INTRUSION DETECTION SYSTEM MENGGUNAKAN *DEEP LEARNING* UNTUK DETEKSI SERANGAN DoS

TUGAS AKHIR

Program Studi
S1 Teknik Komputer

Oleh :
ANDRE ARTA KURNIAWAN
16410200016

FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA
2020

INTRUSION DETECTION SYSTEM MENGGUNAKAN *DEEP LEARNING* UNTUK DETEKSI SERANGAN DoS

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk menyelesaikan
Program Sarjana Teknik

Oleh :

Nama	**: Andre Arta Kurniawan**
NIM	**: 16410200016**
Program Studi	**: S1 Teknik Komputer**

FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA
2020

Tugas Akhir

INTRUSION DETECTION SYSTEM MENGGUNAKAN *DEEP LEARNING* UNTUK DETEKSI SERANGAN DoS

Dipersiapkan dan disusun oleh

Andre Arta Kurniawan

NIM : 16410200016

Telah diperiksa, diuji, dan disetujui oleh Dewan Pembahas

Pada : 6 Agustus 2020

Susunan Dewan Pembahas

Pembimbing:

I. Dr. Jusak

NIDN. 0708017101

Digitally signed by Universitas Dinamika
Date: 2020.09.02 10:13:34 +07'00'

II. Musayyanah, S.ST., M.T.

NIDN. 0730069102

Digitally signed by Universitas Dinamika
Date: 2020.09.03 05:25:02 +07'00'

Pembahas:

Heri Pratikno, M.T., MTCNA., MTCRE.

NIDN. 0716117302

Digitally signed by Universitas Dinamika
Date: 2020.09.01 16:54:20 +07'00'

Tugas Akhir ini telah diterima sebagai salah satu persyaratan

untuk memperoleh gelar Sarjana

Digitally signed by Universitas Dinamika
Date: 2020.09.03 23:48:45 +07'00'

Dr. Jusak

NIDN. 0708017101

Dekan Fakultas Teknologi dan Informatika

UNIVERSITAS DINAMIKA

"Learn From The Past"

Andre Arta Kurniawan

Kupersembahkan Tugas Akhir ini kepada

Tuhan Yesus Kristus yang selalu mempimpin setiap langkah dalam pengerjaan Tugas Akhir saya.

Kedua Orang Tua dan seluruh keluarga yang selalu mendukung, memotivasi, memberikan semangat dan mendoakan yang terbaik untuk saya.

Teman – teman S1 Teknik Komputer dan teman – teman seangkatan seperjuangan yang membantu, mendukung dan memotivasi untuk menjadi pribadi yang lebih baik lagi.

SURAT PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa Fakultas Teknologi Informatika Universitas Dinamika, saya :

Nama	: Andre Arta Kurniawan
NIM	: 16410200016
Program Studi	: S1 Teknik Komputer
Fakultas	: Fakultas Teknologi dan Informatika
Jenis Karya	: Tugas Akhir
Judul Karya	: ***INTRUSION DETECTION SYSTEM* MENGGUNAKAN *DEEP LEARNING* UNTUK DETEKSI SERANGAN DoS**

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, saya menyetujui memberikan kepada Fakultas Teknologi Informatika Universitas Dinamika Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalti Free Right)* atas seluruh isi/ sebagian karya ilmiah saya tersebut di atas untuk disimpan, dialihmediakan dan dikelola dalam bentuk pangkalan data *(database)* untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta
2. Karya tersebut di atas adalah karya asli saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya atau pendapat orang lain yang ada dalam karya ilmiah ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiat pada karya ilmiah ini, maka saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, 24 Agustus 2020

Yang Menyatakan,

Andre Arta Kurniawan

NIM.16410200016

ABSTRAK

Serangan dalam jaringan komputer ataupun internet ke penggunanya dalam era teknologi ini sangat beragam, banyaknya kejadian atau kasus yang telah terjadi membuat ancaman keamanan dalam menggunakan internet atau jaringan komputer menjadi fokus utama. Dari banyaknya jenis serangan yang ada satu yang paling umum digunakan yaitu serangan *Denial of Service* atau sering disebut dengan DoS *attack.* Dengan mengandalkan serangan ini pelaku menyerang target – target dalam jangkauannya dengan teknik membanjiri *packet* atau *request* ke komputer target secara terus menerus dan di saat waktu yang bersamaan hingga membuat komputer target tidak dapat menanggapi *packet* atau *request* tersebut. Dengan banyaknya kejadian dan cara menyerang, dibutuhkan tindakan yang harus dilakukan untuk mencegah serangan terjadi. *Intrusion Detection System* adalah salah satu cara bagaimana mendeteksi sebuah serangan terjadi apabila terjadi serangan pada sebuah komputer atau server atau jaringan komputer. IDS akan memonitor lalu lintas jaringan, namun karena IDS dibutuhkan tindakan *maintenance* untuk memberitahu serangan dengan karakteristik tersendiri dan ditambah dengan majunya sistem teknologi membuat IDS memiliki keterbatasan dalam penerapannya. Dalam beberapa tahun terakhir penerapan *Deep Learning* mulai banyak dipergunakan, salah satunya diterapkan pada masalah IDS. Pada penelitian Tugas Akhir ini dibangun sistem IDS dengan menggunakan *Deep Learning* yang diharapkan dapat mengetahui berapa besar tingkat akurasi serangan DoS yang ada dan menghitung *True Positive Rate* dan *False Positive Rate*. Adapun persiapan *dataset* yang diperlukan sebagai data *input Deep Learning* yang diambil dari hasil *log wireshark* kemudian dilakukan data normalisasi lalu di*input* ke dalam CNN VGG-19 sebagai *Deep Learning* yang digunakan. Dari hasil pengujian yang telah dilakukan rata – rata akurasi yang dihasilkan sebesar 99.32% dengan rata – rata *loss* sebesar 4.08%. Akurasi terhadap variasi iterasi proses *training* rata – rata sebesar 99.17% dengan *loss* rata – rata sebesar 4.46% serta hasil *ROC Curve* untuk *True Positive Rate* dan *False Positive Rate* sebesar 1.

Kata Kunci: *Intrusion Detection System, DoS Attack, Deep Learning, Convolutional Neural Network, Wireshark*

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepada Tuhan Yesus Kristus, karena atas segala rahmat dan hikmat-Nya, penulis dapat menyusun dan menyelesaikan Tugas Akhir pada Program Studi S1 Teknik Komputer di Fakultas Teknologi Informatika Universitas Dinamika ini dengan baik.

Dalam penyusunannya, penulis mendapatkan banyak bimbingan serta dorongan semangat penuh cinta dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan terima kasih yang setulus - tulusnya kepada:

1. Orang tua, saudara-saudara, dan keluarga yang penulis cintai yang dengan sangat tulus memberikan doa dan semangat.
2. Kepada Bapak Dr. Jusak dan Ibu Musayyanah, S.ST., M.T. selaku dosen pembimbing yang rela membagi waktunya untuk bimbingan *online*, walaupun bapak dan ibu banyak urusan itu ini dan lagi lagi tetap revisi, bapak dan ibu tetap sabar mau membimbing.
3. Kepada Bapak Heri Pratikno, M.T., MTCNA., MTCRE. selaku dosen pembahas, terima kasih atas arahan dan masukan yang bapak berikan untuk Tugas Akhir ini.
4. Kepada Bapak Pauladie Susanto, S.Kom., M.T., selaku Ketua Program Studi S1 Teknik Komputer Universitas Dinamika, terima kasih atas ijin yang diberikan untuk mengerjakan Tugas Akhir ini.
5. Semua staff dosen dan laboran yang telah berdedikasi untuk mengajar dan memberikan ilmunya.
6. Teman – teman S1 Teknik Komputer khususnya teman – teman seangkatan seperjuangan angkatan 2016 yang akan menjadi calon wisudawan 2020, kalian hebat, terima kasih telah saling mendukung.
7. Serta semua pihak lain yang tidak dapat disebutkan secara satu per satu, yang telah membantu dalam menyelesaikan Tugas Akhir ini baik secara langsung maupun tidak langsung.

Karena pandemi *Covid*-19 proses yang penulis lalui jelas berbeda dengan proses pada umumnya. Semua proses pengerjaan dan penyusunan Tugas Akhir hingga sidang yang penulis lalui serba *online*, tapi penulis yakin bahwa penulis tetap bisa melalui perjalanan akhir studi ini di masa pandemi *Covid*-19. Kata pengantar ini jelas tak akan cukup untuk mendeskripsikan rasa terima kasih yang ingin penulis sampaikan kepada semua pihak yang terlibat.

Penulis berharap semoga laporan ini dapat berguna dan bermanfaat untuk menambah wawasan bagi pembacanya. Penulis juga menyadari dalam penulisan buku Tugas Akhir ini masih terdapat banyak kekurangan. Oleh karena itu penulis berharap adanya saran maupun kritik dalam memperbaiki kekurangan dan berusaha untuk lebih baik lagi kedepannya. Untuk siapapun juga yang tertarik dengan tema Tugas Akhir penulis kerjakan sangat diijinkan untuk melakukan *research* lebih lanjut. Semangat!

Surabaya, Agustus 2020

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB I
PENDAHULUAN

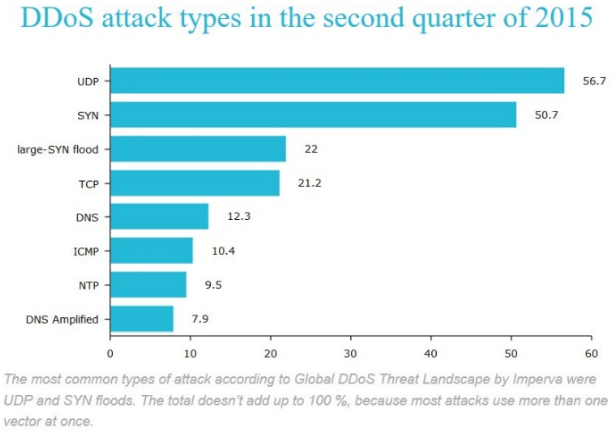
1.1 Latar Belakang Masalah

Serangan – serangan dalam internet ke pengguna internet sangat beragam, dalam contoh realitanya misalnya banyak orang yang tidak bisa mengakses internet *banking* mereka karena diretas (contoh kasus, "Bobol Akun e-Banking, Pemilik Kehilangan Uang Rp1 Miliar" (Abdi, 2019)), adapula seorang yang menggunakan internet tiba tiba *smartphone* milik mereka terkunci dengan sendirinya (contoh kasus, "Hati-hati Pakai WiFi Gratisan, HP Bisa Diretas *Hacker* Jahat" (Franedya, 2019)), atau sebuah perusahaan yang memilki sebuah server yang kemudian server tersebut tidak dapat diakses.

Dari macam – macam kejadian tersebut ancaman keamanan dalam internet pun harus menjadi fokus utama. Terdapat pula berbagai macam serangan yang dapat dilakukan untuk mengancam sebuah keamanan salah satu diantaranya yang paling umum adalah serangan *Denial of Service* (DoS). Serangan ini paling umum digunakan ketika seorang peretas mencegah pengguna yang berada dalam jaringan atau internet dari akses layanan normal untuk mengambil alih *server* atau berkomunikasi dengan *server*. Dengan berhasilnya peretas mencegah pengguna dari akses layanan normal, peretas akan menggunakan *resources* dari pengguna seperti mengkonsumsi sumber daya jaringan komputer, memori hingga prosesor secara berlebihan (Dogru & Masetic, 2017). Contoh teknik serangan DoS yang banyak digunakan adalah dengan "membanjiri" (*flood)* jaringan komputer dengan banyak *request* pada saat yang bersamaan sehingga membuat *server* tidak dapat menanggapi *request* tersebut yang kemudian berujung kelumpuhan pada *server* atau pada jaringan komputer tersebut. (Dogru & Masetic, 2017). Dengan banyaknya macam – macam serangan yang ada pada keamanan jaringan maka sangat dibutuhkan pula tindakan – tindakan yang harus dilakukan untuk mencegah serangan terjadi.

Intrusion Detection System (IDS) adalah salah satu cara bagaimana mendeteksi sebuah serangan apabila terjadi serangan pada sebuah *server.* IDS akan bekerja dengan cara memonitor lalu lintas jaringan atau aktivitas pada jaringan secara *real-time* dan kemudian akan memberikan peringatan kepada administrator apabila terdapat sebuah aktivitas yang tak biasa pada sebuah jaringan. Namun karena majunya sistem teknologi di era jaman sekarang membuat *Intrusion Detection System* memiliki keterbatasan dalam penerapannya dikarenakan serangan – serangan pada jaringan komputer atau *server* semakin canggih (Zamani & Movahedi, 2013), sehingga diperlukan pengembangan atau cara lain dalam mendeteksi sebuah serangan. Dalam beberapa tahun terakhir telah muncul istilah baru yang kini banyak diterapkan yaitu *Machine Learning. Machine Learning* adalah bidang ilmu komputer yang menggunakan teknik statistika untuk memberi kemampuan sistem komputer agar dapat belajar dari data, tanpa diprogram secara eksplisit (Purnama, 2019). Banyak bidang yang telah menerapkan *machine learning,* salah satunya diterapkan pada masalah IDS dengan harapan dapat meningkatkan tingkat deteksi dan kemampuan beradaptasi. (Zamani & Movahedi, 2013).

