TUGAS AKHIR – KM184801

INTRUSION DETECTION MENGGUNAKAN *BACKPROPAGATION NEURAL NETWORK* UNTUK MENCEGAH PELANGGARAN KEAMANAN PADA SISTEM KOMPUTER

KHAFI ANILLAH
06111540000086

Dosen Pembimbing
Prof. Dr. M. Isa Irawan, MT

Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – KM184801

INTRUSION DETECTION MENGGUNAKAN *BACKPROPAGATION NEURAL NETWORK* UNTUK MENCEGAH PELANGGARAN KEAMANAN PADA SISTEM KOMPUTER

KHAFI ANILLAH
NRP 06111540000086

Dosen Pembimbing
Prof. Dr. M. Isa Irawan, MT

DEPARTEMEN MATEMATIKA
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT – KM184801

INTRUSION DETECTION USING BACKPROPAGATION NEURAL NETWORK TO PREVENT SECURITY VIOLATION IN COMPUTER SYSTEMS

KHAFI ANILLAH
NRP 06111540000086

Supervisors
Prof. Dr. M. Isa Irawan, MT

DEPARTMENT OF MATHEMATICS
Faculty Of Science and Data Analytics
Institut Teknologi Sepuluh Nopember
Surabaya 2020

LEMBAR PENGESAHAN

INTRUSION DETECTION MENGGUNAKAN METODE _NEURAL NETWORK_ UNTUK MENCEGAH PELANGGARAN KEAMANAN PADA SISTEM KOMPUTER

INTRUSION DETECTION USING NEURAL NETWORK METHOD TO PREVENT SECURITY VIOLATION IN COMPUTER SYSTEMS

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
untuk Memperoleh Gelar Sarjana Matematika
pada Bidang Studi Ilmu Komputer

Program Studi S-1 Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Spuluh Nopember

Oleh:
KHAFI ANILLAH
06111540000086

Menyetujui
Dosen Pembimbing,

Prof. DR. Mohammad Isa Irawan, MT
NIP. 19631225 198903 1 001

Mengetahui,
Kepala Departemen Matematika
FSAD ITS

Subchan, Ph.D
NIP. 19710513 199702 1 001

INTRUSION DETECTION MENGGUNAKAN *BACKPROPAGATION NEURAL NETWORK* UNTUK MENCEGAH PELANGGARAN KEAMANAN PADA SISTEM KOMPUTER

Nama	**: Khafi Anillah**
NRP	**: 06111540000086**
Departemen	**: Matematika**
Dosen Pembimbing	**: Prof. Dr. Mohammad Isa Irawan, MT**

ABSTRAK

Jaringan komputer saat ini memiliki peran penting dalam berbagai bidang, dari pemerintahan hingga perusahaan pribadi. Seiring fakta tersebut, muncul banyak pelanggar keamanan jaringan komputer yang bertujuan untuk mendapatkan keuntungan pribadi. Oleh karena itu, tiap komputer disarankan memiliki *Intrusion Detection System* (IDS). Namun terdapat masalah fundamental dalam IDS, yakni kuantitas peringatan yang terlalu banyak sehingga merepotkan pengguna, dan performa yang terus menurun karena kesalahan pelaporan dan banyaknya alarm palsu pada kondisi normal. Untuk itu diperlukan sistem yang dapat memeriksa pelanggaran secara efektif dan efisien. *Artificial Neural Networks* (ANN) adalah sebuah metode dari *Machines learning* yang berkerja seperti sistem saraf pada makhluk hidup, dimana sistem ini akan bereaksi pada ransangan. Tujuan dari diciptakannya ANN adalah agar sistem komputer bekerja layaknya otak manusia, namun lebih cepat dan akurat. Sehingga, metode ini dirasa tepat untuk memecahkan masalah ini. Dengan metode ANN, diperoleh IDS dengan akurasi diatas 80%, nilai *recall* dan *precision* diatas 0,5, dan sistem yang mengalami penurunan pada iterasi ke-150.

Kata kunci: *Intrusion Detection System, Artificial Neural Network, Jaringan komputer.*

INTRUSION DETECTION USING BACKPROPAGATION NEURAL NETWORK TO PREVENT SECURITY VIOLATION IN COMPUTER SYSTEMS

Name of Student	***: Khafi Anillah***
NRP	***: 06111540000086***
Department	***: Mathematics***
Supervisor	***: Prof. Dr. Mohammad Isa Irawan, MT***

ABSTRACT

Today's computer networks have an important role in various fields, from government to private companies. Along with this fact, there will be many network security violators that aim to get personal profit from computer networks. Therefore, it is recommended that each computer have an Intrusion Detection System (IDS). However, there are fundamental problems in IDS, namely the quantity of warnings that are too much so that it is troublesome for users, and performance continues to decline due to reporting errors and the number of false alarms under normal conditions. This requires a system that can examine violations effectively and efficiently. Artificial Neural Networks (ANN) is a method of learning machines that work like the nervous system in living things, where the system will react to stimuli. The purpose of the creation of ANN is that computer systems work like the human brain, but are faster and more accurate. Thus, this method is considered appropriate to solve this problem. With the ANN method, IDS is obtained with accuracy above 80%, recall and precision values above 0.5, and the system has decreased at the 150th iteration.

Keywords: *Instrusion Detection System, Artificial Neural Network, Computer network*

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillahirobbil'aalamin, segala puji syukur penulis panjatkan kehadirat Allah SWT, atas segala rahmat, taufiq dan karunia-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul

INTRUSION DETECTION MENGGUNAKAN METODE _NEURAL NETWORK_ UNTUK MENCEGAH PELANGGARAN KEAMANAN PADA SISTEM KOMPUTER

yang merupakan salah satu persyaratan akademis dalam menyelesaikan Program Sarjana Departemen Matematika, Fakultas Matematika, Komputasi, dan Sains Data Institut Teknologi Sepuluh Nopember Surabaya.

