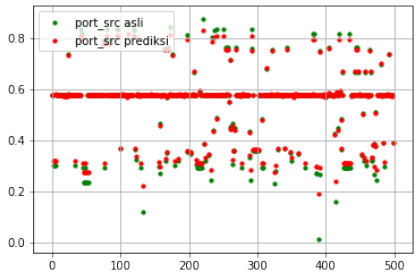


(b) Prediksi IPDst model LSTM4 Neris

Gambar 4.54: Prediksi IP Address model LSTM4 Neris



(a) Prediksi PortSrc model LSTM4 Neris



(b) Prediksi PortSrc model LSTM4 Neris

Gambar 4.55: Prediksi Port model LSTM4 Neris

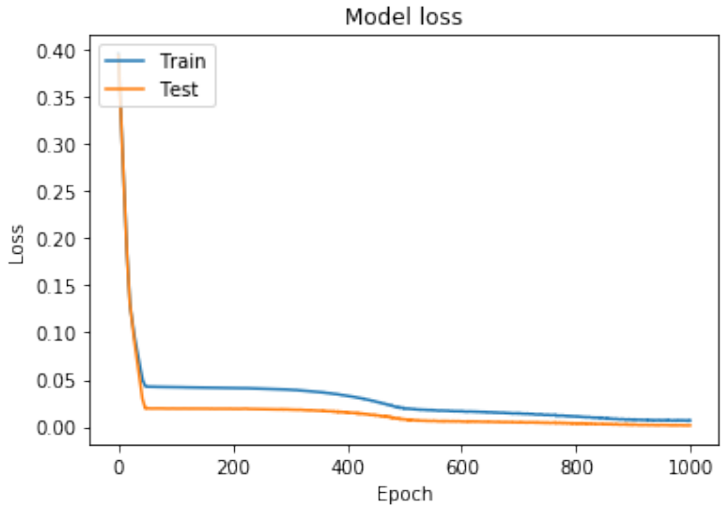
Server.

Berdasarkan grafik hasil prediksi 4.55 dengan mengambil 500 sampel dari dataset dapat diamati bahwa hampir secara keseluruhan hasil prediksi cocok dengan data yang di inputkan. Grafik prediksi **PortSrc** memiliki divergensi yang tinggi berarti banyak Port sumber yang mengakses server. Sedangkan Grafik PortDst yang lurus menandakan hanya ada satu Port yang dituju yakni Port HTTP.

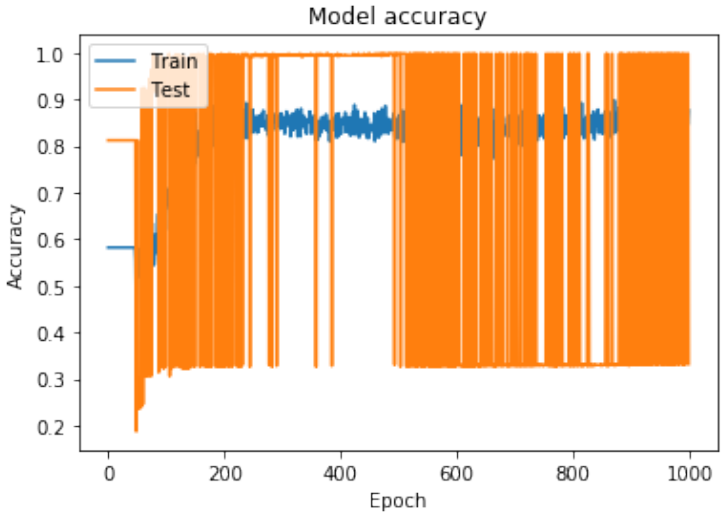
Berdasarkan tabel 4.8 diatas dapat diamati bahwa seiring dengan meningkatnya

Tabel 4.8: Tabel Hasil LSTM4 Neris

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.399	0.000	0.399	2.464	0.405	0.120	0.405	2.539
100	0.049	0.610	0.049	2.268	0.050	0.642	0.050	2.260
200	0.048	0.826	0.048	2.267	0.048	0.882	0.048	2.259
300	0.042	0.786	0.042	2.258	0.042	0.856	0.042	2.250
400	0.027	0.806	0.027	2.243	0.025	0.898	0.025	2.235
500	0.019	0.980	0.019	2.238	0.019	0.936	0.019	2.231
600	0.012	0.984	0.012	2.232	0.014	0.938	0.014	2.226
700	0.006	0.990	0.006	2.228	0.009	0.936	0.009	2.224
800	0.005	0.990	0.005	2.227	0.008	0.934	0.008	2.224
900	0.004	0.830	0.004	2.227	0.008	0.930	0.008	2.224



Gambar 4.56: Loss model LSTM Rbot 4 Directional Header

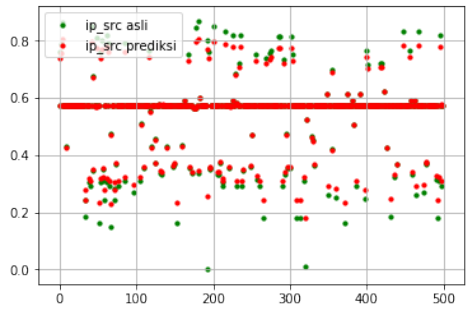


Gambar 4.57: Akurasi model LSTM Rbot 4 Directional Header

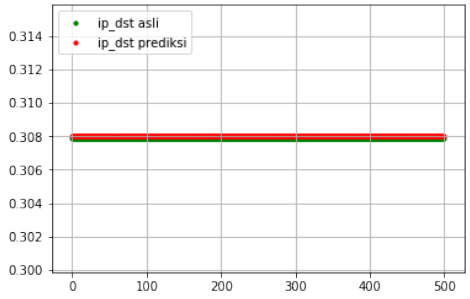
jumlah epoch Nilai *Loss* baik *Value Loss* maupun *Loss* menurun. Begitu pula dengan meningkatnya jumlah epoch Nilai *Accuracy* baik *Value Accuracy* maupun *Accuracy*. Dapat diamati pada tabel diatas akurasi tertinggi yang dicapai adalah 94.0% pada sisi testing dan 83.0%.

4.2.3.9 Hasil Training LSTM Rbot 4 Directional Header

Mengacu pada grafik hasil training diatas dapat dilihat bahwa pada sisi loss terjadi konvergensi baik pada sisi *Loss* maupun pada sisi *Akurasi*. Semakin banyak *epoch* yang terjadi maka semakin kecil *Loss* yang diperoleh. Semakin banyak *epoch*

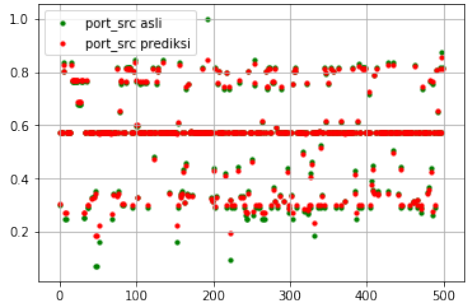


(a) Prediksi IPSrc model LSTM4 rbot

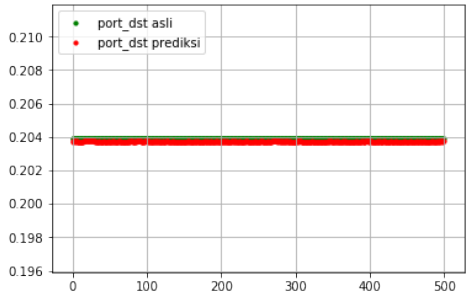


(b) Prediksi IPDst model LSTM4 rbot

Gambar 4.58: Prediksi IP Address model LSTM 4 rbot



(a) Prediksi PortSrc model LSTM4 Rbot



(b) Prediksi PortDst model LSTM4 Rbot

Gambar 4.59: Prediksi Port model LSTM4 Rbot

yang terjadi maka semakin tinggi *Accuracy* yang diperoleh.