Dalam Tugas Akhir ini, sebuah sistem IDS untuk mendeteksi serangan DoS akan dibangun dengan menggunakan salah satu model *deep learning* yaitu algoritma *Convolutional Neural Network* (CNN). Penelitian terdahulu pernah dilakukan dalam membuat sistem IDS menggunakan model *deep learning* dengan algoritma CNN. Namun, pada penelitian tersebut digunakan juga metode lain sebagai pembanding dengan CNN dan data yang digunakan pada penelitian tersebut juga mengambil dari *dataset* KDD 1999 (Sinh-Ngoc Nguyen, Van-Quyet Nguyen, Jintae Choi, & Kyungbaek Kim, 2018). Dalam Tugas Akhir ini penulis akan menggunakan beberapa macam jenis serangan DoS dimana jenis serangan DoS yang digunakan adalah *SYN Flood, UDP Flood,*dan *Ping of Death.* Beberapa macam jenis serangan DoS yang digunakan ini berdasarkan data pada tahun 2015 yang menunjukkan jenis serangan DoS yang paling umum digunakan. Serta hasil *log wireshark* sebagai data proses *training* dan juga akan menggunakan dataset ISCX-IDS-2012 sebagai data tambahan untuk melakukan proses *training.*



Gambar 1.1 Jenis serangan DoS yang paling u mum digunakan pada tahun 2015

(Sumber : https://www.masterdc.com)

1.2 Rumusan Masalah

Rumusan masalah dalam Tugas Akhir ini adalah :

1. Berapa besar tingkat akurasi CNN *VGG-19* untuk mendeteksi serangan DoS
2. Bagaimana pengaruh iterasi proses data *training* CNN *VGG-19* terhadap hasil akurasi dalam mendeteksi serangan DoS
3. Bagaimana menghitung *True Positive Rate* (TPR) dan *False Positive Rate* (FPR) dari hasil deteksi serangan DoS

1.3 Batasan Masalah

Dalam pembuatan Tugas Akhir ini, terdapat beberapa batasan masalah antara lain :

1. *Capture tools* yang akan digunakan dalam menangkap lalu lintas jaringan menggunakan *tools wireshark*
2. Data yang akan digunakan untuk proses *training* akan diambil dari hasil *log wireshark* dan dataset ISCX-IDS-2012
3. Data yang digunakan dari hasil *log wireshark* adalah hasil *capture* serangan DoS
4. Jenis serangan DoS yang digunakan adalah *Ping of Death, UDP Flood,* dan *SYN Flood.*

1.4 Tujuan

Penelitian ini bertujuan untuk mendeteksi apakah terdapat serangan DoS atau bukan serangan DoS, menguji akurasinya dan menghitung True Positive Rate CNN VGG-19 dan False Positive Rate CNN VGG-19 dalam mendeteksi serangan DoS serta mengamati pengaruh terhadap lamanya proses komputasi data training CNN VGG-19 terhadap hasil akurasi dalam mendeteksi serangan DoS.

BAB II
LANDASAN TEORI

2.1 *Intrusion Detection System*

Intrusion detection adalah proses *monitoring event* yang terjadi dalam suatu jaringan atau suatu sistem komputer dan menganalisanya untuk mengetahui adanya tanda-tanda insiden yang mungkin terjadi (Nugroho, Rochim, & Widianto, 2015). IDS akan memonitor dan memberi *alert* (peringatan) apakah suatu aktifitas tergolong *malicious* (berbahaya) atau tidak dan menggolongkannya ke dalam beberapa tingkatan *risk level* (Azaim , 2020).

Pada dasarnya, IDS bekerja sama hal nya seperti sebuah *alarm.* IDS tidak bisa menghentikan aksi pencuri untuk mencuri, IDS hanya akan bekerja dengan memberi peringatan ketika pencuri tersebut masuk ke dalam rumah. Ada beberapa hal yang bisa dan tidak bisa dilakukan oleh IDS diantaranya, IDS dapat melacak sebuah serangan dari serangan tersebut masuk kedalam jaringan hingga akhir berhentinya serangan tersebut. Selain itu, IDS dapat mendeteksi suatu kesalahan dalam konfigurasi yang telah dibuat di dalam sistem. Di sisi lain, ketika jaringan yang digunakan sibuk dan memiliki *traffic* yang besar IDS tidak bisa menganalisa semua paket yang diterimanya dan IDS tidak bisa menganalisa suatu jenis serangan baru (Rozenblum, 2001).

Ada 2 jenis IDS :

1. ***Network Intrusion Detection System*** **(NIDS)**

Tugasnya memonitor dan menganalisis *traffic* pada keseluruhan *subnet network*, dimana NIDS ini akan meng-*capture* semua *traffic* seperti sebuah *sniffer*.

2. ***Host Intrusion Detection System*** **(HIDS)**

Tugasnya memonitor dan menganalisis *traffic network* yang berasal dan keluar dari sebuah *host* dimana perangkat HIDS tersebut diimplementasi. Perbedaannya dengan NIDS, HIDS hanya memonitor spesifik host saja, sedangkan NIDS memonitor seluruh subnet. Karena memonitor spesifik *host*, maka HIDS lebih dapat ‘melihat banyak’ aktifitas yang ada pada suatu *host*, seperti dapat

melihat apakah pada suatu host tersebut terdapat aktifitas *port scan*, malware ataupun adanya *vulnerability* pada *host* tersebut, dengan cara memonitor log yang dikirimkan oleh *host* tersebut.

Pada NIDS, IDS menggunakan teknik dalam melakukan pendeteksian diantaranya :

1. *Signatured-Based Detection*

Suatu aktifitas (khususnya yang berupa serangan) memiliki karakterisitik tersendiri atau memiliki ciri jejak (*footprint*) tersendiri yang membedakan antara satu dengan yang lainnya. *Footprint* tersebut dapat diistilahkan sebagai *signature*. *Footprint* tersebut lalu dikumpulkan menjadi sebuah '*database' signature*. Berdasarkan *footprint* tersebutlah, kemudian digunakan untuk mendeteksi serangan yang sama di kemudian hari. Jadi IDS akan menganalisis setiap aktifitas (berupa *packet*, *log*, dll) pada *network* dan sistem kita, lalu apabila ada yang cocok ciri-cirinya dengan *signature* yang ter-*record*, maka serangan tersebut dapat diklasifikasikan.

2. *Anomaly-Based Detection*

Suatu sistem tentu dibuat untuk memiliki *behavior* dan konfigurasi tertentu. Jika suatu sistem melakukan aktifitas yang bukan berdasarkan behavior dan konfigurasi yang telah ditentukan, itulah yang dinamakan sebuah *anomaly*. IDS akan melakukan *baselining* dan *learning* pada *pattern* dari aktifitas normal sistem untuk mendeteksi adanya intrusi. Misalnya pada server kita terdapat aktifitas *download* file *binary* yang sebenarnya tidak kita kehendaki atau pada server kita terdapat perubahan konfigurasi *registry* yang tidak dikehendaki. Pada IDS, penyimpangan dari *baseline* behavior tersebut akan menyebabkan munculnya alarm.

2.2 *Denial of Service*

Serangan DoS (*Denial of Service*) dan DDoS (*Distributed Denial of Service*) merupakan serangan yang sering dijumpai pada beberapa kasus *cybercrime*. Pada dasarnya DoS dan DDoS adalah sebuah serangan yang sama, namun DDoS adalah sebuah serangan DoS yang dapat dikatakan terstruktur. Dengan mekanisme yang

sama dengan DoS namun dampak yang ditimbulkan oleh serangan DDoS jauh lebih besar daripada serangan DoS (Abdillah, 2015).

Jenis – jenis serangan DoS :

1. *Ping of Death*

Serangan ini umum dilakukan dalam melakukan serangan DoS dan dapat menggunakan *utility ping* pada sistem operasi. *Ping* digunakan untuk mengecek keberadaan suatu *host* di dalam jaringan tersebut. Ketika menggunakan *ping* biasanya data yang akan dikirim secara *default* yaitu 32 *bytes.* Namun *ping* sendiri dapat mengirim data lebih banyak lagi hingga maksimum 65 *Kbytes.*

2. UDP *Flood*

Serangan ini menggunakan protokol UDP dengan memanfaatkan karakteristik protokol UDP yang bersifat *connectionless* untuk menyerang target. Dengan memanfaatkan protokol UDP data dengan jumlah yang besar akan langsung dikirim begitu saja kepada korban. Kondisi ini akan membuat komputer korban tidak siap menerima terlalu banyak.

3. SYN *Flooding*

Serangan ini dilakukan pada saat proses *handshake,* dimana ketika proses *handshake* mau dilakukan komputer pengirim akan mengirim paket SYN kepada komputer penerima yang kemudian akan dibalas dengan paket SYN ACK kepada komputer pengirim. Sesaat setelah proses ini, seharusnya komputer pengirim mengirim paket ACK dan kemudian dilakukan proses *handshake.* Namun kenyataannya, penyerang mengirim kembali paket SYN kepada komputer target dalam jumlah yang banyak sehingga harus membuat komputer target menanggapi permintaan ini.

4. *Remote Control Attack*

Serangan ini dikategorikan serangan DoS terdistribusi karena sebelum penyerang melakukan aksinya untuk menyerang komputer target, penyerang terlebih dulu akan menginfeksi komputer – komputer dalam jaringan yang kemudian akan digunakan untuk menyerang target. Dalam beberapa kasus,

serangan ini akan dilakukan untuk menyerang sebuah server yang memiliki *bandwidth* lebar.

5. *Smurf Attack*

Penyerangan dengan memanfaatkan ICMP *echo request* yang sering digunakan pada saat mem*broadcast* identitas kepada *broadcast address* dalam sebuah *network*. Saat melakukan *broadcast* pada *broadcast address* maka semua komputer yang terkoneksi kedalam jaringan akan ikut menjawab *request* tersebut. Hal ini tentu saja akan melambatkan dan memadatkan trafik di jaringan karena komputer – komputer yang tidak ditanya turut memberikan *request* tersebut.

2.3 *Deep Learning*

Deep Learning merupakan metode *learning* yang memanfaaatkan *Artificial Neural Networks* yang berlapis – lapis (*multi layer). Artificial Neural Networks* ini dibuat mirip dengan otak manusia, dimana *neuron – neuron* terkoneksi satu sama lain sehingga membentuk sebuah jaringan *neuron* yang sangat rumit (Primartha, 2018).

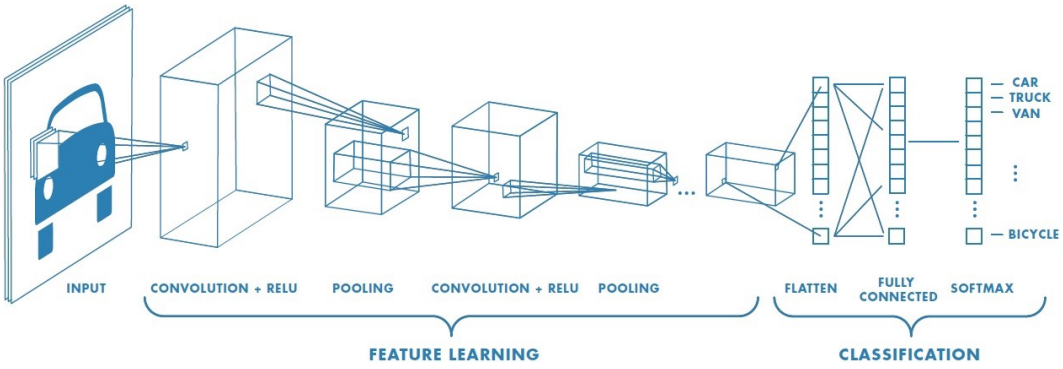
Dalam beberapa buku referensi mengenai definisi *Deep Learning, Deep Learning* memiliki definisi sebagai modernisasi *machine learning* untuk menangani *big data* (Suyanto, Ramadhani, & Mandala, 2019). Ada pun yang menyebutkan bahwa *Deep Learning* dapat dipandang sebagai gabungan *Machine Learning* dengan *Artificial Intelligence* (Primartha, 2018).

2.3.1 Convolutional Neural Network

Langkah awal dalam memahami CNN adalah memahami konvolusi itu sendiri. Dalam konteks jaringan saraf tiruan, konvolusi dilakukan diantara tensor. 2 *input* tensor akan dikalikan yang kemudian akan menghasilkan 1 tensor baru sebagai *output* (Michelucci, 2019).