Dalam menyelesaikan Tugas Akhir ini, penulis telah banyak mendapat bantuan serta masukan dari berbagai pihak. Oleh karena itu, dalam kesempatan ini penulis menyampaikan terima kasih kepada:

1. Bapak Subchan, Ph.D. selaku Kepala Departemen Matematika FSAD ITS yang telah memberikan dukungan dan motivasi selama perkuliahan hingga terselesainya Tugas Akhir ini.

2. Ibu Dr. Dwi Ratna S, MT. dan Dr. Budi Setiyono, MT selaku Sekretaris I dan sekertaris II Departemen Matematika yang telah memberikan dukungan dan motivasi selama perkuliahan hingga terselesaikannya Tugas Akhir ini.

3. Bapak Prof. Dr. techn. Mohammad Isa Irawan, MT selaku Dosen Pembimbing yang telah memberikan bimbingan, arahan, dan motivasi kepada penulis dalam mengerjakan Tugas Akhir ini sehingga dapat selesai dengan baik.

4. Keluarga tercinta dari Jond's Family yang senantiasa dengan ikhlas memberikan kasih sayang, semangat, doa, dan nasihat-nasihat yang sungguh berarti, serta ibu penulis Sri Suartini yang senatiasa memberikan masukan, semangat dan dukungan kepada penulis.

5. Teman-teman dari Klasikarya, yaitu Levin, Adit, Salman, Itok, yang selalu memberikan semangat dan motivasi untuk menyelesaikan Tugas Akhir ini.

6. Seluruh teman-teman angkatan 2015 dan seluruh keluarga besar Matematika ITS terima kasih atas dukungan dan semangat yang diberikan kepada penulis.

7. Seluruh jajaran dosen dan tendik departemen Matematika ITS yang tidak dapat penulis sebutkan satu-persatu.

Apabila dalam Tugas Akhir ini ada kekurangan, penulis mohon kritik dan saran demi penyempurnaan di masa yang akan datang. Semoga Tugas Akhir ini dapat bermanfaat bagi semua pihak yang berkepentingan.

Wassalamu'alaikum Wr. Wb.

Surabaya, 15 Juli 2020

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Bab ini membahas latar belakang yang mendasari penulisan Tugas Akhir. Mencakup identifikasi permasalahan pada topik Tugas Akhir. Kemudian dirumuskan menjadi permasalahan yang akan diberikan batasan-batasan untuk membatasi pembahasan pada Tugas Akhir ini.

1.1 Latar Belakang

Jaringan komputer adalah jaringan komunikasi digital yang memudahkan setiap komputer untuk berbagi informasi. Jaringan komputer mendukung sejumlah besar aplikasi dan layanan seperti akses ke *World Wide Web*, video digital, audio digital, pemakaian bersama dari aplikasi dan ruang penyimpanan, printer, mesin faks, dan penggunaan email beserta aplikasi pesan lainnya. Pertukaran informasi pada jaringan komputer sendiri membuat jaringan ini butuh perlindungan dari pihak luar. Sehingga diperlukan sistem keamanan komputer.

Sistem keamanan komputer, atau *cybersecurity* adalah perlindungan terhadap sistem komputer dari pencuri atau kerusakan pada perangkat keras, perangkat lunak, atau data elektronik, beserta perlindungan dari gangguan dan kekeliruan dari jaringan [1]. Ranah ini sendiri menjadi sangat penting akibat meningkatnya pemakaian jaringan komputer, seperti internet, dan jaringan nirkabel seperti *Bluetooth* dan *Wi-Fi*, serta dikarenakan oleh pertumbuhan dari perangkat pintar, seperti *smartphone*, televisi, dan semua alat yang berhubungan dengan "*Internet of things"*. Karena kompleksitasnya, baik dalam hal politik maupun teknologi, *cybersecurity* merupakan tantangan terbesar dari dunia kontemporer [2]. Saat ini, banyak bagian

dari sistem sangat rentan terhadap serangan *cyber* yang bisa mengganggu fungsi komputer, merusak data penting, dan memaparkan data pribadi. Sehingga penting untuk membuat sistem kebal terhadap serangan.

Intrusion Detection System (IDS) adalah sebuah "alarm pencuri" dari jaringan keamanan komputer. Tujuannya adalah untuk melindungi komputer dengan menggunakan kombinasi dari alarm yang akan memperingatkan *user* apabila terjadi pelanggaran keamanan [3]. IDS juga berperan dalam mengawasi dan menganalisis pelanggaran pada komputer dan jaringannya. Namun terdapat dua masalah fundamental yang belum dapat dipecahkan pada IDS. Pertama adalah kuantitas alarm. Sebuah IDS bisa mengirimkan 8000 alarm setiap harinya, sehingga setiap pengguna hanya mempunyai waktu 10,8 detik untuk memeriksa alarm. Ini jelas membuat para pengguna kerepotan. Sehingga pengguna komputer cenderung mengabaikannya dan membuat IDS menjadi kehilangan fungsinya. Kedua adalah IDS mengalami penurunan kualitas, ini diakibatkan oleh alarm yang salah deteksi dan mengirimkan terlalu banyak alarm palsu pada kondisi normal [4].

Masalah pada sistem keamanan komputer justru akan menyebabkan rentannya jaringan komputer. Kasus serangan *cyber* sendiri mengalami pelonjakan 300% selama 2019 beberapa diantaranya menyerang perusahaan ternama di dunia. First American pada Mei 2019 mengalami kerugian 885 juta USD akibat menyebarnya informasi pribadi pelanggan. Facebook pada September 2019 mengalami pembobolan password, kerugian dilaporkan mencapai 540 juta USD. Capital One pada Juli 2019 mengakui bahwa

telah terjadi pencurian informasi pribadi, dan terjadi kerugian sebesar 106 juta USD[1].

Usaha preventif ini memerlukan metode yang cepat dan akurat, beberapa metode yang pernah digunakan untuk IDS antara lain *Decision Tree, Bayesian learning, Instance-Based learning, Support vector Machines*, dan beberapa metode lainnya. Di penelitian ini, akan dikembangkan sebuah IDS dengan metode *Neural Network*.