Grafik hasil prediksi 4.58 menunjukkan bahwa hampir secara keseluruhan hasil prediksi cocok dengan data yang di inputkan. Grafik prediksi **IPSrc** memiliki divergensi yang tinggi berarti banyak IP yang mengakses Server. Sedangkan Grafik IPDst yang lurus menandakan hanya ada satu alamat IP yang dituju yakni alamat IP Server.

Grafik 4.59 diperoleh dengan mengambil 500 sampel dari dataset dapat diamati bahwa hampir secara keseluruhan hasil prediksi cocok dengan data yang di inputkan. Grafik prediksi **PortSrc** memiliki divergensi yang tinggi berarti banyak Port sumber yang mengakses server. Sedangkan Grafik PortDst yang lurus menandakan hanya ada satu Port yang dituju yakni Port HTTP.

Pada tabel 4.9 diatas dapat diamati bahwa seiring dengan meningkatnya jumlah epoch Nilai *Loss* baik *Value Loss* maupun *Loss* menurun. Begitu pula dengan meningkatnya jumlah epoch Nilai *Accuracy* baik *Value Accuracy* maupun *Accuracy*.

Tabel 4.9: Tabel Hasil LSTM4 Rbot

e	VL	VA	VMAE	VCA	L	A	MAE	CA
0	0.393	0.812	0.393	2.265	0.395	0.582	0.395	4.386
100	0.019	0.980	0.019	2.139	0.042	0.684	0.042	2.115
200	0.019	0.996	0.019	2.139	0.041	0.838	0.041	2.115
300	0.018	0.994	0.018	2.138	0.039	0.856	0.039	2.112
400	0.015	0.994	0.015	2.134	0.033	0.832	0.033	2.105
500	0.008	0.998	0.008	2.129	0.019	0.816	0.019	2.096
600	0.006	0.330	0.006	2.128	0.017	0.806	0.017	2.095
700	0.005	0.328	0.005	2.127	0.014	0.810	0.014	2.092
800	0.004	0.328	0.004	2.126	0.011	0.812	0.011	2.089
900	0.002	0.330	0.002	2.125	0.007	0.856	0.007	2.087

Metode	Karakteristik
Sentiment Analysis	Tidak konvergen, sehingga tidak perlu dilanjutkan
Multivariate Analysis	Konvergen, tetapi banyak parameter yang mengganggu proses training
4 Directional Header	Konvergen dan karena hanya menggunakan bagian yang benar-benar diperhitungkan sehingga hasil yang diperoleh akurat

Tabel 4.10: Data hasil perbandingan metode LSTM

Dapat diamati pada tabel diatas akurasi tertinggi yang dicapai adalah 94.0% pada sisi testing dan 83.0%.

4.2.4 Perbandingan Metode LSTM

Berdasarkan dari ketiga metode LSTM yang digunakan, dapat kita amati beberapa karakteristik perbedaan sebagai berikut :

4.2.5 Hasil Training Data CNN

Pada jaringan CNN terjadi proses filter packet payload. Disini ditentukan apakah payload mengandung virus atau tidak. Proses training dilakukan dengan mengubah-ubah parameter learning_rate. Hal ini dilakukan untuk mengamati perubahan konvergensi proses training dan testing dari CNN.

4.2.5.1 Hasil Training CNN Svchosta

Dapat dilihat dari tabel hasil train 4.11 diatas, semakin banyak epoch maka semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate* .

Dapat diamati berdasarkan parameter *learning rate*, dimana semakin besar nilai

Tabel 4.11: Tabel Hasil Training CNN Svchosta

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.800	0.798	0.794	0.206	0.515	0.537	0.604	0.893
5	1.000	0.998	0.889	0.229	0.007	0.057	0.453	0.816
10	1.000	1.000	0.950	0.308	0.003	0.027	0.337	0.750
15	1.000	1.000	0.973	0.438	0.002	0.018	0.255	0.691
20	1.000	1.000	0.980	0.587	0.001	0.013	0.198	0.645
25	1.000	1.000	0.984	0.741	0.001	0.011	0.160	0.596
30	1.000	1.000	0.991	0.873	0.001	0.009	0.131	0.556
35	1.000	1.000	0.993	0.941	0.001	0.008	0.110	0.518
40	1.000	1.000	0.995	0.973	0.001	0.007	0.096	0.485
45	1.000	1.000	0.995	0.986	0.001	0.006	0.083	0.455

Tabel 4.12: Tabel Hasil Testing CNN Svchosta

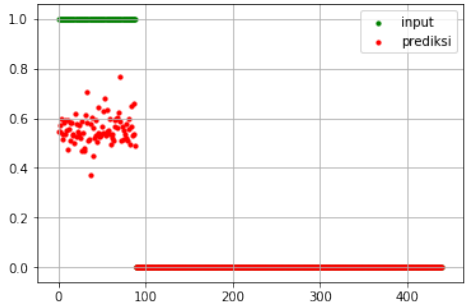
Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.980	0.953	0.861	0.141	0.234	0.267	0.551	0.917
5	1.000	1.000	0.920	0.169	0.008	0.050	0.417	0.832
10	1.000	1.000	0.966	0.277	0.003	0.024	0.306	0.762
15	1.000	1.000	0.981	0.407	0.002	0.016	0.228	0.701
20	1.000	1.000	0.987	0.601	0.001	0.013	0.176	0.649
25	1.000	1.000	0.991	0.782	0.001	0.009	0.140	0.603
30	1.000	1.000	0.995	0.900	0.001	0.008	0.115	0.561
35	1.000	1.000	0.997	0.959	0.001	0.007	0.096	0.523
40	1.000	1.000	0.997	0.981	0.001	0.006	0.083	0.488
45	1.000	1.000	0.997	0.991	0.001	0.005	0.072	0.456

learning rate maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

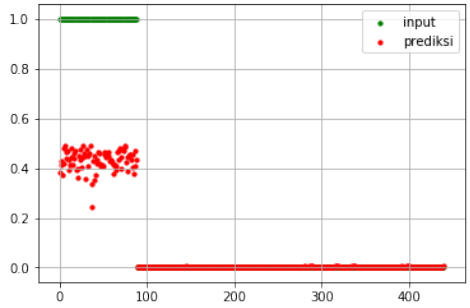
Dapat dilihat dari tabel hasil test 4.12 diatas, semakin banyak epoch maka semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

Dapat dilihat berdasarkan parameter *learning rate*, dimana semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

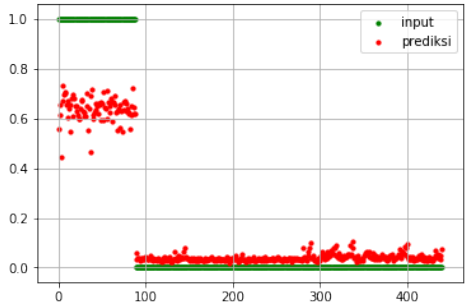
Berdasarkan grafik prediksi 4.60 diatas bahwa terdapat perbedaan hasil prediksi ketika parameter *learning rate* diubah. Semakin kecil nilai parameter learning rate menyebabkan persebaran prediksi menjadi semakkin tinggi.