Secara prinsip, CNN meniru *visual cortex* pada binatang. CNN menggunakan arsitektur 3 dimensi yaitu panjang, lebar, dan tinggi. Tak jauh berbeda dengan *neural network* yang lain, yang membedakan hanyalah pada arsitektur CNN yang terdiri atas *input layer, output layer* dan *hidden layers.* Pada *hidden layers* terdiri 2

layer yang umunya sering digunakan oleh CNN yaitu yang sering disebut dengan *Feature Extraction Layer* (Sena, 2017).



Gambar 2.1 *Convulotional Neural Network* Model

(Sumber : https://medium.com/@samuelsena)

A. *Feature Extraction Layer*

Pada *layer* ini dilakukan proses *encoding,* yang merubah data inputan menjadi angka – angka yang merepresentasikan data inputan tersebut. Pada *layer* ini juga terdiri dari 2 *layer* yang umum terdapat pada CNN yaitu *Convulotional Layer* dan *Pooling layer.*

A.1 *Convolution layer*

Pada *convolution layer,* parameter yang dipelajari adalah *filter.* Misalnya, jika memiliki 32 *filter,* masing – masing dimensi 5x5, maka parameter total yang akan dipelajari sebanyak, 5x5x32=832 parameter (Michelucci, 2019). Maksudnya adalah pada *convolution layer* data dari inputan akan diambil dimensi panjangxtinggi sesuai ukuran dimensi *filter* dari seluruh total dimensi data inputan. Misalnya, ukuran dimensi *filter* 3x3 dan ukuran dimensi data input 5x5 maka dimensi dari data input akan diambil sesuai dengan ukuran *filter* yaitu 3x3, kemudian akan dilakukan proses perkalian sebanyak jumlah *filter* yang digunakan. *Convolution layer* dalam arsitektur CNN umumnya menggunakan lebih daari satu *filter* (Suyanto, Ramadhani, & Mandala, 2019).

A.2 *Pooling layer*

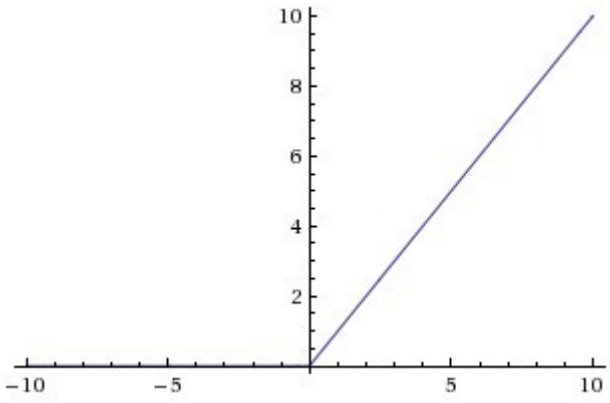
Pooling layer berfungsi menjaga ukuran data ketika *convolution* dilakukan, yaitu dengan melakukan reduksi sampel (*downsampling)* (Suyanto, Ramadhani, & Mandala, 2019). Pada umumnya, proses *pooling* dilakukan menggunakan *max pooling* atau *average pooling. Max pooling* digunakan dengan mengambil nilai yang paling besar sedangkan untuk *average pooling* digunakan dengan mengambil nilai rata – rata. Dari kedua cara proses *pooling* yang paling sering dijumpai adalah menggunakan *max pooling,* untuk *average pooling* sangat jarang digunakan tapi dalam beberapa arsitektur jaringan dapat ditemukan (Michelucci, 2019).

B. *Normalization Layer*

Normalization layer berguna untuk mengatasi perbedaan rentang nilai yang signifikan. Namun saat ini *normalization layer* masih tidak banyak digunakan karena efek yang diberikan pada layer ini tidak begitu besar (Suyanto, Ramadhani, & Mandala, 2019).

C. *ReLU Layer*

ReLU layer adalah bagian dari proses *activation function,* dimana *activation function* berguna untuk membuat *neural network* menjadi non-linear (Hatta, 2017). *ReLU* atau *Rectified Linear Units* ini mengaplikasikan aktivasi f(x) = max(0,x), yang berarti jika nilai x <= 0 maka nilai x = 0 dan jika nilai x > 0 maka nilai x = x.



Gambar 2.2 *ReLU activation function*

(Sumber : https://medium.com/@opam22)

D. *Fully Connected Layer*

Fully connected layer adalah proses dimana setiap *neurons* memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya. (Suyanto, Ramadhani, & Mandala, 2019)

E. *Loss Layer*

Loss layer adalah lapisan terakhir dalam CNN dimana pada proses ini akan memperlihatkan hasil prediksi dan nilai *loss* pada saat proses *training.*

2.3.2 Arsitektur CNN

Pada awal kemunculan CNN, arsitektur CNN didesain dengan hanya berisi sedikit *layer* (kurang dari 10), kemudian perkembangan CNN yang sangat pesat dalam beberapa tahun membuat CNN memiliki variasi arsitektur CNN dengan ratusan hingga ribuan *layer* (Suyanto, Ramadhani, & Mandala, 2019).

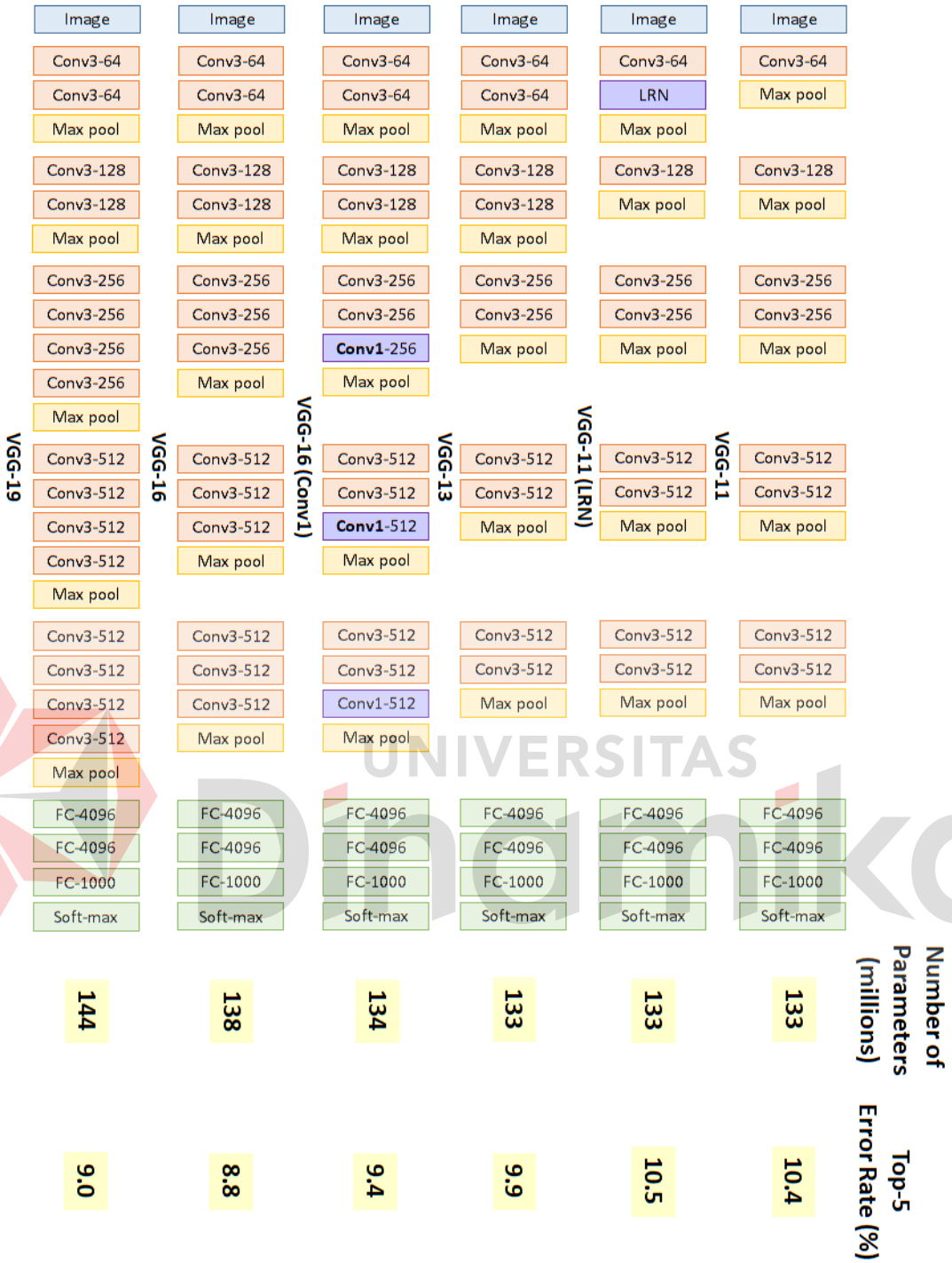
A. *Visual Geometry Group* (VGG)

Menjuarai posisi *1st runner up* dalam ajang lomba *ImageNet Challenge* – ILSVRC 2014 VGG pada awalnya dibangun oleh Karen Simonyan dan Andrew Zisserman (Suyanto, Ramadhani, & Mandala, 2019), VGG awalnya dibentuk bersama kelompok peniliti dari *University of Oxford* yang kemudian VGG membuat sebuah aristektur jaringan untuk CNN yang dinamai dengan *VGGNet.* Jaringan *VGGNet* versi terbaik berisi 16 *convolutional/fully connected layers* dengan arsitektur homogen yang hanya melakukan konvolusi 3x3 dan *pooling* 2x2 dari lapisan awal hingga akhir (Simonyan & Zisserman, 2014).

VGGNet telah memiliki banyak variasi model, diantaranya yang paling popular adalah *VGG-16* dan *VGG-19.* Selain itu, *VGGNet* memiliki banyak parameter, mencapai 140 juta, yang membuat *VGGNet* selalu membutuhkan memori yang banyak untuk memori penyimpanan. Dalam proses CNN, *VGGNet* menggunakan *filter size* 3x3 dan 2z2.

Dengan banyaknya variasi model *VGGNet, layer* yang terdapat pada *VGGNet* pun bervariasi namun pada dasarnya seluruh *layer* pada *VGGNet* akan memiliki *Convolutional layers, Max Pooling layers, Activation layers, Fully connected layers*. Misalnya pada *VGG-16, VGG-16* terdiri atas 13 *Convolutional layer,* 5 *Max*

Pooling layer, dan 3 *Fully-Connected Layer.* Jika setiap *layer* dijumlahkan akan menghasilkan total 21 *layer* tetapi hanya total 16 *layer* yang memiliki bobot, ini dikarenakan pada bagian *pooling layer* tidak memiliki bobot. *VGGNet* memiliki 5 kali proses konvolusi dimana setiap proses konvolusi memiliki jumlah *filter* yang berbeda – beda, dimana *Conv-1* memiliki 64 *filter, Conv-2* memiliki 128 *filter, Conv-3* memiliki 256 *filter, Conv-4* dan *Conv-5* memiliki 512 *filter.*



Gambar 2.3 Variasi dari arsitektur *VGGNet*

(Sumber : https://towardsdatascience.com/@BVPSK)

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Di dalam Tugas Akhir ini, terdapat beberapa tahap metode penelitian yang akan dilakukan :

3.1.1 Persiapan

Pada tahap ini dilakukan pencarian referensi terkait dengan metode *Deep Learning* yang digunakan yaitu *Convolutional Neural Network.* Selain itu dilakukan juga pencarian referensi terkait DoS dengan jenis – jenis serangan yang akan digunakan yaitu *Ping of Death, UDP Flood,* dan *SYN Flood.* Selain itu pada tahap ini juga dilakukan persiapan dan pengumpulan data yang nantinya akan digunakan sebagai *data input deep learning.* Dalam hal ini data yang akan digunakan adalah hasil *log wireshark* yang nantinya juga akan diolah sebelum di*input*kan ke dalam proses *deep learning.*

3.1.2 Analisis

Pada tahap ini akan dilakukan analisis dari arsitektur yang ada pada model *Convolutional Neural Network* dengan data *input* yang akan digunakan.

3.1.3 Desain

Pada tahap ini akan dilakukan perancangan aplikasi untuk menyelesaikan masalah yang telah dirumuskan dari tahap analisis.

3.1.4 Implementasi

Pada tahap ini akan dilakukan implementasi dari hasil analisis dan desain ke dalam bentuk program.