Neural Network, atau *Artificial Neural Network* (ANN), adalah sebuah metode komputasi yang terinspirasi dari jaringan saraf biologis, terutama otak. Sehingga metode ini melakukan kegiatan dengan "belajar" untuk menyelesaikan sebuah masalah tanpa perlu diprogram dengan sangat spesifik. Tujuan awal dari ANN adalah untuk memecahkan masalah dengan cara yang sama seperti otak manusia. ANN sendiri telah digunakan pada banyak hal, seperti citra komputer, pengenalan suara, mesin penerjemah, media sosial, *video Games*, dan diagnosa medis[5].

Menggunakan tujuan dari ANN, pada Tugas Akhir ini dikembangkan sebuah IDS dengan metode *neural networks* untuk mengatasi pelanggaran keamanan pada sistem komputer.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah yang akan dibahas pada proposal Tugas Akhir ini adalah:

1. Bagaimana performa IDS dengan metode *neural Networks* dalam mendeteksi intrusi.

---

[1] Patrick Flesch, Juli 2019. "5 Biggest Cyberattack of 2019 (So Far) and Lessons Learned" Gordon Flesch Company, Inc. https://www.gflesch.com/blog/biggest-cyberattacks-2019

2. Bagaimana performa klasifikasi pelanggaran keamanan yang dilakukan oleh IDS.

1.3 Batasan Masalah

Ruang lingkup permasalahan yang akan dibahas dalam proposal Tugas Akhir ini adalah:

1. Metode yang digunakan dalam Tugas Akhir ini merupakan metode *neural network*.
2. Data yang digunakan dalam Tugas Akhir ini merupakan data yang diperoleh dari Kaggle.
3. Dalam Tugas Akhir ini digunakan jenis-jenis pelanggaran secara acak.
4. Dalam Tugas Akhir ini menggunakan bahasa pemrograman Phyton.

1.4 Tujuan

Adapun tujuan penelitian dari proposal Tugas Akhir ini adalah:

1. Mengetahui performa IDS dengan metode *Neural Networks* dalam mendeteksi intrusi.
2. Mengetahui kemampuan klasifikasi pelanggaran keamanan pada IDS.

1.5 Manfaat

Manfaat dari penelitian Tugas Akhir ini adalah memberikan informasi tentang performa IDS dengan metode *neural network*.

1.6 Sistematika Penulisan

Sistematika penulisan dalam laporan Tugas Akhir ini adalah sebagai berikut:

1. BAB I PENDAHULUAN
   Bab ini menjelaskan latar belakang penyusunan Tugas Akhir, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan laporan Tugas Akhir.
2. BAB II TINJAUAN PUSTAKA
   Pada bab ini diberikan beberapa definisi dan teorema yang digunakan sebagai acuan dalam pembahasan, serta untuk membantu memahami permasalahan yang akan dibahas.
3. BAB III METODOLOGI
   Bab ini menjelaskan tentang tahap-tahap yang dilakukan dalam penyusunan Tugas Akhir ini.
4. BAB IV HASIL DAN PEMBAHASAN
   Bab ini menjelaskan secara detail mengenai hasil penelitian Tugas Akhir ini.
5. BAB V KESIMPULAN DAN SARAN
   Bab ini menjelaskan tentang penarikan kesimpulan yang diperoleh dari pembahasan masalah pada bab sebelumnya serta saran yang diberikan untuk perkembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini diberikan beberapa definisi dan teorema yang digunakan sebagai acuan dalam pembahasan, serta untuk membantu memahami permasalahan yang akan dibahas.

2.1 Penelitian Terdahulu

Dalam praktiknya, penelitian mengenai IDS telah banyak dilakukan. Pada Januari 2018, Bahram Hajimirzaei dan Nima Jafari Navimipour mengajukan sebuah IDS baru berbasis kombinasi jaringan *multilayer perceptron* (MLP), *artificial bee colony* (ABC), dan algoritma *clustering fuzzy*. Lalu lintas jaringan normal dan abnormal akan diidentifikasi oleh MLP, sementara latihan MLP selesai dilakukan oleh algoritma ABC dengan mengoptimalkan nilai dari berat dan bias tautan. *Mean absolut error, root mean square error,* dan statistik kappa digunakan sebagai kriteria evaluasi. Hasil yang diperoleh pada penelitian ini menunjukkan superioritas pada metode kombinasi jaringan [6].

Pada Februari 2018, Cheng-Ru Wang, dkk menggunakan metode *equality constrained-optimization-based* ELM (C-ELM) yang merupakan modifikasi dari ELM dengan menggabungkan metode ini dengan beberapa fitur dari *support vector network*. ELM, atau *extreme machine learning*, adalah satu lapis ANN yang tidak ditujukan untuk dilatih berulang kali namun memiliki kecepatan belajar yang tinggi. Pada penelitian ini metode C-ELM diterapkan pada IDS. Sebuah strategi belajar tambahan digunakan untuk memperoleh nilai optimal dari *neuron* tersembunyi. Hasil berbagai percobaan yang dilakukan dalam penelitian ini menunjukkan bahwa metode ini efektif karena memiliki deteksi serangan yang baik dan kecepatan belajar yang tinggi [7].

Pada Agustus 2018, Alex Shenfield, David Day, dan Aladdin Ayesh melakukan sebuah pendekatan baru untuk mendeteksi lalu lintas jaringan berbahaya dengan ANN yang cocok digunakan pada IDS berbasis *deep pocket inspection*. Penelitian dilakukan menggunakan berbagai macam data (gambar, file *dynamic link library*, catatan, musik, dan dokumen pengolah kata) dan menunjukkan bahwa rancangan ANN mampu membedakan lalu lintas jaringan yang jinak dan berbahaya. Rancangan ANN juga mendapatkan rata-rata akurasi sebesar 98% dan rata-rata salah positif kurang dari 2% pada 10 kali validasi kebenaran. Pendekatan ini dapat membantu meningkatkan kinerja IDS baik pada bidang konvensional maupun *cyber-physical* [8].