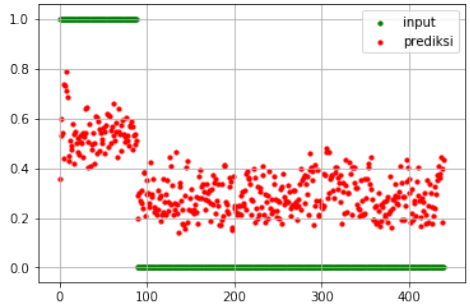
(a) Prediksi train lr=0.1



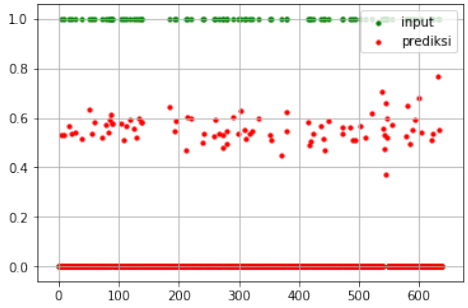
(b) Prediksi train lr=0.01



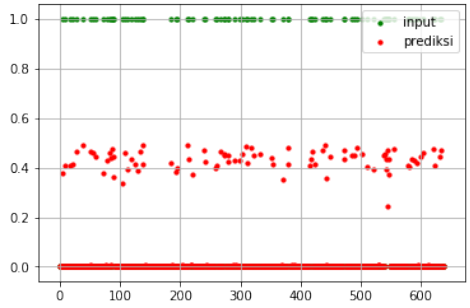
(c) Prediksi train lr=0.001



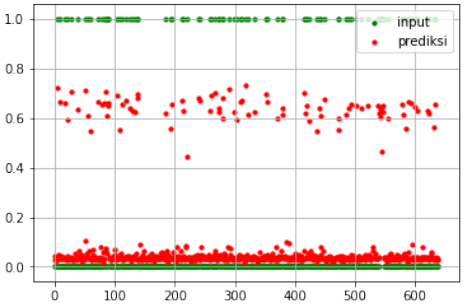
(d) Prediksi train lr=0.0001



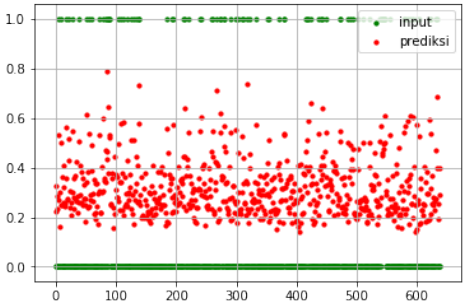
(e) Prediksi train lr=0.1



(f) Prediksi test lr=0.01

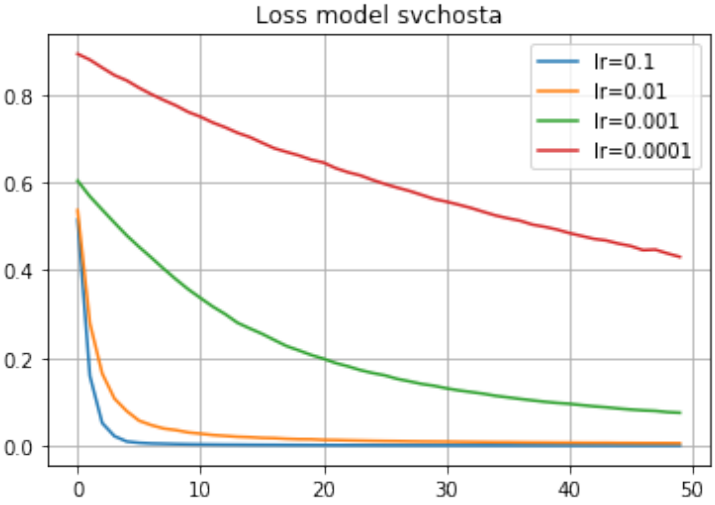


(g) Prediksi test lr=0.001

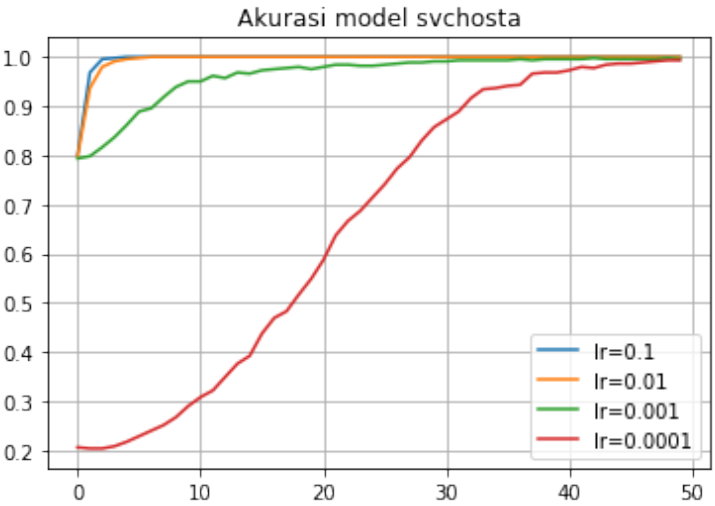


(h) Prediksi test lr=0.0001

Gambar 4.60: Grafik Prediksi Svchosta CNN



Gambar 4.61: Grafik Loss Svchosta CNN



Gambar 4.62: Grafik Akurasi Svchosta CNN

Tabel 4.13: Tabel Hasil Training CNN Rbot

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.771	0.488	0.723	0.798	0.513	0.701	0.666	0.590
5	1.000	0.998	0.864	0.803	0.006	0.056	0.580	0.561
10	1.000	1.000	0.927	0.819	0.003	0.026	0.484	0.533
15	1.000	1.000	0.950	0.830	0.002	0.017	0.388	0.508
20	1.000	1.000	0.971	0.839	0.001	0.013	0.302	0.484
25	1.000	1.000	0.980	0.862	0.001	0.011	0.237	0.464
30	1.000	1.000	0.982	0.882	0.001	0.009	0.191	0.442
35	1.000	1.000	0.984	0.905	0.001	0.008	0.157	0.423
40	1.000	1.000	0.986	0.916	0.001	0.007	0.132	0.406
45	1.000	1.000	0.989	0.923	0.001	0.006	0.112	0.387

Dapat dilihat dari grafik diatas dimana parameter yang memiliki parameter *learning rate* terbesar memiliki waktu konvergen yang lebih singkat dari *learning rate* lainnya.

Waktu konvergen dapat dilihat saat kurva menuju ke satu nilai tertentu yakni nilai 1, dari grafik diatas juga dapat dilihat bahwa kurva dengan learning rate tertinggi yakni kurva berwarna biru memiliki waktu konvergen yang paling singkat dari yang lainnya.

4.2.5.2 Hasil Training CNN Rbot

Berikut ini adalah tabel hasil training CNN untuk Botnet RBot

Dari tabel hasil train 4.13 diatas dapat diamati bahwa semakin banyak epoch maka semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

Berdasarkan parameter *learning rate*, dimana semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

Tabel hasil test 4.14 diatas menunjukkan bahwa semakin banyak epoch maka semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

Dapat dilihat berdasarkan parameter *learning rate* memiliki karakteristik dimana semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik

Tabel 4.14: Tabel Hasil Testing CNN Rbot

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
00	0.884	0.984	0.831	0.861	0.251	0.284	0.639	0.568
5	1.000	1.000	0.900	0.865	0.008	0.047	0.549	0.538
10	1.000	1.000	0.941	0.873	0.003	0.024	0.460	0.512
15	1.000	1.000	0.973	0.881	0.002	0.015	0.362	0.489
20	1.000	1.000	0.981	0.890	0.001	0.011	0.277	0.466
25	1.000	1.000	0.986	0.909	0.001	0.009	0.215	0.446
30	1.000	1.000	0.986	0.919	0.001	0.007	0.169	0.426
35	1.000	1.000	0.989	0.933	0.001	0.006	0.139	0.407
40	1.000	1.000	0.992	0.942	0.001	0.006	0.116	0.389
45	1.000	1.000	0.992	0.947	0.001	0.005	0.097	0.372

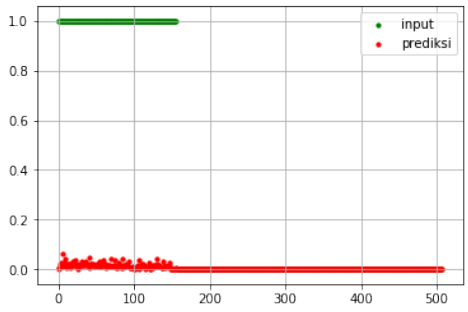
Tabel 4.15: Tabel Hasil Training CNN Neris

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.885	0.848	0.749	0.460	0.335	0.435	0.570	0.669
5	0.926	0.968	0.926	0.875	0.232	0.135	0.328	0.558
10	0.933	0.982	0.927	0.926	0.223	0.076	0.260	0.467
15	0.937	0.983	0.935	0.923	0.211	0.061	0.225	0.445
20	0.935	0.989	0.930	0.927	0.197	0.059	0.196	0.421
25	0.939	0.987	0.934	0.923	0.199	0.058	0.179	0.408
30	0.943	0.992	0.935	0.925	0.166	0.049	0.162	0.390
35	0.933	0.992	0.943	0.929	0.212	0.046	0.146	0.377
40	0.945	0.992	0.952	0.927	0.191	0.046	0.135	0.364
45	0.934	0.992	0.958	0.926	0.188	0.045	0.128	0.353

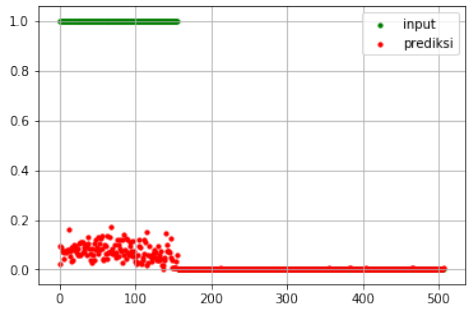
tertinggi, dan loss mencapai titik terendah.