3.1.5 Uji Coba dan Evaluasi

Pada tahap ini akan dilakukan uji coba pada aplikasi yang telah dibuat. Uji coba dilakukan dengan cara membandingkan presentase hasil deteksi aktivitas jaringan normal dan DoS serta membandingkan hasil deteksi dari CNN *VGG-19* dengan menggunakan variasi data dan variasi jumlah iterasi yang telah ditentukan serta menentukan hasil

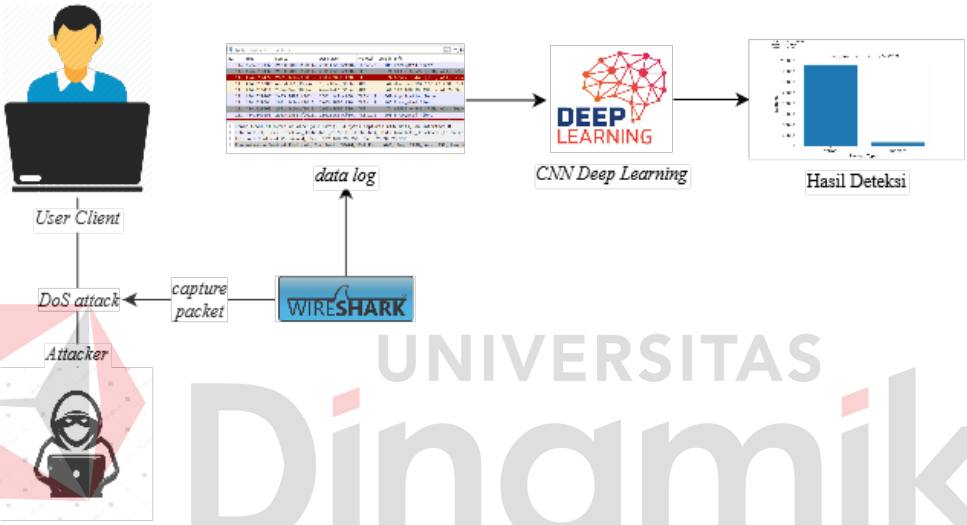
3.1.6 Penyusunan Laporan

Pada tahap ini akan dilakukan penyusunan laporan berdasarkan langkah-langkah yang dilakukan dalam proses pembuatan tugas akhir ini mulai dari tahap analisis sampai tahap evaluasi.

3.2 Gambaran Sistem

3.2.1 Rancangan Program

Pada Gambar 3.1 merupakan gambar rancangan sistem yang nantinya akan dijalankan.



Gambar 3.1 Rancangan Sistem IDS Menggunakan *Deep Learning*

Dalam rancangan sistem di Gambar 3.1 dapat dijelaskan bagaimana IDS dengan menggunakan *deep learning* mendeteksi serangan DoS. Pertama, *attacker* melakukan serangan DoS ke *client* atau *server,* Jenis serangan DoS yang digunakan adalah *Ping of Death, UDP Flood,* dan *SYN Flood.*

Pada saat proses *attacker* melakukan penyerangan DoS terhadap *client, wireshark* sebuah *program* yang umum digunakan dalam bidang jaringan komputer akan melakukan *capture packet. Capture packet* ini adalah *wireshark* menangkap seluruh aktivitas dari dan menuju *host.* Seluruh aktivitas ini nantinya akan direkam dan kemudian dapat disimpan menjadi hasil *log.*

Dari hasil *log* yang didapat, *log* akan diolah terlebih dahulu sebelum kemudian dijadikan data *input* CNN *VGG-19*. Pada *log,* akan diambil beberapa nilai parameter yang nantinya akan dijadikan parameter *input* CNN *VGG-19*. Setelah data siap, dilakukan proses konversi data dimana data *log* tadi harus diubah *format*nya menjadi data *input* yang dapat dibaca oleh CNN *VGG-19*.

Tahap selanjutnya, memasukkan data yang telah siap menjadi data *input* CNN *VGG-19* kedalam proses *deep learning.* Pada proses CNN *VGG-19* akan dilakukan proses *training* hingga nantinya memberikan *output* deteksi apakah itu serangan DoS atau bukan serangan DoS dan juga presentase dalam bentuk grafik untuk *True Positive Rate* dan *False Positive Rate*. Selain itu, pada proses CNN *VGG-19* juga akan dilakukan variasi data dan variasi lama waktu proses *training* untuk mengetahui akurasi dalam mendeteksi.

3.2.2 Dataset

Pada proses *training,* diperlukan sebuah data *input* yang kemudian akan digunakan CNN *VGG-19* untuk mempelajari data yang akan diolah. Sebagai contoh, berikut adalah contoh hasil *log* dari *wireshark* yang telah diubah kedalam bentuk tabel yang kemudian akan diubah kembali kedalam *format* data yang dapat dibaca oleh CNN *VGG-19*.

Tabel 3.1 Hasil *log wireshark* untuk *dataset training* CNN *VGG-19*

Source IP	Source Port	Destination IP	Destination Port	Protocol	Packet Length	Data Length	Data	Data Text	Info
196.189.157.81	2836	100.100.100.14	80	UDP	1042	1000	35858585858	XXXXXXXXXX	2836 > 80 Len=1000
100.100.100.14	2836	196.189.157.81	80	ICMP	590	520	35858585858	XXXXXXXXXX	Destination unreachable (Port unreachable)
141.58.193.200	2836	100.100.100.14	80	TCP	554				2836 > 80 [SYN] Seq=0 Win=512 Len=500 [TCP segment of a reassembled PDU]
100.100.100.14	80	141.58.193.200	2836	TCP	58				80 > 2836 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
28.136.224.11	2877	100.100.100.14	80	UDP	1042	1000	35858585858	XXXXXXXXXX	2877 > 80 Len=1000
100.100.100.14	2877	28.136.224.11	80	ICMP	590	520	35858585858	XXXXXXXXXX	Destination unreachable (Port unreachable)
8.4.50.159	2875	100.100.100.14	80	TCP	554				2875 > 80 [SYN] Seq=0 Win=512 Len=500 [TCP segment of a reassembled PDU]
100.100.100.14	80	8.4.50.159	2875	TCP	58				80 > 2875 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

3.2.3 Training Data

Dataset yang digunakan untuk proses *training* nantinya akan bervariasi, begitu juga dengan lamanya waktu proses *training*. Untuk variasi data yang akan digunakan diambil berdasarkan *capture packet* pada *wireshark* dengan simulasi

jaringan dan DoS *attack* yang dilakukan, dengan begitu data *file log* yang dihasilkan akan bervariasi. Selain itu, variasi lama waktu proses *training* akan dilakukan dengan mengatur jumlah iterasi yang akan dicoba sebanyak 10 kali mulai dari 10 iterasi hingga 100 iterasi dengan kelipatan 10. Parameter yang akan digunakan sebagai parameter *input* pada CNN *VGG-19* nantinya akan menggunakan *source IP address, source port, destination IP address, destination port, protocol, packet length, data length, data, data text,* dan *info.* Selain itu, presentase pembagian *dataset* yang digunakan dibagi menjadi *training data* dan *test data* dengan masing – masing presentase sebesar 80% dan 20%.

3.2.4 Data Normalization

Model CNN *VGG-19* dibuat karena terkenal dalam mengklasifikasikan sebuah gambar sehingga *format input* CNN *VGG-19* menggunakan gambar sebagai *input.* Dalam beberapa kasus, CNN *VGG-19* juga dapat diterapkan untuk mengklasifikasikan sebuah teks atau suara, sehingga data input yang digunakan bukan lagi sebuah gambar. Untuk itu, diperlukan normalisasi agar data – data yang bukan gambar ini dapat di*input*kan ke dalam model CNN *VGG-19*.

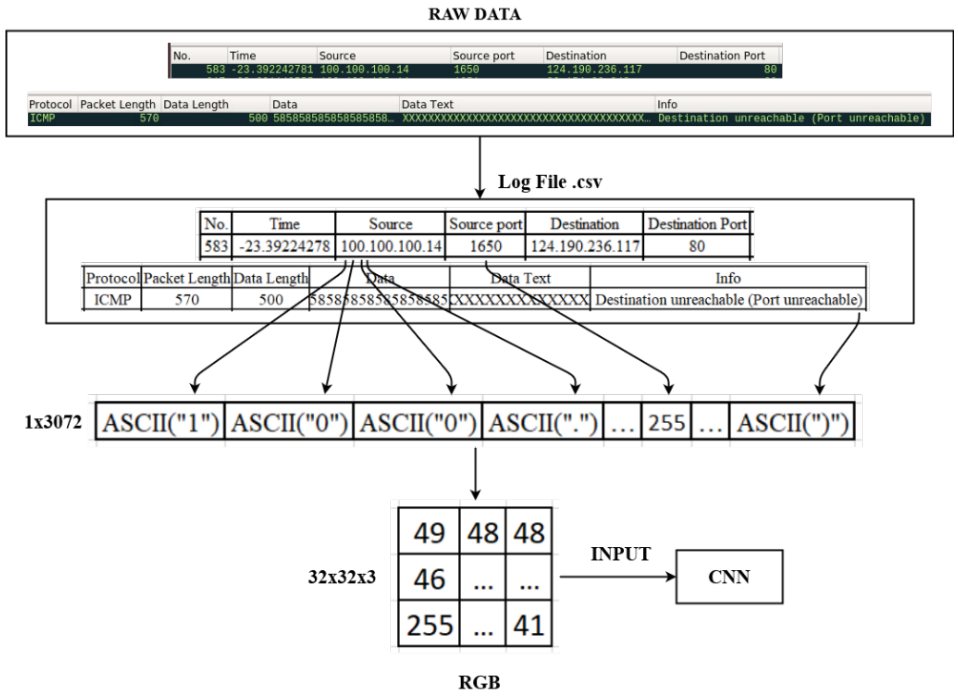
Sama hal nya seperti data gambar, nilai – nilai *pixel* dalam gambar akan diubah kedalam bentuk matriks sesuai ukuran gambar tersebut sehingga matriks *pixel* dari gambar ini nantinya akan di*input*kan ke dalam model CNN *VGG-19*. Dalam kasus data yang di*input*kan bukan berupa gambar perlu dilakukan normalisasi, tujuannya data yang bukan berupa gambar misal teks akan diubah kedalam bentuk matriks dengan nilai disetiap *pixel*nya.

Pada tugas akhir ini penulis menggunakan 10 parameter utama sebagai data yang di*input*kan kedalam CNN *VGG-19*, yaitu *source IP address, source port, destination IP address, destination port, protocol, packet length, data length, data, data text,* dan *info.* Agar data dari hasil *log* ini dapat dimasukkan kedalam model CNN *VGG-19* perlu dilakukan normalisasi dengan membuat matriks persegi. Dengan memerhatikan 10 parameter yang digunakan dan banyaknya jumlah data dari hasil *capture wireshark,* maka akan digunakan matriks persegi ukuran 32x32x3 dimana seluruh *pixel* akan digunakan atau diisi dengan nilai dari setiap parameter yang digunakan hingga memenuhi ukuran matriks.

Untuk ukuran matriks persegi yang digunakan disesuaikan dengan kebutuhan data dan arsitektur CNN *VGG-19* yang digunakan, ukuran 32x32x3 pun digunakan karena arsitektur CNN *VGG-19* yang digunakan menerima inputan minimal 32x32x3 dengan penjelasan 32x32 *pixel* yang berarti memiliki 1024 *pixel* dengan 3 channel, dimana channel yang dimaksud adalah channel nilai RGB. Dengan begitu total seluruh *pixel* yang harus diinputkan ke dalam CNN *VGG-19* adalah sebanyak 3072 *pixel*.

Nantinya nilai dari setiap parameter akan dikonversi dan diisikan kedalam setiap *pixel* matriks. Apabila data dari salah satu parameter berupa angka dan angka tersebut melebihi nilai 255, maka nilai yang akan dimasukkan kedalam *pixel* matriks sebesar 255. Ini dilakukan karena *neural network* yang digunakan menggunakan gambar sebagai *input*an dan nilai setiap *pixel* yang dimiliki sebuah gambar memiliki nilai maksimal 255. Dan apabila data dari salah satu parameter berupa *string,* maka nilai dari setiap karakter *string* akan diambil dan diubah menjadi nilai ASCII.

Jika jumlah total data yang telah dinormalisasi masih kurang dari 3072 maka jumlah total data yang sudah ada akan diduplikasi hingga memenuhi total *pixel* 3072. Selanjutnya jika sudah dilakukan normalisasi data, maka *dataset* ini akan di*input*kan kedalam model CNN *VGG-19*.