2.2 Intrusion Detection System

Intrusion Detection System (IDS) adalah sebuah perangkat lunak yang mengawasi jaringan atau sistem dari aktivitas berbahaya atau pelanggaran aturan. Setiap aktivitas berbahaya atau pelanggaran akan dilaporkan ke pengguna komputer atau dikumpulkan terpusat menggunakan sistem *security informaton and event management* (SIEM). Sistem SIEM mengombinasikan keluaran dari beberapa sumber dan menggunakan metode penyaringan alarm untuk membedakan aktivitas berbahaya dan alarm palsu [9]. Beberapa IDS memiliki kemampuan untuk bertindak pada pelanggaran yang terdeteksi. IDS juga bisa ditugaskan untuk mengklasifikasikan lalu lintas berbahaya [10].

2.2.1 Klasifikasi dari *Intrusion Detection Systems*

Pada umumnya, IDS terbagi menjadi 2 kategori, yaitu:

1. Host-Based IDS
   IDS ini mengevaluasi informasi pada satu atau banyak *host*, termasuk pada sistem operasi, file sistem, dan file aplikasi

2. Network-based IDS
   IDS ini mengevaluasi informasi dari komunikasi jaringan dan menganalisis aliran dari paket yang melintas di jaringan. Paket ditangkap menggunakan beberapa sensor.

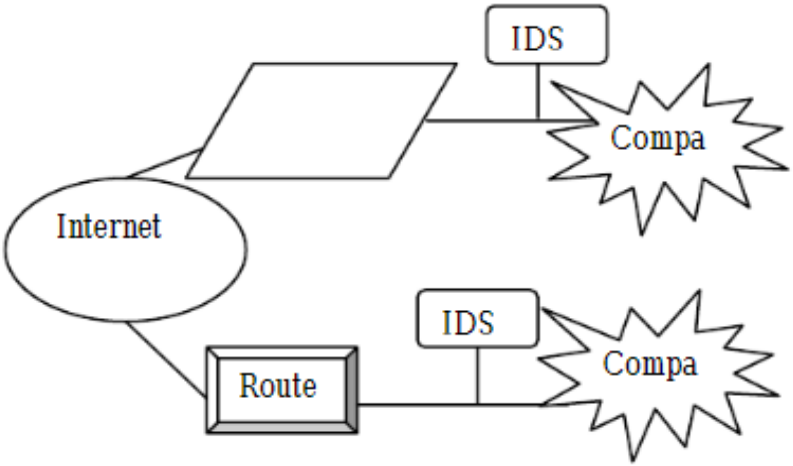
IDS biasanya menggunakan *Anomaly detection* dan *Misuse detection* untuk mendeteksi masalah. *Anomaly detection* menggunakan pemantauan ambang batas untuk menunjukkan kapan metrik yang ditetapkan telah tercapai, biasanya menciptakan basis pengetahuan yang berisi profil kegiatan yang dipantau. *Misuse detection* menggunakan pendekatan berbasis aturan, biasanya menggunakan perbandingan kegiatan pengguna dengan perilaku penyerang yang diketahui menembus system. Mekanisme deteksi intrusi mengidentifikasi potensi serangan jika aktivitas pengguna sesuai dengan aturan yang ditetapkan [11].

2.2.2 *Intrusion Detection* dalam Jaringan Komputer

Jaringan komputer adalah sebuah jaringan komunikasi digital untuk berbagi informasi antara sebuah komputer dengan komputer lainnya. Jaringan komputer mendukung banyak aplikasi dan layanan, seperti *World Wide Web,* video digital, audio digital, dan lain-lain. Jaringan komputer yang paling populer dan paling banyak digunkan saat ini adalah internet.

Dalam jaringan komputer, IDS bisa diletakkan di satu atau lebih tempat, bergantung pada topologi jaringan, jenis aktivitas intrusi, dan kebijakan keamanan. Sebagai contoh, jika pengguna hanya ingin mendeteksi aktivitas intrusi eksternal dan hanya memiliki satu *router* yang terhubung ke internet, maka tempat terbaik untuk meletakkan IDS adalah didalam *router* atau *firewall*. Sedangkan, jika pengguna memiliki banyak jalur untuk ke internet dan pengguna juga ingin mendeteksi ancaman internal, maka IDS bisa diletakkan di setiap bagian jaringan [11].

Peletakan umum IDS pada jaringan komputer dapat dilihat pada gambar dibawah ini.



Gambar 2.1 *Letak umum dari IDS dalam Jaringan Komputer[10]*

2.3 Computer Intrusion

Intrusion dapat diartikan sebagai sebuah tindakan untuk mengganggu dan menyusup ke dalam sebuah sistem. *Computer Intrusion* adalah sebuah keadaan dimana seseorang berusaha untuk mendapatkan akses pada bagian manapun dari sistem komputer. *Intruder* atau *hacker* biasanya menggunakan sebuah program komputer otomatis ketika mereka mencoba menyusup pada sistem komputer.[2]

Ketika berhasil menyusup ke sistem komputer, mereka bisa:

1. Memasuki komputer untuk melihat, mengganti, dan menghapus informasi.
2. Merusak atau memperlambat komputer.

[2] https://www.dmts.biz/faqs-firewalls/what-are-computer-intrusions-or-attacks/

3. Mendapatkan data pribadi pengguna komputer.
4. Menggunakan komputer untuk mengakses komputer lainnya.

2.4 Artificial Neural Network

Artificial Neural Network (ANN) atau *Neural Network* saja adalah sistem komputasi yang terinspirasi dari jaringan syaraf pada makhluk hidup. Sistem ini terdiri dari sejumlah besar *node* yang terhubung satu sama lain dan bekerja secara paralel untuk memecahkan masalah secara spesifik [5]. Sebuah ANN terdiri dari beberapa unit yang terhubung satu sama lain yang disebut *artificial neurons*, yang dibuat berdasarkan neuron pada otak makhluk hidup. Setiap neuron dapat menghantarkan sinyal kepada neuron lainnya.