Dapat dilihat dari grafik prediksi 4.63 diatas bahwa terdapat perbedaan hasil prediksi ketika parameter *learning rate* diubah. Semakin kecil nilai parameter learning rate menyebabkan persebaran prediksi menjadi semakkin tinggi.

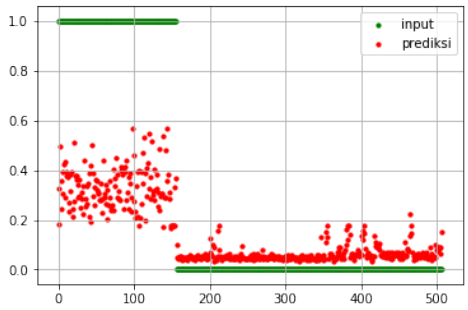
Grafik akurasi, dan loss menunjukkan bahwa parameter *learning rate* terbesar memiliki waktu konvergen yang lebih singkat dari *learning rate* lainnya. Dapat diamati juga sama seperti pola data hasil training sebelumnya bahwa hasil training paling konvergen diperoleh pada learning rate terendah yakni 0.1.



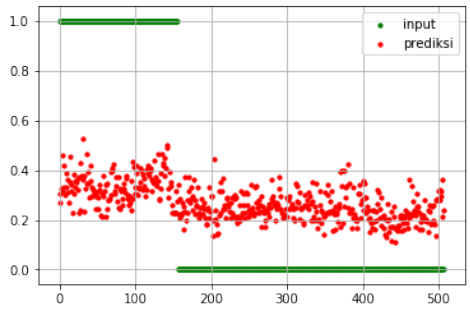
(a) Prediksi train lr=0.1



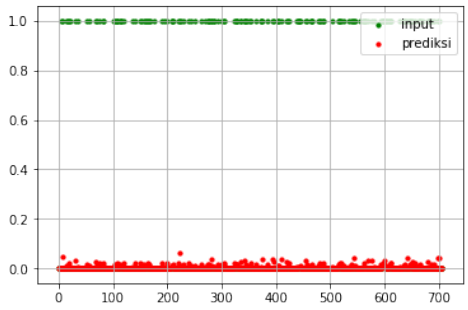
(b) Prediksi train lr=0.01



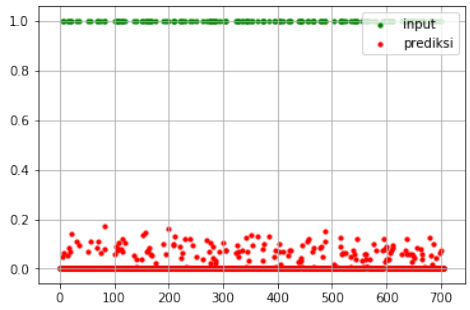
(c) Prediksi train lr=0.001



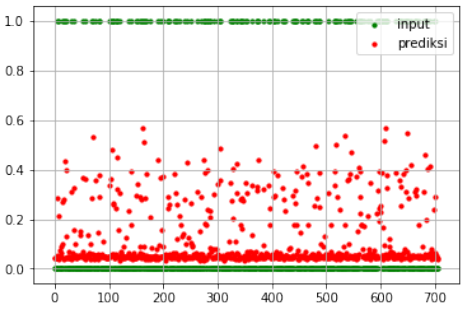
(d) Prediksi train lr=0.0001



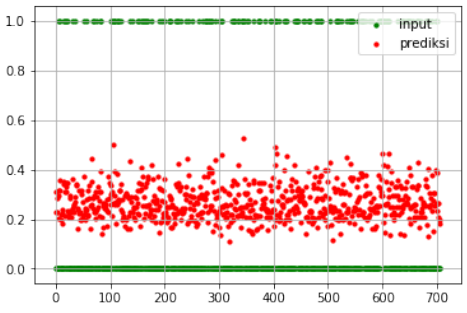
(e) Prediksi train lr=0.1



(f) Prediksi test lr=0.01

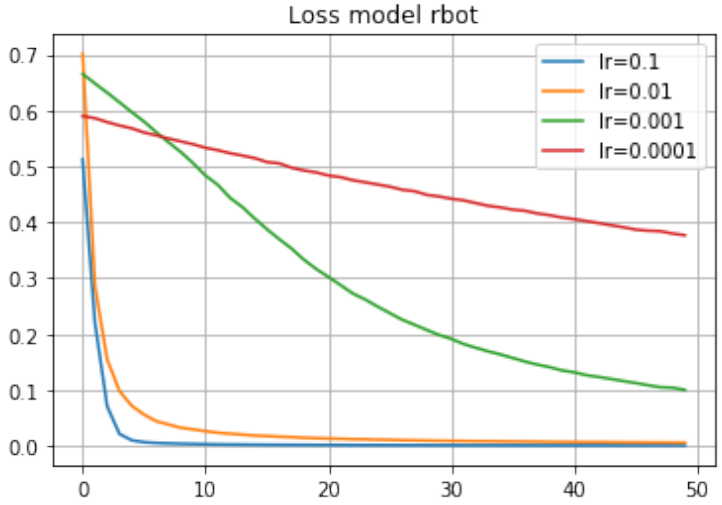


(g) Prediksi test lr=0.001

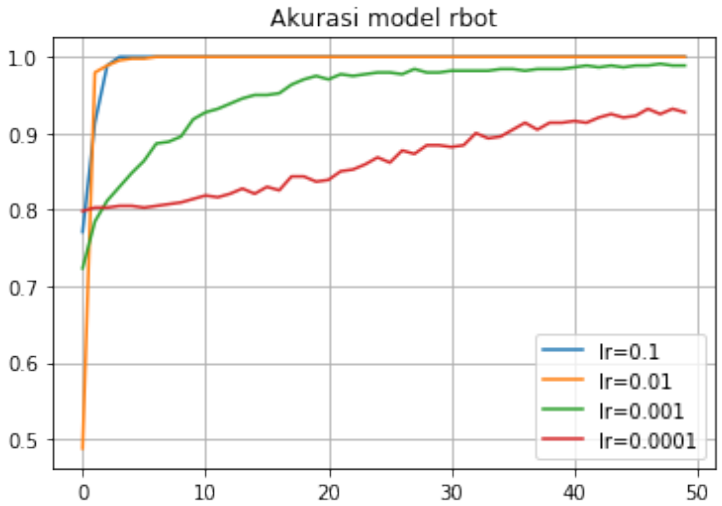


(h) Prediksi test lr=0.0001

Gambar 4.63: Grafik Prediksi Rbot CNN



Gambar 4.64: Grafik Loss Rbot CNN



Gambar 4.65: Grafik Akurasi Rbot CNN

Tabel 4.16: Tabel Hasil Testing CNN Neris

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.937	0.915	0.933	0.570	0.298	0.294	0.418	0.610
5	0.576	0.577	0.940	0.939	0.855	0.344	0.282	0.480
10	0.939	0.983	0.939	0.939	0.211	0.074	0.241	0.421
15	0.942	0.990	0.940	0.938	0.209	0.284	0.206	0.393
20	0.949	0.985	0.946	0.940	0.288	0.051	0.182	0.374
25	0.941	0.993	0.946	0.940	0.188	0.057	0.162	0.359
30	0.931	0.993	0.947	0.941	0.203	0.042	0.148	0.345
35	0.933	0.993	0.947	0.941	0.208	0.041	0.138	0.333
40	0.576	0.993	0.953	0.941	1.248	0.039	0.122	0.321
45	0.950	0.993	0.969	0.941	0.209	0.040	0.123	0.310