Gambar 3.2 Data Normalisasi

3.3 Evaluasi dan Pengujian

Evaluasi ini berisi uraian tentang rencana uji coba yang akan dilakukan. Yaitu dengan melakukan pengujian yang berkaitan dengan rumusan masalah dan juga tujuan dari Tugas Akhir. Adapun beberapa pengujian yang dilakukan adalah :

3.3.1 Pengujian akurasi CNN terhadap variasi data

Pengujian ini dilakukan untuk melihat seberapa akurat sistem dalam mendeteksi serangan DoS dengan data yang bervariasi. Adapun proses pengujian bisa dilakukan dengan menggunakan seperti pada Tabel 3.2 berikut :

Tabel 3.2 Pengujian akurasi CNN terhadap variasi data

No	Jumlah Total Data	Aktivitas Jaringan		Loss (%)	Akurasi (%)
		Normal (%)	DoS (%)		
1					
2					
3					
4					
5					

3.3.2 Pengujian akurasi CNN terhadap variasi *iterasi* proses *training*

Pengujian ini dilakukan untuk melihat pengaruh akurasi CNN VGG-19 terhadap variasi lama waktu proses *training.* Adapun proses pengujian bisa dilakukan dengan menggunakan seperti pada Tabel 3.3 berikut :

Tabel 3.3 Pengujian akurasi CNN VGG-19 terhadap variasi *iterasi* proses *training*

No	Jumlah Total Data	Lama Proses Training (Jumlah Iterasi)	Aktivitas Jaringan		Lama Waktu (detik)	Loss(%)	Akurasi(%)
			Normal (%)	DoS (%)			
1		10					
2		20					
3		30					
4		40					
5		50					
6		60					
7		70					
8		80					
9		90					
10		100					

3.3.3 Pengujian TPR dan FPR dari hasil deteksi serangan DoS

Pengujian TPR dan FPR atau sering disebut dengan ROC *Curve* ini dilakukan dengan menampilkan hasil grafik antara TPR dan FPR. Adapun parameter *True Positive Rate* dapat dihitung dengan menggunakan rumus :

$$\textit{True Positive Rate}\ (\text{TPR}) = \text{TP} / (\text{TP} + \text{FN}) \qquad \textbf{(3.1)}$$

Dimana :

- TPR = *True Positive Rate,* rata – rata hasil prediksi benar oleh CNN VGG-19
- TP = *True Positive*, jika serangan DoS itu ada dan benar sebuah serangan DoS
- FN = *False Negative*, jika serangan DoS itu ada dan bukan dianggap sebuah serangan DoS

Parameter *False Positive Rate* dapat dihitung dengan menggunakan rumus :

False Positive Rate (FPR) = FP / (FP + TN) **(3.2)**

Dimana :

- FPR = *False Positive Rate,* rata – rata hasil prediksi salah oleh CNN VGG-19
- FP = *False Positive*, jika serangan DoS itu tidak ada dan dianggap benar sebuah serangan DoS
- TN = *True Negative*, jika serangan DoS itu tidak ada dan benar tidak terdapat sebuah serangan DoS

BAB IV
HASIL PENGUJIAN DAN ANALISIS

Pada bab ini akan dibahas dan dijelaskan mengenai hasil pengujian dan analisis pengujian yang telah dilakukan. Pada pengujian untuk melakukan deteksi serangan DoS ini digunakan pula *deep learning* dengan arsitektur CNN *VGG-19.* Dengan menggunakan arsitektur *VGG-19* terdapat *layer – layer* yang digunakan diantaranya, *feature extraction layer* yang berisi *convolutional layer* dan *pooling layer, ReLU layer* atau *activation layer,* dan *fully-connected layer* serta *filter size* yang digunakan berukuran 3x3.

Untuk setiap *layer* yang digunakan terdapat, 16 *convolutional layer,* 5 *pooling layer,* dan 3 *fully-connected layer* dengan total 24 *layer* dan 19 *layer* yang memiliki bobot. Pada 16 *convolutional layer,* jumlah *filter* yang dimiliki berbeda – beda, dimana *Conv-1* memiliki 64 *filter, Conv-2* memiliki 128 *filter, Conv-3* memiliki 256 *filter, Conv-4* dan *Conv-5* memiliki 512 *filter.* Karena arsitektur *VGG-19* yang digunakan maka *activation layer* yang digunakan adalah *ReLU.* Sedangkan untuk *loss function* yang digunakan adalah *binary classification* karena pada penelitian ini dibutuhkan 2 *class* untuk deteksi apakah itu serangan DoS atau bukan serangan DoS.

Pada hasil pengujian ini terdapat beberapa tahap yang dilakukan dalam pengujian pada tugas akhir ini. Diantaranya sebagai berikut:

4.1 Pengujian Akurasi CNN VGG-19 terhadap Variasi Data

4.1.1 Tujuan

Tujuan dari pengujian ini dilakukan adalah untuk melihat seberapa akurat sistem dalam mendeteksi serangan DoS dengan data yang bervariasi. Data bervariasi yang dimaksud adalah data *log* hasil *capture wireshark* dimana *capture wireshark* dijalankan selama kurang lebih 5 menit, dan dari hasil *log capture wireshark* tersebut didapatkan hasil *traffic log* yang berbeda – beda dengan jumlah data yang berbeda – beda juga.

4.1.2 Peralatan yang Digunakan

1. Komputer dengan *port* HTTP atau *port* 80 aktif atau Komputer Server dengan kebutuhan :
   - *Linux Operating System.*
   - *Apache Service* untuk mengaktifkan *port* HTTP atau *port 80.*
2. Komputer dengan *Linux Operating System* dan *Hping3 Linux Tools.*
3. Komputer dengan aplikasi *Wireshark* yang telah di*install.*
4. *Google Colab Account* untuk menjalankan *Deep Learning* dan *Data Normalization.*

4.1.3 Langkah – Langkah dan Cara Pengujian

1. Pada komputer target *port* HTTP atau *port* 80 dalam kondisi aktif, atau apabila menggunakan komputer *server* sebagai komputer target *install Linux* OS dan *Apache Service.*
2. Pada komputer yang akan digunakan sebagai *attacker, install Linux* OS dan apabila belum tersedia *hping3 package*, *install hping3 package.*
3. Pada komputer dengan aplikasi *wireshark install wireshark* kemudian atur parameter pada kolom *wireshark* sesuai kebutuhan.
4. Memastikan komputer dengan aplikasi *wireshark* berada satu jaringan dengan komputer target.
5. Memastikan wireshark dapat digunakan untuk *Capture packet* pada komputer yang telah di*install* aplikasi tersebut ke *ip address* komputer target.
6. Simulasikan jaringan untuk membuat aktivitas jaringan pada komputer target. (ping, browsing internet, dll.)
7. Pada *wireshark,* apabila *traffic capture* pada *wireshark* muncul berarti *wireshark* telah berhasil melakukan *capture* pada komputer target.
8. Pada komputer *attacker,* masuk sebagai *root user* untuk mendapatkan *root privileges* kemudian buka direktori *file deployattacks.sh* dan beri hak akses *owner file deployattacks.sh* menjadi *read, write,* dan *execute.*
9. Menjalankan program *attack* menggunakan *bash shell* di terminal dengan *file deployattack.sh* ke komputer target, sesuaikan waktu *attacking* dengan kapasitas *storage* komputer saat melakukan *capture* dan ikuti arahan atau petunjuk program.

10. Membuat simulasi jaringan untuk membuat aktivitas jaringan pada komputer target.
11. Pada *wireshark,* setelah program *deployattack.sh* selesai dijalankan, *export log file* hasil *capture wireshark* menjadi *.csv file.*
12. Olah data dari *log wireshark* di dalam program *Data Normalization*
13. Analisa hasil *Deep Learning* yang telah dijalankan.

4.1.4 Hasil Pengujian

Hasil pengujian untuk variasi data dilakukan sebanyak 5 kali dengan masing – masing jumlah data yang berbeda. Proses *training* pada pengujian ini dilakukan dengan jumlah 50 iterasi. Pada Tabel 4.1 menunjukkan hasil pengujian.

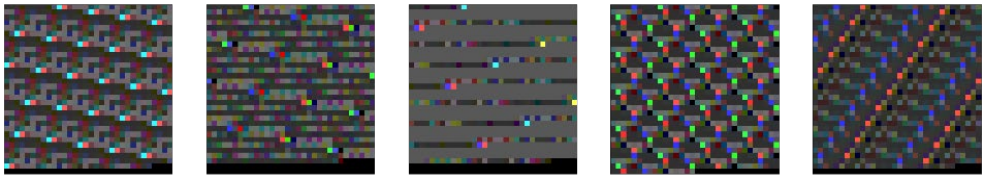
Tabel 4.1 Hasil pengujian akurasi CNN VGG-19 terhadap variasi data

No	Jumlah Total Data	Aktivitas Jaringan		Loss (%)	Akurasi (%)
		Normal (%)	DoS (%)		
1	74926	62.72%	37.28%	4.26%	99.27%
2	59934	58.42%	41.58%	4.11%	99.24%
3	50981	63.26%	36.74%	5.76%	99.33%
4	30995	69.96%	30.04%	2.96%	99.55%
5	43375	68.10%	31.90%	3.30%	99.20%

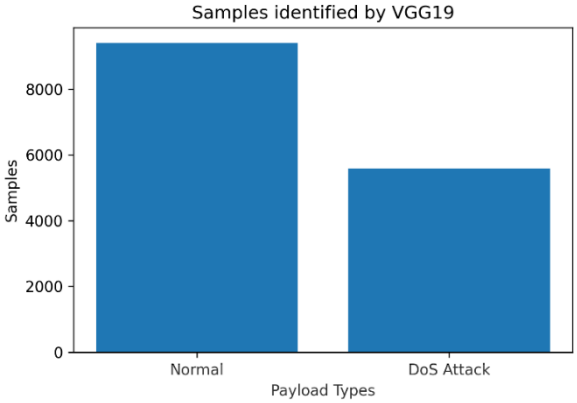
A. Pengujian ke-1

```
Normalization Data
==================
Folder selected : /content/drive/My Drive/TeA/project/dataset/DATA1
Extracting data : DATA1.csv
counting data total..
74926 data
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:110: Deprecа
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:130: Deprecа
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:152: Deprecа
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:204: Deprecа
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:192: Deprecа
Timer : 04:28:03
Processed 74926 lines data.
saving file...
Saved to => /content/drive/My Drive/TeA/project/datasets/FINAL
Time Start : 17:19:10
Time taken to extracting data : 4:28:17.505043
Time End : 21:47:27
data extracted. DONE.
```

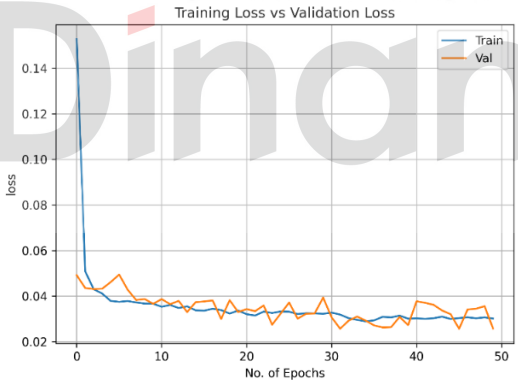
Gambar 4.1 Proses normalisasi data pengujian ke-1 dan lama waktu proses data normalisasi



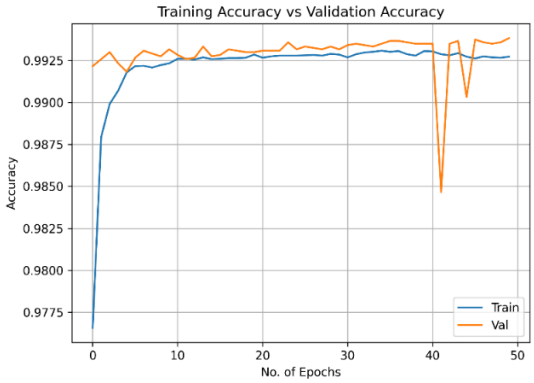
Gambar 4.2 Hasil normalisasi data pengujian ke-1 yang di *plot* menjadi gambar



Gambar 4.3 Grafik Bar Akrivitas Jaringan



Gambar 4.4 Grafik *Loss* proses training dan proses validasi dengan 50 iterasi

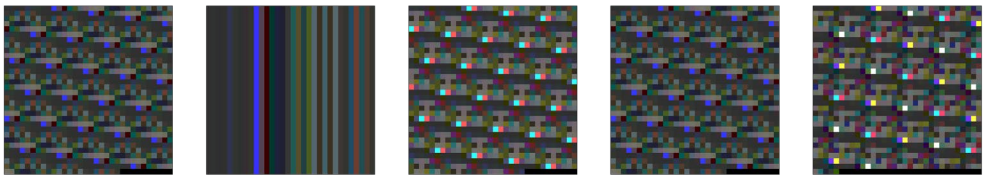


Gambar 4.5 Grafik *Accuracy* proses training dan proses validasi dengan 50 iterasi

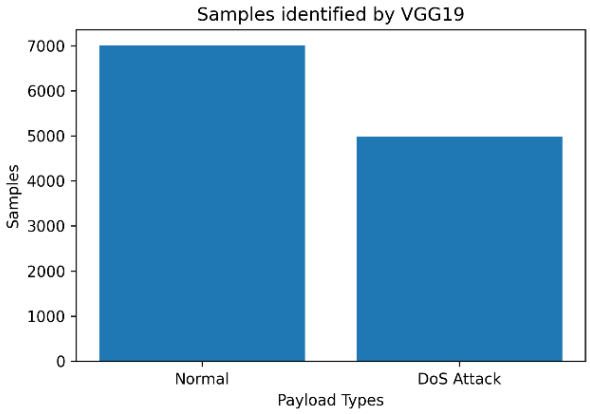
B. Pengujian ke-2

```
Normalization Data
==================
Folder selected : /content/drive/My Drive/TA/dataset
Extracting data : DATA2.csv
counting data total..
59934 data
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:110: Depre
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:130: Depre
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:152: Depre
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:192: Depre
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:204: Depre
Timer : 01:03:28
Processed 59934 lines data.
saving file...
Saved to => /content/drive/My Drive/TA/dataset/datasets/FINAL
Time Start : 17:13:39
Time taken to extracting data : 2:59:56.049021
Time End : 20:13:35
data extracted. DONE.
```