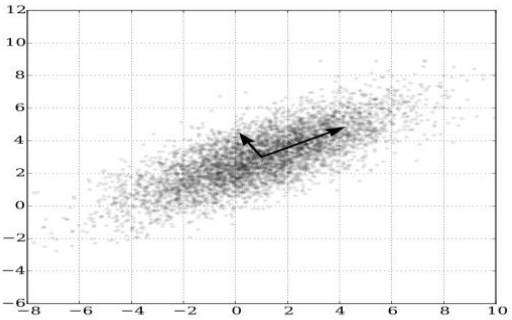
Pada implementasi ANN, sinyal pada tiap hubungan adalah bilangan riil dan keluaran dari tiap neuron akan dihitung oleh fungsi non-linear dari jumlah masukannya. Neuron biasanya memiliki berat yang bertambah seiring ANN belajar. Biasanya, sinyal akan dibedakan menjadi lapisan-lapisan. Setiap lapisan mungkin akan melakukan perubahan pada masukan. Sinyal masuk dari lapisan pertama ke lapisan terakhir setelah melewati lapisan berulang kali [4].

Tujuan dari ANN adalah untuk menyelesaikan masalah dengan cara yang sama seperti otak manusia. Namun, terkadang ANN mengalihkan perhatian untuk menyelesaikan tugas spesifik yang membuatnya berbeda dari otak makhluk hidup. Dalam ilmu matematika, ANN adalah metode yang menggabungkan prinsip biologi dengan statistika lanjutan untuk memecahkan masalah dalam bagian *pattern recognition* dan *game-play*.

2.4.1 *Principal Component Analysis*

Diberikan titik-titik dalam ruang dimensi dua, tiga, atau lebih, maka garis terbaik dapat didefinisikan sebagai garis yang meminimalkan jarak kuadrat rata-rata dari satu titik ke garis. Proses ini dilakukan secara repetitif untuk menghasilkan basis orthogonal dimana dimensi dari data

yang berbeda tidak berkorelasi. Vektor ini disebut *Principal Component* dan merupakan prosedur dari *Principal Component Analysis (PCA)* [12]. PCA didefinisikan sebagai transformasi linear orthogonal yang mengubah data menjadi sistem koordinat baru sehingga varian terbesar data terletak pada koordinat pertama, varian terbesar kedua terdapat pada koordinat kedua, dan seterusnya [13].



Gambar 2.2 *PCA dari Distribusi Gaussian[14]*

Dalam statiska, PCA biasanya digunakan untuk menyederhanakan suatu data, dengan melakukan transformasi secara linier [15]. PCA juga dapat digunakan untuk mereduksi dimensi suatu data tanpa mengurangi karakteristik data tersebut secara signifikan [16]. Dalam ilmu komputer, PCA umumnya digunakan pada dataset yang memiliki dimensi terlalu besar. Pada penelitian ini, PCA digunakan untuk mengurangi dimensi yang terlalu besar, sehingga peluang untuk terjadi *overfitting* dan *underfitting* menjadi kecil. Untuk mengimplimentasikan PCA kepada dataset, ada beberapa langkah yang harus dilakukan [15], yaitu

1. Substraksi Mean

Dimisalkan terdapat matriks X berukuran $n \times p$, dimana n adalah baris yang merepresentasikan repetisi berbeda pada percobaan, sedangkan p merupakan fitur dari dataset. Untuk mendapatkan komponen utama,

maka kolom dari matriks X harus bernilai 0. Apabila kolom matriks X belum bernilai 0, maka matriks X harus dikurangi dengan mean kolom. Misalkan u adalah vektor mean kolom berdimensi $p \times 1$, maka u adalah

$$u_j = \frac{1}{n} \sum_{i=1}^{n} X_{ij}$$

Kemudian vektor u^t akan digunakan untuk mensubstraksi mean dari matriks X. Matriks X yang telah disubstraksi akan di simpan kedalam matriks B dengan persamaan

$$B = X - u^t$$

2. Faktorisasi

Pada PCA salah satu langkah untuk menentukan komponen utama adalah dengan menggunakan dekomposisi nilai singular. Jika B merupakan sebuah matriks riil $n \times p$, maka dekomposisi nilai singular dari B adalah

$$B = U\Sigma V^T$$

Dengan U merupakan matriks orthogonal $n \times n$, V matriks orthogonal $p \times p$, dan Σ adalah matriks diagonal $n \times p$ bernilai riil tak negatif yang disebut nilai-nilai singular.

3. Mendapatkan data baru yang telah direduksi

Komponen utama dari PCA umumnya diambil dari matriks orthogonal hasil dekomposisi nilai singular. Matriks ini nantinya akan di operasikan dengan matriks yang telah direduksi sebelumnya. Misalkan C adalah matriks yang telah diimplementasikan PCA, maka C memiliki persamaan

$$C = X \times U$$

Dimana X adalah matriks $n \times p$ dengan mean kolom bernilai 0, dan U adalah matriks orthogonal $n \times n$ dari dekomposisi nilai singular X.

PCA sendiri memiliki beberapa sifat [16], yaitu

1. Sifat Pertama

Untuk setiap integer q, $1 \leq q \leq p$, dimisalkan transformasi linier orthogonal

$$y = B^t x$$

Dimana y adalah vektor berelemen q dan B^t adalah matriks (qxp), dan misalkan $\Sigma_y = B^t \Sigma B$ adalah matriks varian-kovarian dari y. Maka Σ_y dimaksimalkan dengan $B = A_q$, dimana A_q adalah kolom q pertama dari A.

2. Sifat Kedua

Dimisalkan kembali

$$y = B^t x$$

Maka Σ_y diminimalkan dengan $B = A_q^*$, dimana A_q^* adalah kolom q terakhir dari A

3. Sifat Ketiga

Sifat ketiga juga disebut dengan dekomposisi spektral dari $\sum$, dimana berbentuk

$$\Sigma = \lambda_1 a_1 \alpha_1 + \cdots + \lambda_p a_p \alpha_p$$

Untuk lebih memahami PCA, maka akan diberikan sebuah contoh, misalkan terdapat sebuah dataset dengan 4 fitur dan 5 data di setiap fitur.