4.2.5.3 Hasil Training CNN Neris

Dapat dilihat dari tabel 4.15 hasil train diatas, semakin banyak epoch maka semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

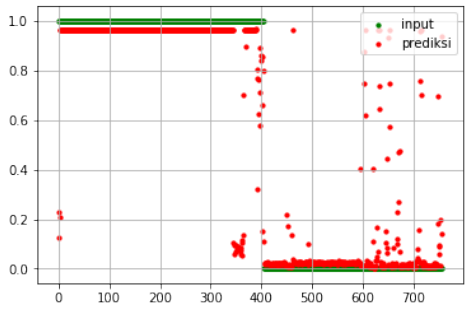
Berdasarkan parameter *learning rate*, dapat diamati bahwa semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

Dapat dilihat dari tabel hasil test 4.16 diatas, semakin banyak epoch maka semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

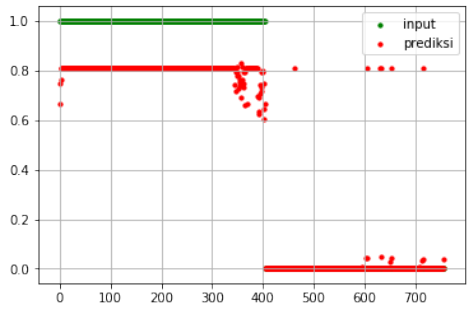
Dapat dilihat berdasarkan parameter *learning rate*, dimana semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

Sama seperti sebelumnya dari grafik prediksi 4.66 diatas bahwa terdapat perbedaan hasil prediksi ketika parameter *learning rate* diubah. Semakin kecil nilai parameter learning rate menyebabkan persebaran prediksi menjadi semakkin tinggi.

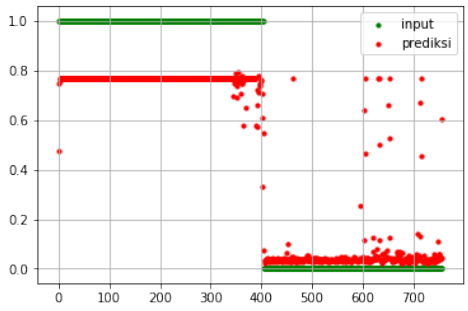
Grafik akurasi, dan loss diatas menunjukkan bahwa parameter yang memiliki *learning rate* terbesar memiliki waktu konvergen yang lebih singkat dari *learning rate* lainnya. Data yang diperoleh juga sama seperti data hasil training sebelumnya,



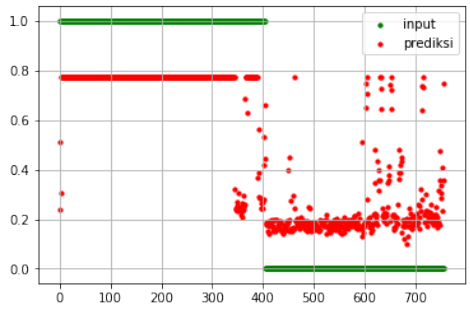
(a) Prediksi train lr=0.1



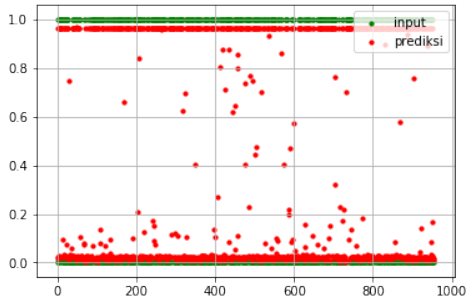
(b) Prediksi train lr=0.01



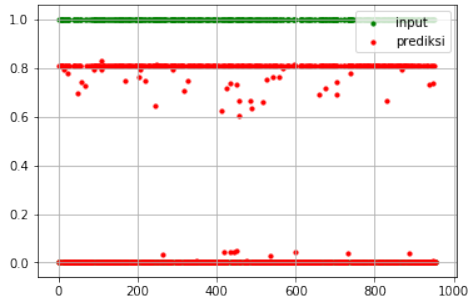
(c) Prediksi train lr=0.001



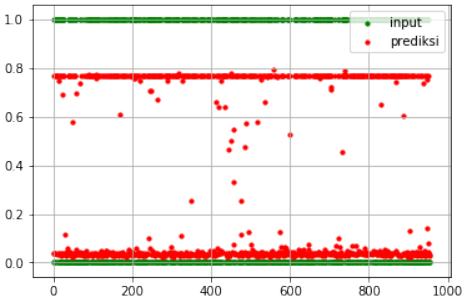
(d) Prediksi train lr=0.0001



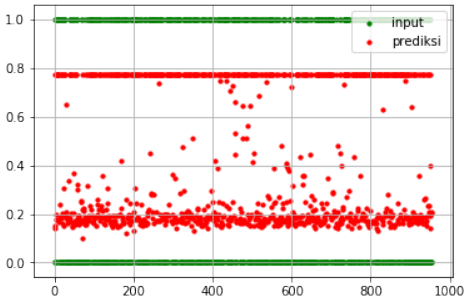
(e) Prediksi train lr=0.1



(f) Prediksi test lr=0.01

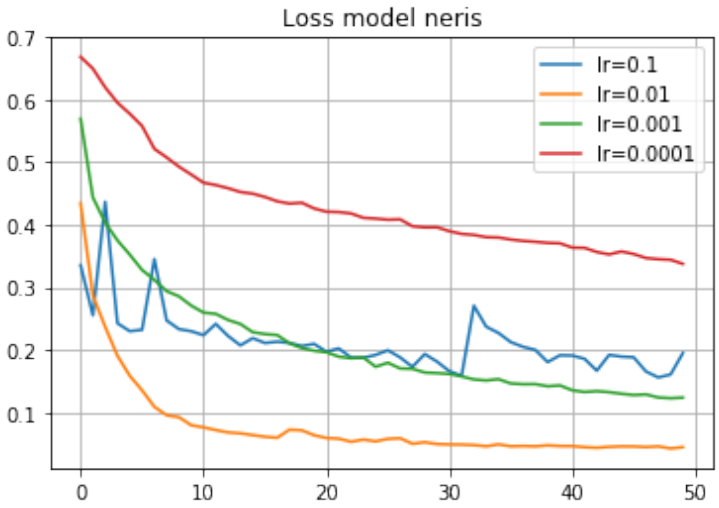


(g) Prediksi test lr=0.001

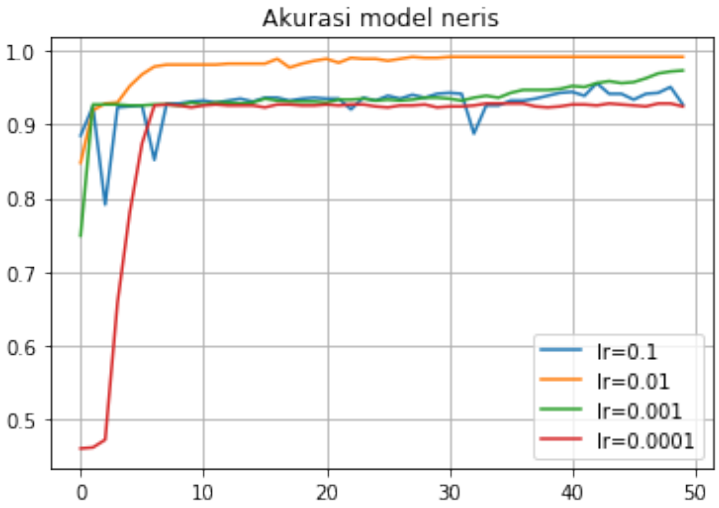


(h) Prediksi test lr=0.0001

Gambar 4.66: Grafik Prediksi Neris CNN



Gambar 4.67: Grafik Loss Neris CNN



Gambar 4.68: Grafik Akurasi Neris CNN

Tabel 4.17: Tabel Hasil Training CNN Single

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.835	0.472	0.659	0.102	0.397	0.681	0.664	1.022
5	0.948	0.942	0.879	0.152	0.098	0.170	0.386	0.866
10	0.942	0.950	0.948	0.365	0.082	0.105	0.279	0.756
15	0.948	0.945	0.950	0.619	0.074	0.094	0.222	0.668
20	0.945	0.950	0.961	0.772	0.074	0.086	0.186	0.597
25	0.940	0.950	0.955	0.843	0.069	0.082	0.162	0.540
30	0.945	0.940	0.955	0.877	0.068	0.080	0.147	0.495
35	0.948	0.950	0.955	0.906	0.067	0.079	0.134	0.454
40	0.942	0.950	0.948	0.927	0.066	0.077	0.126	0.421
45	0.945	0.945	0.945	0.937	0.066	0.075	0.120	0.392