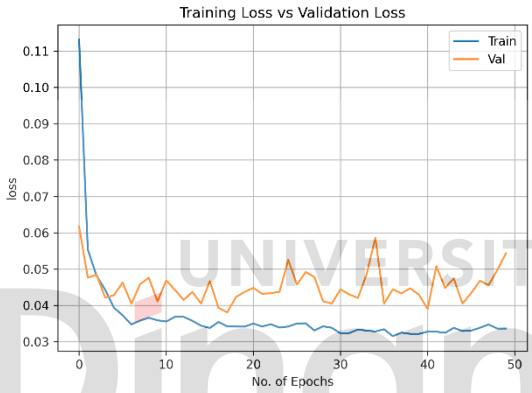
Gambar 4.6 Proses normalisasi data pengujian ke-2 dan lama waktu proses data normalisasi



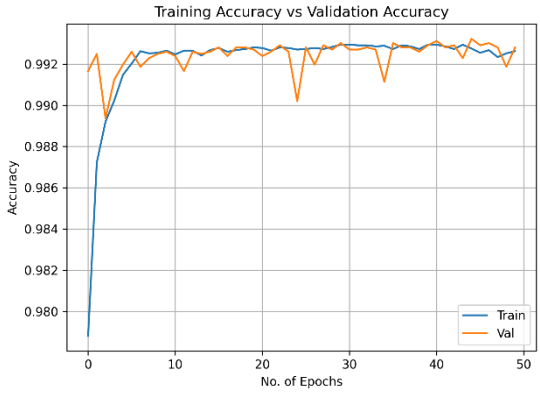
Gambar 4.7 Hasil normalisasi data pengujian ke-2 yang di *plot* menjadi gambar



Gambar 4.8 Grafik Bar Akrivitas Jaringan



Gambar 4.9 Grafik Loss proses training dan proses validasi dengan 50 iterasi

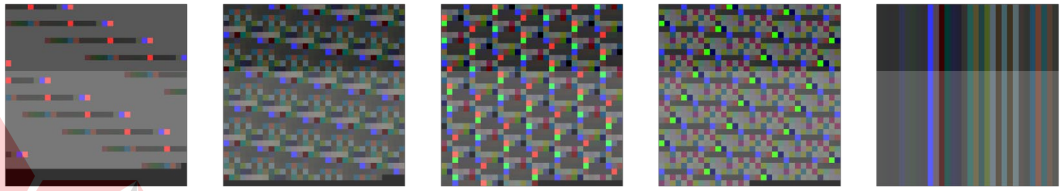


Gambar 4.10 Grafik Accuracy proses training dan proses validasi dengan 50 iterasi

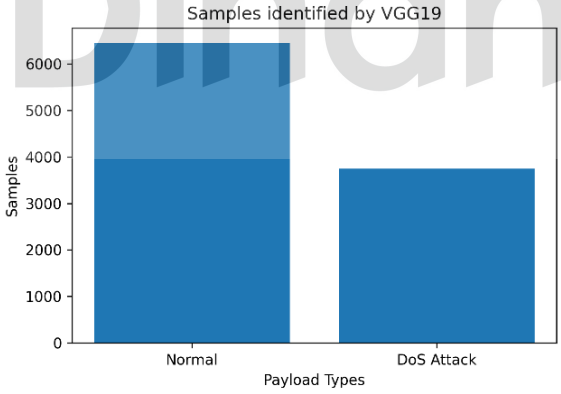
C. Pengujian ke-3

```
Data Normalization
==================
File selected : DATA3.csv
Extracting data : DATA3.csv
counting data total..
50981 data
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:110: DeprecationWarnir
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:130: DeprecationWarnir
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:152: DeprecationWarnir
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:204: DeprecationWarnir
Timer : 00:00:01/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:192: D
Timer : 02:05:55
Processed 50981 lines data.
saving file...
Saved to => /content/drive/My Drive/TeA/project/datasets/FINAL1
Time Start : 01:31:17
Time taken to extracting data : 2:06:00.806678
Time End : 03:37:18
data extracted. DONE.
```

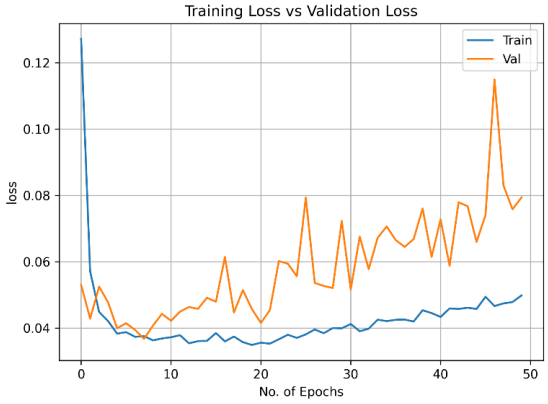
Gambar 4.11 Proses normalisasi data pengujian ke-3 dan lama waktu proses data normalisasi



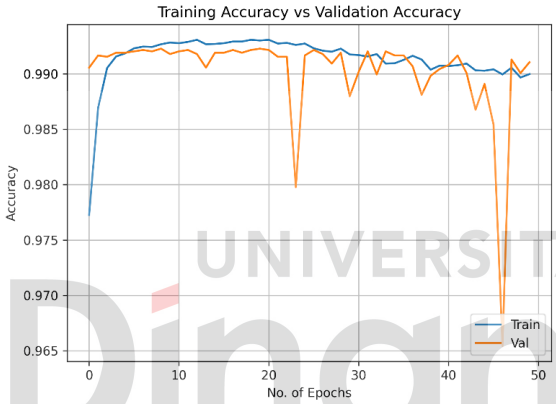
Gambar 4.12 Hasil normalisasi data pengujian ke-3 yang di *plot* menjadi gambar



Gambar 4.13 Grafik Bar Akrivitas Jaringan

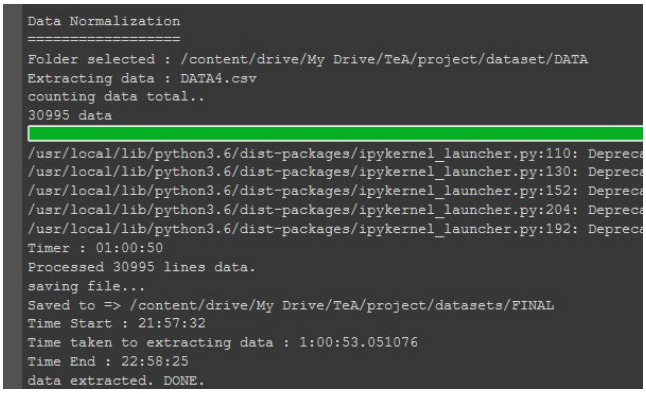


Gambar 4.14 Grafik Loss proses training dan proses validasi dengan 50 iterasi

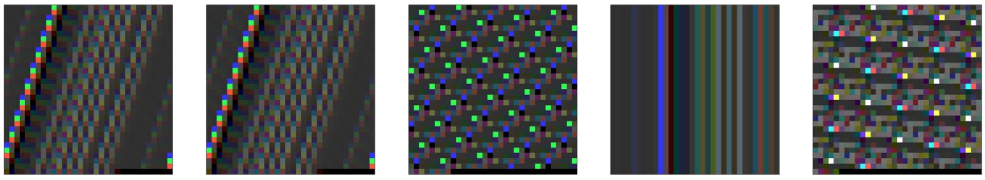


Gambar 4.15 Grafik Accuracy proses training dan proses validasi dengan 50 iterasi

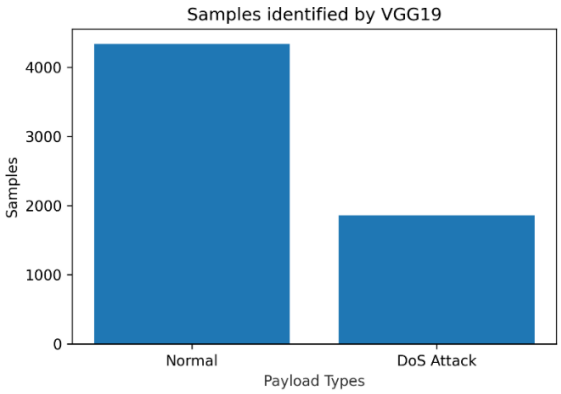
D. Pengujian ke-4



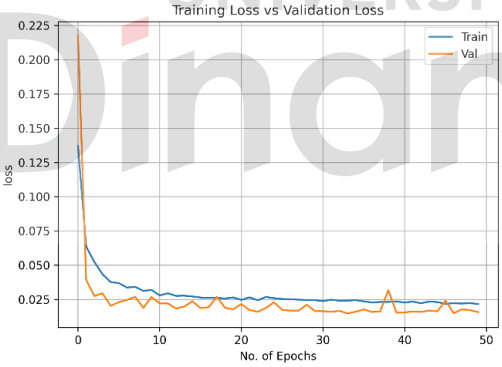
Gambar 4.16 Proses normalisasi data pengujian ke-4 dan lama waktu proses data normalisasi



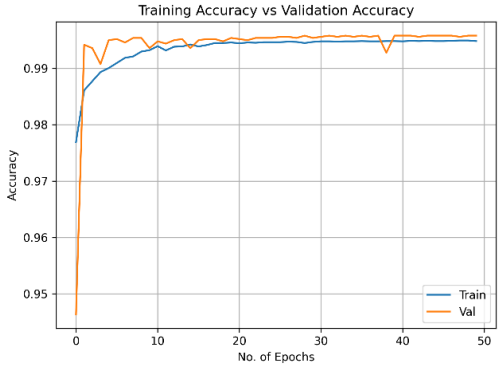
Gambar 4.17 Hasil normalisasi data pengujian ke-4 yang di plot menjadi gambar



Gambar 4.18 Grafik Bar Akrivitas Jaringan

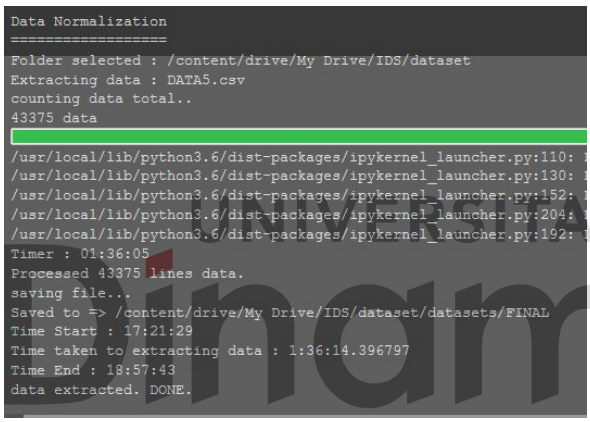


Gambar 4.19 Grafik Loss proses training dan proses validasi dengan 50 iterasi

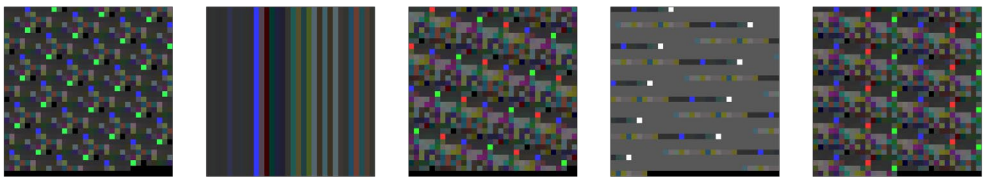


Gambar 4.20 Grafik Accuracy proses training dan proses validasi dengan 50 iterasi

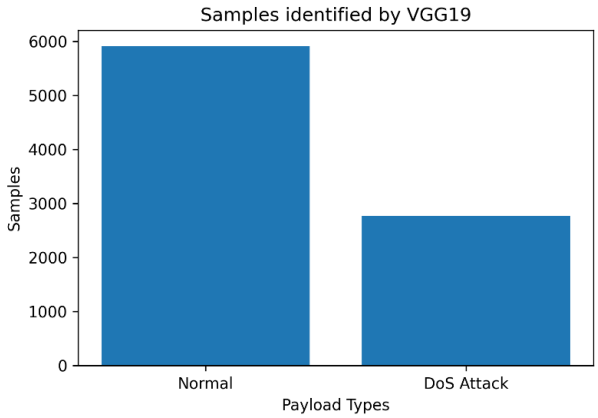
E. Pengujian ke-5



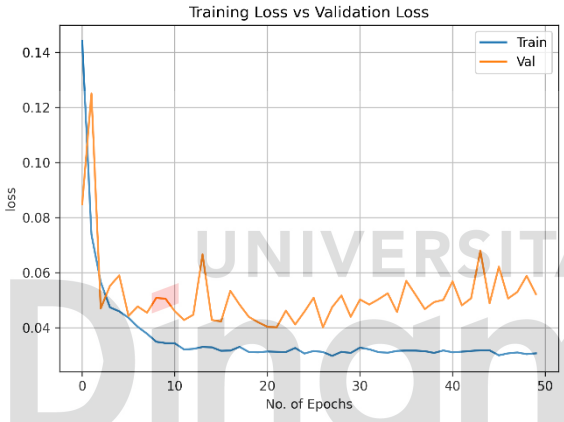
Gambar 4.21 Proses normalisasi data pengujian ke-5 dan lama waktu proses data normalisasi