Tabel 2.1 *Dataset Contoh*

F1	F2	F3	F4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2

Langkah pertama yang perlu dilakukan adalah melakukan subtraksi mean pada dataset diatas, mulanya dicari mean dan deviasi standar untuk setiap fitur. Mean (μ) dan deviasi standar (σ) untuk setiap fitur disajikan seperti pada **Tabel 2.2** berikut

Tabel 2.2 *Mean dan Deviasi Standar*

	F1	F2	F3	F4
μ	4	3	3	3.4
σ	3	1.59	1.73	2.3

Kemudian mean dan deviasi standar akan di implementasikan kedalam dataset, sehingga dataset akan membentuk data dengan mean bernilai 0 dan deviasi standar bernilai 1. Dataset tersebut disajikan dalam **Tabel 2.3** berikut

Tabel 2.3 *Dataset yang telah disubstraksi*

F1	F2	F3	F4
-1	-0.63	0	0.26
0.33	1.26	1.73	1.56
-1	0.63	-0.58	-0.17
0.33	0	-0.58	-1.04
1.33	-1.26	-0.58	-0.6

Kemudian dapat dicari varian dan kovarian dari dataset. Mencari varian dan kovarian efektif digunakan untuk dataset yang memiliki data sedikit, namun untuk dataset yang besar dekomposisi nilai singular adalah metode yang lebih cocok untuk faktorisasi.

$$var(f1) = \frac{(-1.0-0)^2 + (0.33-0)^2 + (-1.0-0)^2 + (0.33-0)^2 + (1.33-0)^2}{5}$$

$$var\,(f1) = 0.8$$

$$cov(f1, f2) = \frac{\begin{array}{c}(-1.0-0) * (-0.632456-0) + (0.33-0) * (1.264911-0) + (-1.0-0) * \\ (0.632456-0) + (0.33-0) * (0.000000-0) + \\ (1.33-0) * (-1.264911-0)\end{array}}{5}$$

$$cov(f1, f2) = -0.25298$$

Dengan melakukan langkah yang sama untuk fitur lainnya, maka dataset yang berisi varian dank ovarian adalah seperti pada **Tabel 2.4** berikut

Tabel 2.4 *Dataset Kovarian*

	F1	F2	F3	F4
F1	0.8	-0.25	0.04	-0.14
F2	-0.25	0.8	0.51	0.49
F3	0.04	0.51	0.8	0.75
F4	-0.14	0.49	0.75	0.8

Kemudian dapat dicari nilai eigen dan vektor eigen dari dataset tersebut, **Tabel 2.4** akan berubah menjadi **Tabel 2.5** seperti berikut

Tabel 2.5 *A- λI = 0*

	F1	F2	F3	F4
F1	0.8 -λ	-0.25	0.04	-0.14
F2	-0.25	0.8 -λ	0.51	0.49
F3	0.04	0.51	0.8 –λ	0.75
F4	-0.14	0.49	0.75	0.8 -λ

Dari tabel diatas ditemukan bahwa $\lambda = 2.52,\ 1.07,\ 0.39,\ 0.03$. Kemudian **Tabel 2.5** akan dikalikan dengan matriks $\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$ sehingga menghasilkan vector eigen seperti pada **Tabel 2.6** berikut

Tabel 2.6 *Vektor Eigen*

E1	E2	E3	E4
0.16	-0.92	-0.31	0.2
-0.52	0.20	-0.82	0.12
-0.59	-0.32	0.19	-0.72

-0.6	-0.12	0.45	0.65

Setelah diperoleh vektor eigen, maka akan dicari dataset yang ditransformasikan dengan PCA, dataset ini diperoleh dari mengalikan dataset yang telah disubstraksi dengan sampel vektor eigen, pada contoh ini vektor eigen yang diambil adalah fitur E1 dan E2. Sehingga dataset yang telah ditransformasikan disajikan dalam **Tabel 2.7** sebagai berikut

Tabel 2.7 *Dataset yang telah ditransformasikan*

Nf1	Nf2
0.01	0.76
-2.56	-0.78
-0.05	1.25
1.01	0.002
1.58	-1.22

Dataset tersebut adalah dataset yang telah diimplementasikan PCA

2.4.2 *Multilayer Perceptron* dan *Backpropagation*

Multilayer Perceptron (MLP) adalah bagian dari *feedforward* ANN [17]. Istilah ini tidak merujuk pada satu *perceptron* yang memiliki banyak lapisan, tetapi MLP berisi banyak *perceptron* yang disusun dari beberapa lapisan. *Perceptron* dari MLP bisa menggunakan sebarang fungsi aktivasi. Sebuah *perceptron* dapat melakukan klasifikasi biner dan bebas untuk melakukan klasifikasi atau regresi tergantung pada fungsi aktivasi. MLP setidaknya terdiri tiga lapisan node, yaitu lapisan *input*, lapisan *hidden*, dan lapisan *output*. Semua lapisan menggunakan fungsi aktivasi nonlinear, kecuali lapisan *input*. MLP menggunakan metode *supervised learning* yang disebut *backpropagation* untuk melakukan *train* [18].

MLP pada umumnya digunakan dalam penilitian karena memiliki kemampuan untuk memecahkan masalah rumit secara stokastik. MLP saat

ini banyak diterpakan diberbagai bidang, seperti pengenalan suara, pengenalan gambar, dan *software* penerjemah. Umumnya MLP menggunakan fungsi aktivasi di setiap layer, dimana saat ini fungsi aktivasi paling umum dari MLP adalah fungsi aktivasi ReLU yang memiliki bentuk

$$f(x) = x^{+} = \max(0, x) \tag{2.1}$$

Dimana x adalah input terhadap neuron.