Tabel 4.18: Tabel Hasil Testing CNN Single

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.950	0.820	0.843	0.067	0.147	0.466	0.593	1.031
5	0.971	0.967	0.919	0.102	0.063	0.106	0.344	0.878
10	0.971	0.971	0.967	0.356	0.054	0.074	0.234	0.755
15	0.972	0.971	0.971	0.686	0.048	0.063	0.177	0.657
20	0.972	0.971	0.971	0.831	0.046	0.061	0.147	0.580
25	0.972	0.971	0.971	0.884	0.045	0.055	0.129	0.515
30	0.972	0.971	0.971	0.914	0.045	0.053	0.114	0.465
35	0.972	0.971	0.971	0.936	0.044	0.052	0.104	0.422
40	0.972	0.972	0.971	0.950	0.043	0.052	0.097	0.388
45	0.972	0.972	0.971	0.960	0.043	0.050	0.091	0.359

dimana data yang paling konvergen merupakan data dengan *learning rate* terendah yakni 0.1.

4.2.5.4 Hasil Training CNN Single

Dapat diamati pada tabel hasil train 4.17 diatas, semakin banyak epoch maka semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

Berdasarkan parameter *learning rate*, dapat diamati bahwa semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

Dapat dilihat dari tabel hasil test 4.18 diatas, semakin banyak epoch maka

Tabel 4.19: Tabel Hasil Training CNN Multiple

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.698	0.589	0.310	0.304	0.595	0.654	0.805	1.053
5	0.771	0.779	0.729	0.304	0.326	0.337	0.604	0.950
10	0.792	0.753	0.792	0.336	0.268	0.293	0.503	0.866
15	0.792	0.775	0.794	0.496	0.261	0.278	0.442	0.797
20	0.783	0.771	0.806	0.601	0.256	0.272	0.395	0.745
25	0.798	0.775	0.787	0.638	0.254	0.267	0.364	0.704
30	0.808	0.785	0.791	0.670	0.250	0.262	0.345	0.672
35	0.812	0.743	0.785	0.684	0.244	0.260	0.330	0.644
40	0.785	0.791	0.785	0.700	0.243	0.256	0.320	0.623
45	0.794	0.787	0.802	0.717	0.242	0.254	0.312	0.598

semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

Dapat diamati berdasarkan parameter *learning rate*, memiliki karakteristik dimana semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

Dapat dilihat dari grafik prediksi 4.69 diatas bahwa terdapat perbedaan hasil prediksi ketika parameter *learning rate* diubah. Semakin kecil nilai parameter learning rate menyebabkan persebaran prediksi menjadi semakkin tinggi.

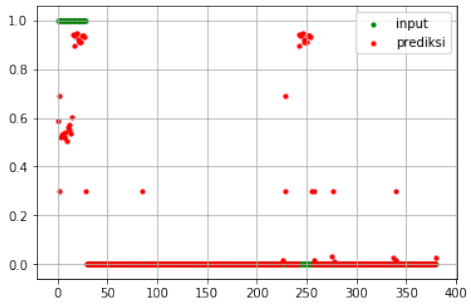
Dari grafik akurasi, dan loss dapat diamati bahwa parameter dengan *learning rate* terbesar memiliki waktu konvergen yang lebih singkat dari *learning rate* lainnya. Sama seperti data hasil training sebelumnya, data dengan konvergensi tertinggi ada pada *learning rate* 0.1.

4.2.5.5 Hasil Training CNN Multiple

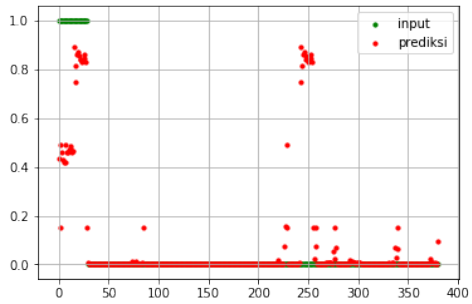
Berdasarkan dari tabel hasil train 4.19 diatas, semakin banyak epoch maka semakin tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

Dapat dilihat parameter *learning rate* menunjukkan bahwa semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

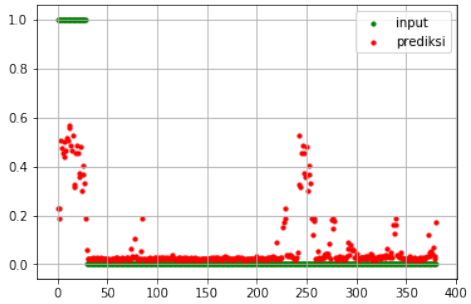
Tabel hasil test 4.20 diatas menunjukkansemakin banyak epoch maka semakin