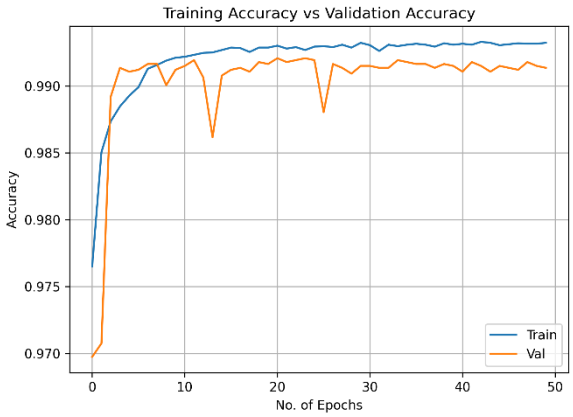
Gambar 4.22 Hasil normalisasi data pengujian ke-5 yang di plot menjadi gambar



Gambar 4.23 Grafik Bar Akrivitas Jaringan



Gambar 4.24 Grafik Loss proses training dan proses validasi dengan 50 iterasi



Gambar 4.25 Grafik Accuracy proses training dan proses validasi dengan 50 iterasi

4.1.5 Analisis Data

Hasil data dari Tabel 4.1, pengujian dilakukan dengan melakukan simulasi jaringan DoS *attack* ke server selama kurang lebih 5 menit sebanyak 5 kali. Dari ke 5 data hasil *log wireshark* dilakukan data normalisasi kemudian data hasil normalisasi dimasukkan kedalam *Deep Learning.* Dari hasil pengujian *Deep Learning* setelah menjalankan ke 5 data didapatkan hasil rata – rata akurasi terhadap variasi data sebesar 99.32% dengan rata – rata *loss* sebesar 4.08%.

4.2 Pengujian Akurasi CNN VGG-19 terhadap Variasi *Iterasi* Proses *Training*

4.2.1 Tujuan

Tujuan dari pengujian ini dilakukan adalah untuk melihat pengaruh akurasi CNN VGG-19 terhadap variasi lama waktu proses *training*. Variasi proses *training* yang dimaksud adalah mengganti jumlah iterasi proses *training* yang dilakukan.

4.2.2 Peralatan yang Digunakan

1. Komputer dengan *port* HTTP atau *port* 80 aktif atau Komputer Server dengan kebutuhan :
   - *Linux Operating System.*
   - *Apache Service* untuk mengaktifkan *port* HTTP atau *port 80.*
2. Komputer dengan *Linux Operating System* dan *Hping3 Linux Tools.*
3. Komputer dengan aplikasi *Wireshark* yang telah di*install.*
4. *Google Colab Account* untuk menjalankan *Deep Learning* dan *Data Normalization.*

4.2.3 Langkah – Langkah dan Cara Pengujian

1. langkah – langkah untuk cara pengujian ini sama seperti pada pengujian pertama sebelumnya.
2. Ketika menjalankan *Deep Learning,* atur pula jumlah iterasi sesuai dengan kebutuhan dimulai dari nilai yang kecil.
3. Analisa hasil *Deep Learning* yang telah dijalankan dan perbedaan hasil dengan jumlah iterasi yang berbeda.

4.2.4 Hasil Pengujian

Hasil pengujian untuk variasi *training* ini dilakukan dengan menggunakan 5 data pengujian yang berbeda – beda dan data yang berbeda juga dengan data dari pengujian sebelumnya. Proses *training* pada pengujian ini dilakukan 10 kali proses *training* dengan variasi iterasi yang berbeda pada setiap datanya, dengan total percobaan pengujian sebanyak 50 kali setiap data di *training* dengan iterasi di antara 10 hingga 100 dengan kelipatan 10 setiap proses *training*nya. Pada tabel 4.2 hingga tabel 4.7 menunjukkan hasil pengujian ke-6 hingga pengujian ke-10. Untuk hasil gambar dan atau hasil berupa grafik lainnya bisa dilihat di bagian lampiran.

Tabel 4.2 Hasil pengujian ke-6 untuk akurasis CNN VGG-19 terhadap variasi *iterasi* proses training

No	Jumlah Total Data	Lama Proses Training (Jumlah Iterasi)	Aktivitas Jaringan		Lama Waktu (detik)	Loss(%)	Akurasi(%)
			Normal (%)	DoS (%)			
1	35738	10	61.05%	38.95%	82	5.76%	99.03%
2		20	61.32%	38.68%	161	5.29%	98.87%
3		30	62.59%	37.41%	241	4.72%	98.78%
4		40	61.64%	38.36%	326	4.68%	98.92%
5		50	60.62%	39.38%	400	4.57%	98.94%
6		60	61.82%	38.18%	484	4.55%	98.99%
7		70	62.60%	37.40%	569	5.39%	98.92%
8		80	60.41%	39.59%	661	6.99%	98.99%
9		90	61.40%	38.60%	789	7.11%	98.85%
10		100	62.19%	37.81%	1056	5.28%	98.92%

Tabel 4.3 Hasil pengujian ke-7 untuk akurasi CNN VGG-19 terhadap variasi iterasi proses training

No	Jumlah Total Data	Lama Proses Training (Jumlah Iterasi)	Aktivitas Jaringan		Lama Waktu (detik)	Loss(%)	Akurasi(%)
			Normal (%)	DoS (%)			
1	70352	10	55.28%	44.72%	211	3.77%	99.41%
2		20	55.77%	44.23%	322	3.17%	99.32%
3		30	55.43%	44.57%	525	3.04%	99.40%
4		40	56.63%	43.37%	631	3.17%	99.45%
5		50	56.02%	43.98%	793	2.11%	99.52%
6		60	56.13%	43.87%	1197	3.56%	99.37%
7		70	55.67%	44.33%	1440	6.12%	99.35%
8		80	55.99%	44.01%	1636	4.60%	99.45%
9		90	55.99%	44.01%	2202	3.60%	99.32%
10		100	55.59%	44.41%	1919	3.60%	99.55%

Tabel 4.4 Hasil pengujian ke-8 untuk akurasi CNN VGG-19 terhadap variasi iterasi proses training

No	Jumlah Total Data	Lama Proses Training (Jumlah Iterasi)	Aktivitas Jaringan		Lama Waktu (detik)	Loss(%)	Akurasi(%)
			Normal (%)	DoS (%)			
1	47285	10	76.17%	23.83%	136	4.51%	99.18%
2		20	75.52%	24.48%	214	4.57%	98.97%
3		30	75.14%	24.86%	320	3.87%	99.35%
4		40	75.59%	24.41%	426	3.70%	99.24%
5		50	75.65%	24.35%	576	3.47%	99.23%
6		60	74.83%	25.17%	755	3.14%	99.41%
7		70	75.66%	24.34%	810	2.99%	99.38%
8		80	75.18%	24.82%	859	4.71%	99.28%
9		90	75.40%	24.60%	963	4.36%	99.28%
10		100	75.33%	24.67%	1247	5.82%	97.96%

Tabel 4.5 Hasil pengujian ke-9 untuk akurasi CNN VGG-19 terhadap variasi iterasi proses training

No	Jumlah Total Data	Lama Proses Training (Jumlah Iterasi)	Aktivitas Jaringan		Lama Waktu (detik)	Loss(%)	Akurasi(%)
			Normal (%)	DoS (%)			
1	44042	10	57.59%	42.41%	117	4.02%	98.97%
2		20	57.17%	42.83%	256	4.23%	99.13%
3		30	57.11%	42.89%	380	4.94%	98.94%
4		40	57.96%	42.04%	472	3.91%	99.04%
5		50	56.71%	43.29%	498	4.59%	98.99%
6		60	57.46%	42.54%	715	4.82%	99.18%
7		70	57.69%	42.31%	707	6.49%	99.14%
8		80	58.60%	41.40%	1035	3.63%	99.10%
9		90	57.42%	42.58%	910	4.21%	99.02%
10		100	57.49%	42.51%	1157	5.52%	98.98%

Tabel 4.6 Hasil pengujian ke-10 untuk akurasi CNN VGG-19 terhadap variasi iterasi proses training

No	Jumlah Total Data	Lama Proses Training (Jumlah Iterasi)	Aktivitas Jaringan		Lama Waktu (detik)	Loss(%)	Akurasi(%)
			Normal (%)	DoS (%)			
1	51291	10	70.49%	29.51%	150	4.87%	99.27%
2		20	70.39%	29.61%	275	4.31%	99.36%
3		30	71.11%	28.89%	436	4.02%	99.40%
4		40	70.07%	29.93%	530	4.74%	99.43%
5		50	70.73%	29.27%	656	3.47%	99.51%
6		60	69.51%	30.49%	791	4.35%	99.33%
7		70	70.40%	29.60%	800	4.85%	99.55%
8		80	70.47%	29.53%	1022	3.64%	99.37%
9		90	70.58%	29.42%	1045	4.61%	99.44%
10		100	70.59%	29.41%	1350	5.78%	98.83%

4.2.5 Analisis Data

Hasil data dari Tabel 4.2 hingga Tabel 4.6, pengujian dilakukan dengan melakukan simulasi jaringan DoS *attack* ke server selama kurang lebih 5 menit sebanyak 5 kali. Dari ke 5 data hasil *log wireshark* dilakukan data normalisasi kemudian data hasil normalisasi dimasukkan kedalam *Deep Learning*. Pada proses *Deep Learning* setiap data di*training* dengan masing – masing data dilakukan 10

kali percobaan dengan jumlah iterasi yang berbeda. Jumlah iterasi yang dipakai mulai dari 10 hingga 100 dengan keliapatan 10. Dari hasil pengujian *Deep Learning* setelah menjalankan ke 5 data dengan setiap data dilakukan 10 kali percobaan *training* dengan jumlah iterasi yang berbeda, didapatkan hasil :

1. Lama waktu proses *training* akan semakin lama semakin bertambahnya jumlah iterasi, namun jumlah data yang dimasukkan akan mempengaruhi lama waktu proses *training*.
2. Akurasi terhadap variasi data dan variasi iterasi proses *training* rata – rata sebesar 99.17%.
3. *Loss* terhadap variasi data dan variasi iterasi proses *training* yang dihasilkan rata – rata sebesar 4.46%.

4.3 Pengujian TPR dan FPR dari Hasil Deteksi Serangan DoS

4.3.1 Tujuan

Tujuan dari pengujian ini dilakukan adalah untuk mengetahui dan melihat hasil ROC *Curve* atau *True Positive Rate* dan *False Positive Rate.* Dengan data – data yang telah diambil dari hasil *log wireshark,* akan dilihat hasil ROC *Curve* dengan *output* yang akan ditampilkan berupa grafik antara *True Positive Rate* dan *False Positive Rate.*

4.3.2 Peralatan yang Digunakan

1. Komputer dengan *port* HTTP atau *port* 80 aktif atau Komputer Server dengan kebutuhan :
   - *Linux Operating System.*
   - *Apache Service* untuk mengaktifkan *port* HTTP atau *port 80.*
2. Komputer dengan *Linux Operating System* dan *Hping3 Linux Tools.*
3. Komputer dengan aplikasi *Wireshark* yang telah di*install.*
4. *Google Colab Account* untuk menjalankan *Deep Learning* dan *Data Normalization.*

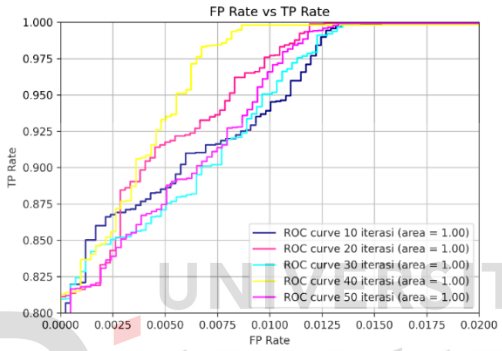
4.3.3 Langkah – Langkah dan Cara Pengujian

1. langkah – langkah untuk cara pengujian ini sama seperti pada pengujian pertama sebelumnya.
2. Analisa hasil *Deep Learning* yang telah dijalankan.

3. Analisa grafik ROC *Curve* hasil program untuk mengetahui *True Positive Rate* dan *False Positive Rate.*

4.3.4 Hasil Pengujian

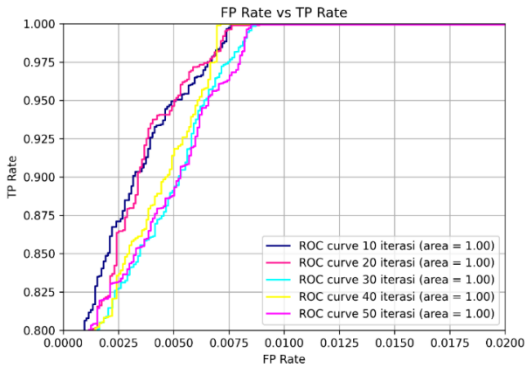
Hasil pengujian untuk variasi *training* ini dilakukan dengan dengan menggunakan 5 data pengujian yang berbeda – beda dan data yang berbeda juga dengan data dari pengujian sebelumnya. Proses *training* pada pengujian ini dilakukan sebanyak 5 kali proses *training* dengan nantinya hasil setiap proses akan dijadikan menjadi 1 grafik ROC *Curve.* Pada Gambar 4.26 hingga Gambar 4.35 menunjukkan hasil pengujian ke-11 hingga pengujian ke-15.