MLP merupakan bagian dari *backpropagation neural network*, yang menjadi algoritma untuk *feedforward* pada *supervised learning*. *Backpropagation* merupakan algoritma yang menghitung gradient pada ruang bobot dengan mempertimbangkan *loss function* [17]. Umumnya *backpropagation* memiliki persamaan berupa kombinasi dari fungsi komposisi dan multiplikasi matriks

$$g(x) \coloneqq f^{L}(W^{L} f^{L-1}(W^{L-1} \dots f^{1}(W^{1} x) \dots))$$

Dimana x merupakan input, L merupakan jumlah layer, W adalah bobot, dan f adalah fungsi aktivasi.

Untuk setiap dataset *train* maka dimiliki himpunan pasangan *input-output* $\{(x_i, y_i)\}$. Untuk setiap pasangan *input-output* (x_i, y_i), maka terdapat *loss function* yang merupakan selisih dari *predicted output* $g(x_i)$ dan *target output* y_i

$$C(y_i, g(x_i))$$

Backpropagation menghitung tiap gradient secara efisien dengan menghindari perhitungan berganda dan tidak menghitung nilai yang tidak dibutuhkan, dengan menghitung gradient dari tiap layer, dari belakang ke depan [18]. Algoritma dari *backpropagation* sendiri adalah sebagai berikut [19]:

Langkah 0 Inisialisasi Bobot
(Menetapkan bobot pada nilai acak yang kecil)

Langkah 1 Ketika belum mencapai *stop condition* lakukan langkah 2-9

Langkah 2 Untuk setiap pasangan *training*, lakukan langkah 3-8.

F*eedforward:*

Langkah 3 Setiap input $(X_i, i = 1,..,n)$ menerima sinyal *input* x_i dan memancarkan sinyal kepada semua unit di *hidden layer*

Langkah 4 Setiap *hidden unit* $(Z_j, j = 1, \dots, p)$ menghitung bobotnya,

$$z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i$$

menerapkan fungsi aktivasi untuk menghitung sinyal *output*,

$$z_j = f(z_{in_j})$$

dan mengirimkan sinyal ke setiap unit di *output layer*

Langkah 5 Setiap *output unit* $(Y_k, k = 1, \dots, m)$ menghitung bobotnya

$$y_{in_k} = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

dan menerapkan fungsi aktivasi untuk menghitung sinyal *output*,

$$y_k = f(y_{in_k})$$

Backpropagation dari *error*:

Langkah 6 Setiap unit *output* $(Y_k, k = 1, \dots, m)$ menerima sebuah pola tujuan yang berhubungan dengan pola *train*, kemudian unit menghitung syarat informasi *error*,

$$\delta_k = (t_k - y_k) f'(y_{in_k}),$$

menghitung syarat koreksi bobot,

$$\Delta w_{jk} = \alpha \delta_k z_j$$

menghitung syarat koreksi bias,

$$\Delta w_{0k} = \alpha \delta_k$$

dan mengirim δ_k ke *layer* selanjutnya

Langkah 7 Setiap *hidden unit* $(Z_j, j = 1, \dots, p)$ menjumlahkan *input delta*,

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk}$$

kemudian dikalikan dengan fungsi aktivasi nya untuk menghitung syarat informasi *error*

$$\delta_j = \delta_{in_j} f'(z_{in_j})$$

menghitung syarat koreksi bobot

$$\Delta v_{ij} = \alpha \delta_j x_i$$

dan menghintung syarat koreksi bias

$$\Delta v_{0j} = \alpha \delta_j$$

Langkah 8 Setiap unit *output* $(Y_k, k = 1, \dots, m)$ memperbarui bobot dan biasnya $(j = 0, \dots, p)$

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$$

Dan setiap *hidden unit* $(Z_j, j = 1, \dots, p)$ memperbarui bobot dan biasnya $(i = 0, \dots, n)$

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$$

Langkah 9 *Stopping Condition*

BAB III
METODOLOGI PENELITIAN

Bab ini menjelaskan langkah-langkah yang digunakan dalam penyelesaian masalah pada tugas akhir ini. Dijelaskan pula prosedur dan proses pelaksanaan tiap-tiap langkah yang dilakukan dalam menyelesaikan Tugas Akhir.

3.1. Langkah Pengerjaan

Langkah-langkah yang digunakan dalam menyelesaikan permasalahan pada penelitian ini adalah sebagai berikut:

1. Studi Literatur

Tahap ini merupakan tahap untuk melakukan identifikasi permasalahan, yaitu mencari referensi-referensi yang menunjang penelitian dalam Tugas Akhir ini. Referensi ini bisa berupa buku, jurnal, tugas akhir maupun artikel terkait *Intrusion Detection* dan *Neural Networks*, diantaranya yaitu jurnal yang disusun oleh E. Kesavulu Reddy yang menjelaskan penerapan *neural networks*, kemudian jurnal yang disusun oleh Asma Abbas Hassan dkk. yang membahas tentang IDS dan penggunaan data set sebagai simulator pelanggaran keamanan pada komputer.

2. *Preprocessing* Data

Pada tahap ini dilakukan *preprocessing* data dengan mengubah semua fitur pada dataset menjadi biner. Ini berguna agar data menjadi lebih mudah dimasukkan ke sistem dan menghindari adanya kerusakan pada *hardware* akibat dataset yang terlalu berat. Langkah *preprocessing* adalah sebagai berikut:

1. Mengidentifikasi fitur kategorikal

   Pada tahap ini akan dilakukan identifikasi pada fitur data yang masih belum bersifat numerik, dan melihat apakah

fitur data tersebut sudah terdistribusi merata, dan melihat apakah perlu dibuat *dummy* untuk fitur tersebut