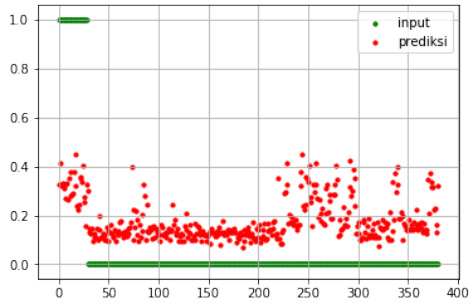
(a) Prediksi train lr=0.1



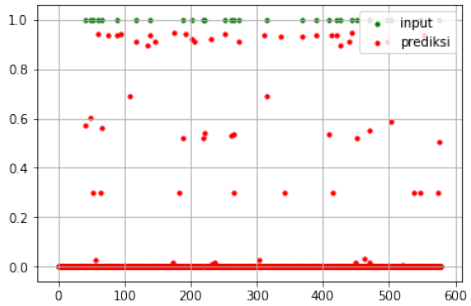
(b) Prediksi train lr=0.01



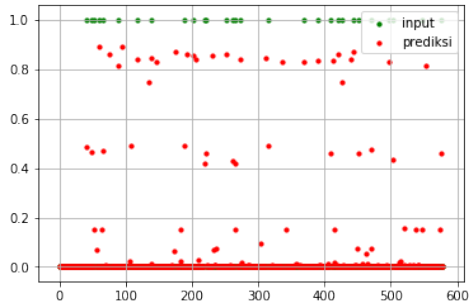
(c) Prediksi train lr=0.001



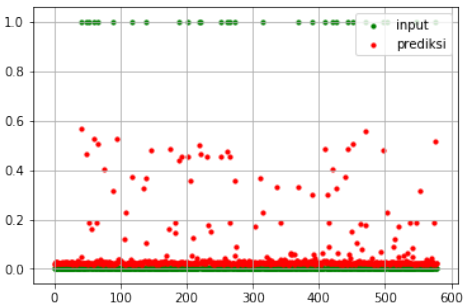
(d) Prediksi train lr=0.0001



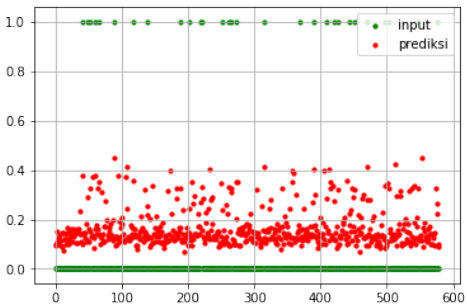
(e) Prediksi train lr=0.1



(f) Prediksi test lr=0.01

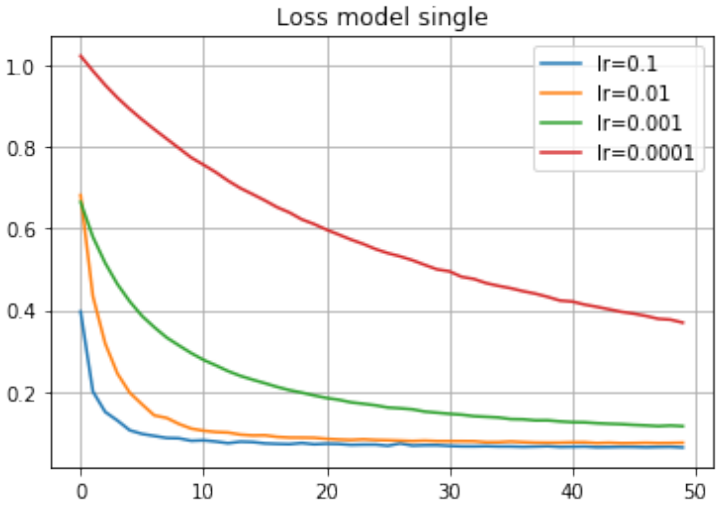


(g) Prediksi test lr=0.001

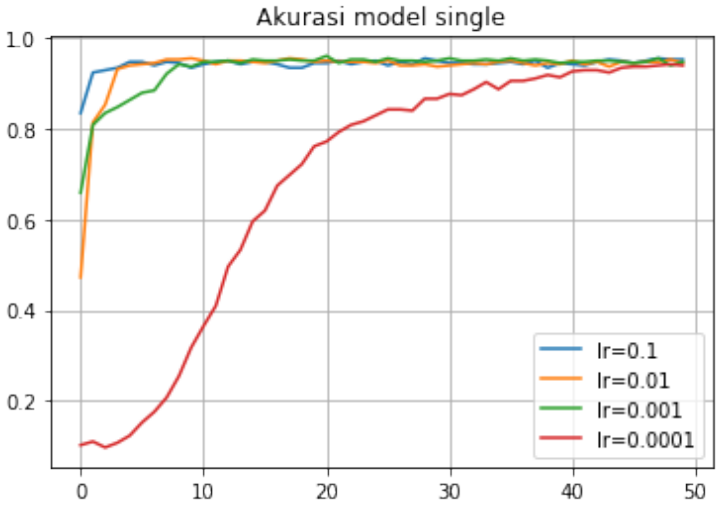


(h) Prediksi test lr=0.0001

Gambar 4.69: Grafik Prediksi Single CNN



Gambar 4.70: Grafik Loss Single CNN



Gambar 4.71: Grafik Akurasi Single CNN

Tabel 4.20: Tabel Hasil Testing CNN Multiple

Epoch	Akurasi				Loss			
	1e-1	1e-2	1e-3	1e-4	1e-1	1e-2	1e-3	1e-4
0	0.764	0.797	0.241	0.219	0.429	0.471	0.790	1.119
5	0.855	0.864	0.797	0.219	0.272	0.250	0.562	1.024
10	0.868	0.871	0.847	0.222	0.206	0.215	0.439	0.922
15	0.871	0.874	0.857	0.375	0.188	0.203	0.368	0.818
20	0.876	0.874	0.857	0.621	0.185	0.198	0.322	0.735
25	0.876	0.871	0.862	0.716	0.181	0.192	0.292	0.675
30	0.874	0.871	0.865	0.740	0.179	0.189	0.270	0.631
35	0.876	0.876	0.866	0.747	0.175	0.186	0.255	0.598
40	0.876	0.874	0.865	0.760	0.175	0.186	0.247	0.570
45	0.876	0.874	0.869	0.786	0.174	0.184	0.236	0.546

Ukuran Paket	Runtime			Detection Rate (p/s)
	Snort (s)	LSTM (s)	CNN (s)	
100	1.170	0.877	4.445	14.221
200	1.473	0.563	20.068	9.048
300	1.598	0.549	29.517	9.474
400	1.641	0.576	32.629	11.479
500	1.308	0.461	40.888	11.721
600	1.370	0.433	49.999	11.582
700	1.418	0.463	61.193	11.098
800	1.669	0.488	88.764	8.798
900	1.210	0.501	68.418	12.833
1000	1.857	0.526	82.615	11.764

Tabel 4.21: Data hasil detection rate preprocessor

tinggi akurasi dan semakin rendah loss nya. Lama akurasi mencapai titik tertinggi, dan loss mencapai titik terendah ditentukan oleh *learning rate*.

Berdasarkan parameter *learning rate* menunjukkan bahwa semakin besar nilai *learning rate* maka semakin cepat pula akurasi mencapai titik tertinggi, dan loss mencapai titik terendah.

Dapat dilihat dari grafik prediksi 4.72 diatas bahwa terdapat perbedaan hasil prediksi ketika parameter *learning rate* diubah. Semakin kecil nilai parameter learning rate menyebabkan persebaran prediksi menjadi semakkin tinggi.

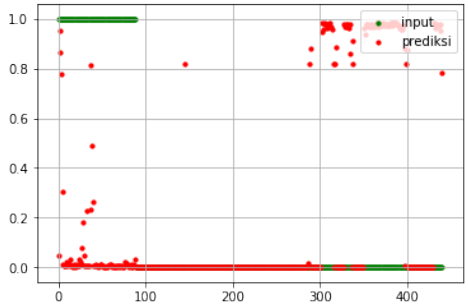
Dapat dilihat dari grafik akurasi, dan loss dimana data yang memiliki parameter *learning rate* terbesar memiliki waktu konvergen yang lebih singkat dari *learning rate* lainnya.

Semakin kecil *learning rate* nya, maka semakin lama waktu konvergensi nya. Dari grafik diatas diperoleh waktu konvergensi tercepat ada pada *learning rate* 0.1.

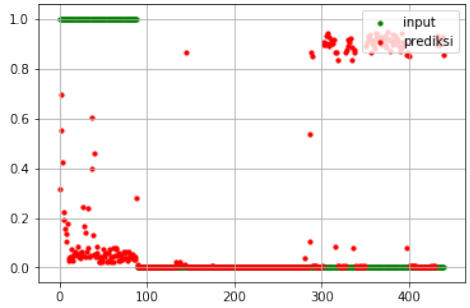
4.2.6 Data Hasil Detection Rate

Berikut ini adalah data hasil yang diperoleh dari kinerja snort dan keras dalam proses pendeteksian paket yang dipresentasikan dalam bentuk detection rate.

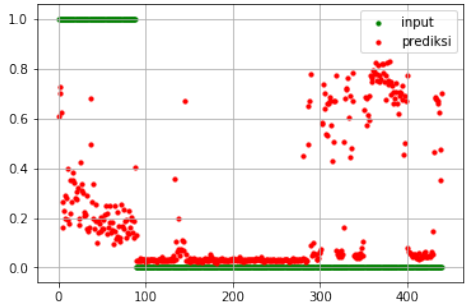
Berdasarkan tabel 4.21 dapat diamati bahwa *detection rate* pada sisi snort, dan LSTM memiliki variasi yang fluktuatif. Pada sisi CNN *detection rate* memiliki pola dimana semakin besar ukuran paket, maka semakin lama pula proses pendeteksian pada sisi CNN nya. Untuk total detection rate diperhitungkan terhadap ukuran paket