Gambar 4.26 Hasil grafik ROC *Curve* pengujian ke-11

```
Confusion Matrix =
[[7225    0]
 [3034    0]]
```

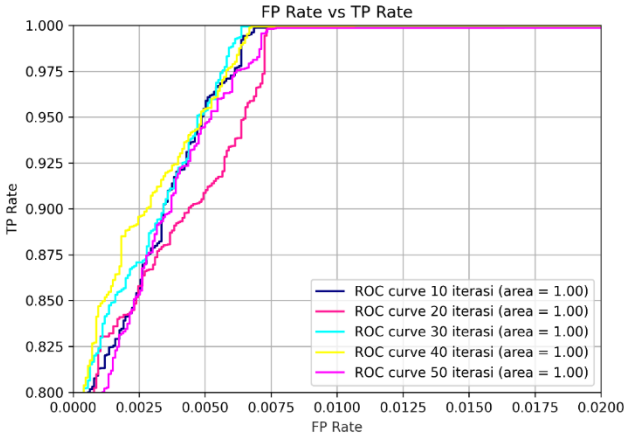
Gambar 4.27 *Confusion Matrix* pengujian ke-11



Gambar 4.28 Hasil grafik ROC *Curve* pengujian ke-12

```
Confusion Matrix =
[[10351     0]
 [ 3839     0]]
```

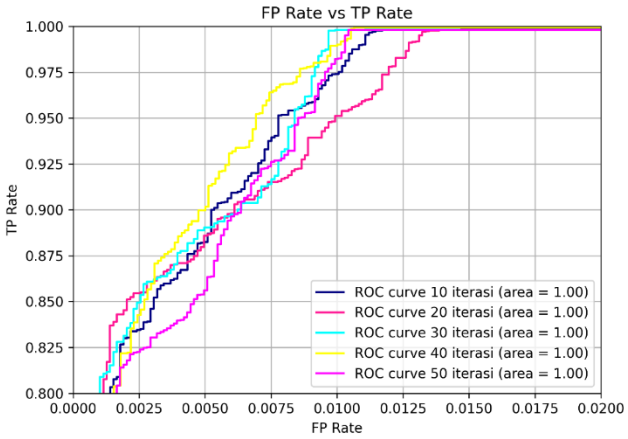
Gambar 4.29 *Confusion Matrix* pengujian ke-12



Gambar 4.30 Hasil grafik ROC *Curve* pengujian ke-13

```
Confusion Matrix =
[[12536     0]
 [ 4201     0]]
```

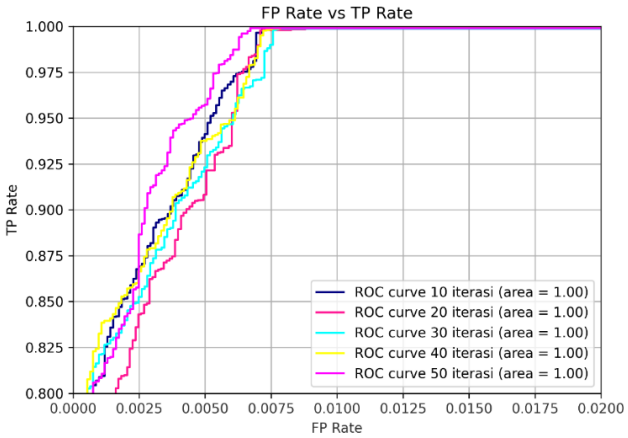
Gambar 4.31 *Confusion Matrix* pengujian ke-13



Gambar 4.32 Hasil grafik ROC *Curve* pengujian ke-14

```
Confusion Matrix =
[[7861    0]
 [3040    0]]
```

Gambar 4.33 *Confusion Matrix* pengujian ke-14



Gambar 4.34 Hasil grafik ROC *Curve* pengujian ke-15

```
Confusion Matrix =
[[9316    0]
 [2987    0]]
```

Gambar 4.35 *Confusion Matrix* pengujian ke-14

4.3.5 Analisis Data

Hasil data dari Gambar 4.26 hingga Gambar 4.35, pengujian dilakukan dengan melakukan simulasi jaringan DoS *attack* ke server selama kurang lebih 5 menit sebanyak 5 kali. Dari ke 5 data hasil *log wireshark* dilakukan data normalisasi kemudian data hasil normalisasi dimasukkan kedalam *Deep Learning*. Proses *training* pada pengujian ini dilakukan sebanyak 5 kali dan hasil data telah disajikan ke dalam bentuk grafik. Dari hasil data grafik, untuk hasil semua data dengan setiap proses *training* yang telah dilakukan didapatkan hasil *ROC Curve* sebesar 1, dimana nilai ini menunjukkan bahwa CNN VGG-19 telah benar memprediksi setiap data dari hasil *training*. Hasil prediksi data yang salah semua akan memiliki nilai *ROC Curve* sebesar 0.0 hingga yang paling tinggi sebesar 1 yang berarti hasil prediksi data benar semua. Hasil grafik yang *coverage* juga dikarenakan CNN VGG-19

hanya melakukan deteksi 2 *class* apakah itu serangan DoS atau bukan serangan DoS dari data hasil *log wireshark.* Selain itu, didapatkan hasil *confusion matrix* dimana *output confusion matrix* ini berupa tabel yang pada bagian kolom adalah kolom prediksi benar dan salah, dan pada bagian baris adalah baris *actual* benar dan salah. Nilai *confusion matrix* pada setiap hasil pengujian ini berhubungan dengan grafik *ROC Curve.* Dengan menggunakan nilai dari *confusion matrix,* dapat dihasilkan nilai – nilai untuk parameter yang lain seperti akurasi, *precision,* TPR, FPR.

BAB V
PENUTUP

Pada bab ini berisi tentang keimpulan dan saran dari hasil pengujian yang telah dilakukan pada Tugas Akhir ini, maka dapat diperoleh beberapa kesimpulan dan saran untuk pengembangan penelitian berikutnya.

5.1 Kesimpulan

Kesimpulan yang diperoleh dari hasil pengujian IDS menggunakan *Deep Learning* untuk deteksi serangan DoS adalah sebagai berikut :

1. Dengan menggunakan data hasil *log wireshark* dan melalui proses data normalisasi implementasi IDS menggunakan *Deep Learning* bisa dilakukan.
2. Hasil IDS menggunakan *Deep Learning* dapat mendeteksi aktivitas jaringan antara serangan Dos dengan bukan serangan DoS ke dalam bentuk grafik bar.
3. Pada pengujian akurasi CNN VGG-19 terhadap variasi data dengan menjalankan 5 data pengujian didapatkan hasil rata – rata akurasi sebesar 99.32% dengan rata – rata *loss* sebesar 4.08%.
4. Pada pengujian akruasi CNN VGG-19 terhadap variasi iterasi proses *training,* dijalankan 5 data pengujian dengan setiap data dilakukan 10 kali percobaan dengan iterasi yang berbeda, kemudian didapatkan hasil semakin bertambah jumlah iterasi waktu yang dibutuhkan untuk melakukan *training* akan semakin lama dan jumlah data yang di*input*kan juga akan mempengaruhi lama waktu proses *training.* Untuk akurasi yang dihasilkan rata – rata sebesar 99.17% dengan *loss* rata – rata sebesar 4.46%.
5. Hasil *True Positive Rate* dan *False Positive Rate* yang didapatkan dengan menjalankan 5 data pengujian yang setiap data dilakukan 5 kali percobaan dengan iterasi yang berbeda kemudian hasil 5 kali percobaan disajikan kedalam satu grafik, dengan nilai *ROC Curve* untuk ke-5 data pengujian sebesar 1, dimana ini menunjukkan bahwa CNN VGG-19 telah benar memprediksi setiap data dari hasil *training.*

5.2 Saran

Adapun saran yang ingin disampaikan adalah :

1. Menambahkan metode serangan yang digunakan sehingga dapat menguji CNN VGG-19 dalam mendeteksi variasi jenis serangan yang ditambahkan.
2. Data yang di*input*kan dapat ditambah lebih banyak lagi, dengan menambah metode serangan dan data yang di*input*kan lebih banyak, dapat dilakukan dianalisa lebih lanjut untuk hasil *loss,* akurasi, FPR, dan TPR.
3. Untuk hasil yang maksimal, implementasi IDS menggunakan *Deep Learning* ini dapat dilakukan di dalam jaringan komputer yang sesungguhnya dengan begitu data yang akan di*input*kan ke dalam CNN VGG-19 sesuai dengan data *real* yang terjadi pada aktivitas jaringan komputer yang terjadi sesungguhnya.

DAFTAR PUSTAKA

Abdi, A. P. (2019). *Bobol Akun e-Banking, Pemilik Kehilangan Uang Rp1 Miliar*. Retrieved from https://tirto.id/bobol-akun-e-banking-pemilik-kehilangan-uang-rp1-miliar-efZZ

Abdillah. (2015). Konsep DoS dan DDoS (Distributed Denial of Service), serta Mekanisme Serangan DoS dan DdoS dan cara penanggulangannya. *STMIK Dipanegara Makassar*.

Arthana, R. (2019, April 5). *Mengenal Accuracy, Precision, Recall dan Specificity serta yang diprioritaskan dalam Machine Learning*. Retrieved from medium.com: https://medium.com/@rey1024/mengenal-accuracy-precission-recall-dan-specificity-serta-yang-diprioritaskan-b79ff4d77de8

Azaim , H. (2020, February). *Mengenal Intrusion Detection System (IDS)*. Retrieved from netsec.id: https://netsec.id/intrusion-detection-system/

Dogru, N., & Masetic, Z. (2017). A Review Of Machine Learning Techniques Efficiency In DoS Attack. *International Journal Of Scientific Research*.

Franedya, R. (2019). *Hati-hati Pakai WiFi Gratisan, HP Bisa Diretas Hacker Jahat*. Retrieved from https://www.cnbcindonesia.com/tech/20190830060632-37-95838/hati-hati-pakai-wifi-gratisan-hp-bisa-diretas-hacker-jahat

Google Developers. (n.d.). *Classification: ROC Curve and AUC*. Retrieved from Machine Learning Crash Course: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Hanák, J. (2016, February 15). *Don't Get Swept Away: The Most Common DDoS Attacks are SYN and UDP Floods*. Retrieved from master data in motion: https://www.masterdc.com/blog/how-ddos-attack-work-types-of-ddos-syn-udp-floods/

Hatta, P. (2017, May 16). *Menilik Activation Function*. Retrieved from medium.com: https://medium.com/@opam22/menilik-activation-functions-7710177a54c9

Koehrsen, W. (2018, March 4). *Beyond Accuracy: Precision and Recall*. Retrieved from towards data science: https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

Michelucci, U. (2019). *Advanced Applied Deep Learning Convolutional Neural Networks and Object Detection.* SSBM Finance Inc.

Mirza, T. (2018, September). *Building an Intrusion Detection System using Deep Learning*. Retrieved from Towards Data Science:

https://towardsdatascience.com/building-an-intrusion-detection-system-using-deep-learning-b9488332b321

Nugroho, D. A., Rochim, A. F., & Widianto, E. D. (2015). Perancangan Dan Implementasi Intrusion Detection System Di Jaringan Universitas Diponegoro. *Jurnal Teknologi dan Sistem Komputer, Vol.3, No.2*.

Primartha, R. (2018). *Belajar Machine Learning Teori Dan Praktik.* Penerbit INFORMATIKA.

Purnama, B. (2019). *Pengantar Machine Learning.* Penerbit INFORMATIKA.

Rozenblum, D. (2001). Understanding Intrusion Detection Systems. *SANS Institute Information Security Reading Room*.

Sena, S. (2017, November 13). *Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN)*. Retrieved from medium.com: https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks For Large-Scale Image Recognition. *Cornell University Library's arXiv.org*.

Sinh-Ngoc Nguyen, Van-Quyet Nguyen, Jintae Choi, & Kyungbaek Kim. (2018). Design and Implementation of Intrusion Detection System using Convolutional Neural Network for DoS Detection. *ICMLSC 2018: Proceedings of the 2nd International Conference on Machine Learning and Soft Computing*.

Suyanto, Ramadhani, K. N., & Mandala, S. (2019). *Deep Learning Modernisasi Machine Learning Untuk Big Data.* Penerbit INFORMATIKA.

Zamani, M., & Movahedi, M. (2013). In *Machine learning techniques for intrusion detection* (pp. 1-9).