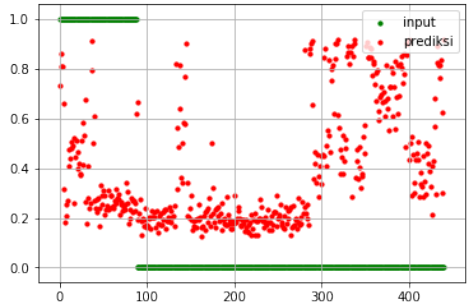
(a) Prediksi train lr=0.1



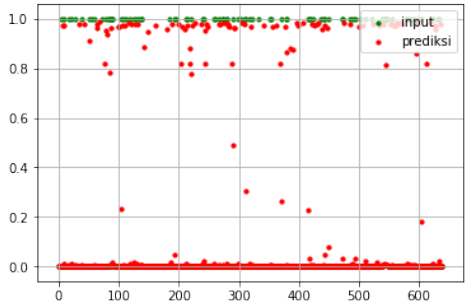
(b) Prediksi train lr=0.01



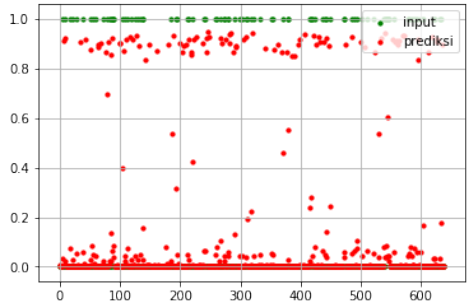
(c) Prediksi train lr=0.001



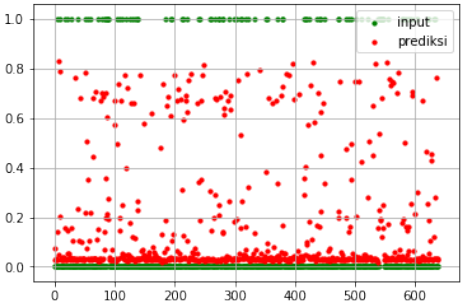
(d) Prediksi train lr=0.0001



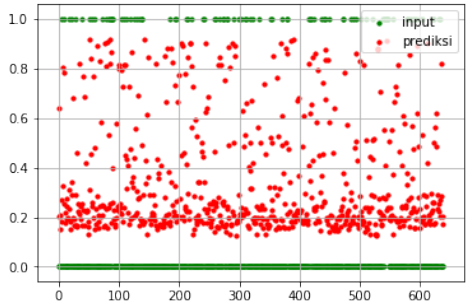
(e) Prediksi train lr=0.1



(f) Prediksi test lr=0.01

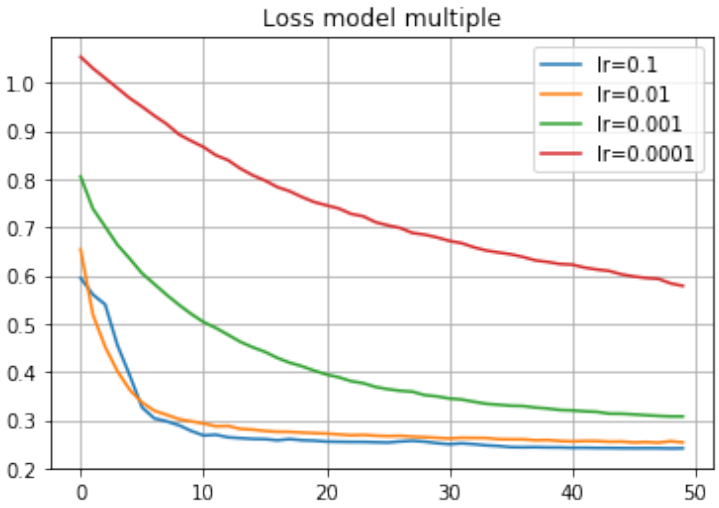


(g) Prediksi test lr=0.001

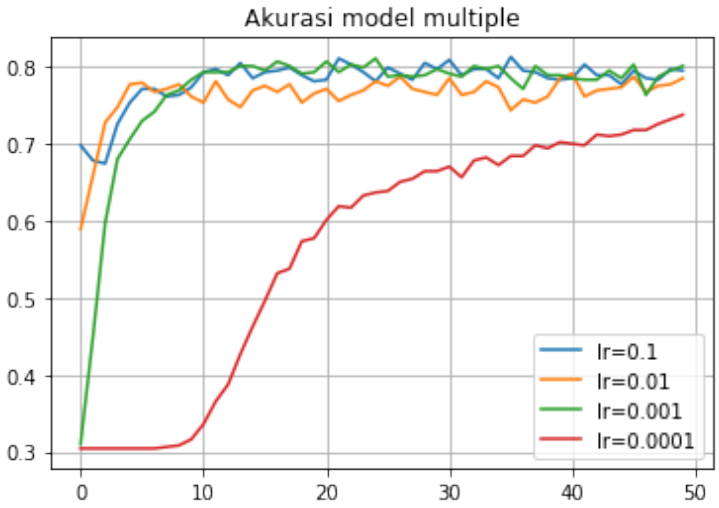


(h) Prediksi test lr=0.0001

Gambar 4.72: Grafik Prediksi Multiple CNN



Gambar 4.73: Grafik Loss Multiple CNN



Gambar 4.74: Grafik Akurasi Multiple CNN

per total waktu pada Snort CNN dan LSTM, diperoleh Rate yang fluktuatif.

BAB 5

PENUTUP

5.1 Kesimpulan

1. Hasil yang diperoleh pada metode LSTM untuk proses profiling trafik yang terbaik diperoleh dengan menggunakan metode 4 directional header, dapat dilihat dari konvergensi dan ketepatan prediksi data dengan input data.

2. Hasil yang diperoleh pada metode CNN untuk proses filtering payload memiliki pola semakin tinggi learning rate yang digunakan maka konvergensi atau peningkatan akurasi dan pengurangan loss terjadi lebih cepat beberapa epoch dari learning rate yang rendah.

3. Hasil detection rate memiliki pola paling jelas terlihat pada bagian CNN, dimana semakin besar ukuran paket maka semakin lama pula proses pendeteksian terjadi. Kecepatan pendeteksian total diperoleh fluktuatif.

5.2 Saran

1. Agar integrasi Sistem Pendeteksi Intrusi Snort dan Keras framework dapat dioptimasi sehingga pendeteksian dapat terjadi dengan cepat atau mendekati realtime. Penelitian sebaiknya dilakukan dengan hardware yang memiliki daya komputasi yang kuat.

2. Agar mekanisme Sistem Pendeteksi Intrusi dapat diterapkan secara portabel, perlu adanya sistem yang dapat mengkonversi dari satu model dengan backend lain selain keras.

DAFTAR PUSTAKA

Axelsson, S. (2000), Intrusion detection systems: A survey and taxonomy, Technical report, Technical report.

Carr, J. (2007), 'Snort: Open source network intrusion prevention', *Esecurity Planet, http://www. esecurityplanet. com/prevention/article. php/ 3681296/Snort-Open-Source-Network- Intrusion-Prevention. htm* .

Gers, F. A., Schmidhuber, J. & Cummins, F. (1999), 'Learning to forget: Continual prediction with lstm'.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep learning*, MIT press.

Greenemeier, L. (2006), Sourcefire has big plans for open-source snort, Technical report, Retrieved 2010-06-23.

Hassoun, M. H. et al. (1995), *Fundamentals of artificial neural networks*, MIT press.

Jacobus, A. & Winarko, E. (2014), 'Penerapan metode support vector machine pada sistem deteksi intrusi secara real-time', *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)* **8**(1), 13–24.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C. & Tung, K.-Y. (2013), 'Intrusion detection system: A comprehensive review', *Journal of Network and Computer Applications* **36**(1), 16–24.

Martellini, M. & Malizia, A. (2017), *Cyber and chemical, biological, radiological, nuclear, explosives challenges*, Springer.

Newman, R. C. (2009), *Computer security: Protecting digital resources*, Jones & Bartlett Publishers.

Stanger, J. (2011), *How to Cheat at Securing Linux*, Elsevier.

Wirawan, I. N. T. & Eksistyanto, I. (2015), 'Penerapan naive bayes pada intrusion detection system dengan diskritisasi variabel', *JUTI: Jurnal Ilmiah Teknologi Informasi* **13**(2), 182–189.

Zhang, Z., Li, J., Manikopoulos, C., Jorgenson, J. & Ucles, J. (2001), Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification, *in* 'Proc. IEEE Workshop on Information Assurance and Security', pp. 85–90.