2. Memasukkan fitur kategorikal ke array 2D
   Proses ini dilakukan untuk memudahkan perubahan data dari kategorikal menjadi numerik. Fitur data yang pada tahap 1 masih berupa fitur kategorikal, akan dimasukkan kedalam array 2D
3. Membuat kolom untuk *dummy*
   Proses ini dilakukan agar fitur kategorikal yang sebelumnya tidak bersifat numerik, dapat dengan mudah diubah menjadi numerik.
4. Mengubah fitur kategorikal menjadi numerik
   Pada tahap ini semua fitur kategorikal pada data *train* dan *test* akan diubah menjadi numerik menggunakan *LabelEncoder*().
5. Menggunakan *One-Hot-Encoding*
   One-Hot-Encoding digunakan untuk mengubah fitur kategorikal menjadi biner. *One-Hot-Encoding* hanya bisa dilakukan apabila *input* memiliki nilai *integer*, yang memnunjukkan nilai yang diambil dari fitur kategorikal, dengan kisaran nilai [0, n]. Pada tahap ini semua fitur data yang telah diubah pada tahap 4, akan di-*input* kedalam *One-Hot-Encoding*.
6. Membuat dataset baru
   Pada tahap ini, fitur yang hilang akibat *preprocessing* akan ditambahkan, kemudian semua data yang telah diubah akan digabungkan, dan semua kolom label serangan akan diganti, dengan 0 sebagai normal atau benar positif, dan 1 sebagai serangan atau salah negatif. Kemudian dataset baru akan disimpan dengan format .mat sebagai format dasar data

berbentuk biner, dan untuk mempemudah pemanggilan dataset dalam system.

3. Pembuatan Sistem

Pada tahap ini dilakukan pembuatan sistem untuk menyimulasikan IDS dengan ANN pada jaringan komputer berdasarkan referensi-referensi yang telah didapatkan pada saat studi literatur. Sistem yang akan dibuat meninjau data set yang didapatkan. Dengan langkah-langkah sebagai berikut:

1. Meng-*input* data dan mengimplementasikan PCA

   Pada tahap ini *dataset* yang telah di *preprocessing* akan di-input, dan diberikan fitur baru untuk mempermudah sistem. *Dataset* kemudian akan melalui proses PCA untuk melakukan reduksi dimensi sehingga memungkinkan sistem untuk berjalan lebih cepat. Proses PCA akan melakukan substraksi nilai mean terhadap dataset, sehingga memberikan sebuah nilai x yang telah direduksi, dimana x adalah banyaknya input dari dataset. Dari substraksi tersebut akan dibentuk sebuah matriks σ untuk mendapatkan nilai komponen utama pada PCA, matriks σ diperoleh dengan persamaan berikut,

   $$\sigma = \frac{[x\ yang\ telah\ direduksi]^t \times [x\ yang\ telah\ direduksi]}{panjang\ x\ yang\ telah\ direduksi} \quad (3.1)$$

   Dikarenakan telah diperoleh nilai matriks σ, maka akan dilakukan dekomposisi nilai singular untuk memperoleh komponen utama dari PCA

   $$\sigma = u\ \times diag(s) \times\ v^t \quad (3.2)$$

   Dan kemudian matriks diagonal s akan digunakan untuk memperoleh *score* pada PCA, dengan persamaan sebagai berikut

   $$Score = \frac{\sum_k s}{\sum s} \quad (3.3)$$

Dimana $\sum_k s$ merupakan jumlah dari matriks s sampai pada indeks k, sedangkan $\sum s$ merupakan jumlah dari keseluruhan matriks s.

Kemudian didapatkan sebuah nilai dari dataset train yang telah di reduksi. Selanjutnya juga dilakukan inisiasi nilai akurasi, dengan akurasi merupakan

$$acc = \frac{correct\ data}{length\ of\ predicted\ data} \times 100 \quad (3.4)$$

2. Mengimplementasikan MLP

   Pada tahap ini akan diimplementasikan MLP menggunakan fungsi aktivasi ReLU seperti pada persamaan (2.1) yang saat ini merupakan fungsi aktivasi dasar bagi MLP untuk menciptakan kesederhanaan pada komputasi. Namun pada layer terakhir akan digunakan fungsi softmax untuk memetakan output yang belum dinormalisasi. Fungsi softmax yang umum adalah

   $$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} untuk\ i = 1, \dots, K \text{ dan } z = (z_1, \dots, z_K) \in \mathbb{R}^k \quad (3.5)$$

3. Membuat model untuk *train* dan *test*

   Pada tahap ini akan dibangun kelas yang mengolah data *train* untuk membantu system dalam melakukan *test*. Pada kelas ini akan diberikan konfigurasi model dan *train*, kemudian diberikan optimisasi pada system dengan memberikan loss function sebagai pengevaluasi kemampuan system dan dengan memberikan algoritma optimisasi ADAM pada system. ADAM adalah *Adaptive Moment Estimation*, optimisasi ini menghitung *individual learning rate* untuk setiap parameter yang berbeda. ADAM menggunakan momen dari gradien untuk menyesuaikan *learning rate* pada setiap bobot. Sehingga, dapat diberikan persamaan dari ADAM adalah

   $$m_n = E[X^n] \quad (3.6)$$

Dimana m_n adalah momen ke-n dari variabel acak, dan $E[X^n]$ adalah nilai harapan dari pangkat ke-n dari variabel tersebut.

Kemudian system diberikan *verbose* untuk mencegah sistem menjadi lambat dan melakukan proses yang tidak perlu. Kemudian dilakukan prediksi, dan sistem akan mencatat hasil dari *train* dan *test*.

4. Simulasi

Pada tahap ini sistem disimulasikan secara numerik menggunakan Python, untuk melihat kemampuan system dalam bekerja. Pada tahap ini akan dilihat waktu yang diperlukan system untuk menyelesaikan proses *train*, melihat akurasi dari system, dan melihat *precision, recall, f1-score,* dan *support* pada bagian salah positif dan benar positif.

5. Analisis Hasil Simulasi

Pada tahap ini dilakukan analisis terhadap hasil simulasi sistem yang dilakukan pada tahap 4. Hasil analisis ini akan menjelaskan secara detail terkait kemampuan sistem dalam mendeteksi intrusi, dan kemampuan sistem dalam mengklasifikasikan data.

6. Penarikan Kesimpulan dan Saran

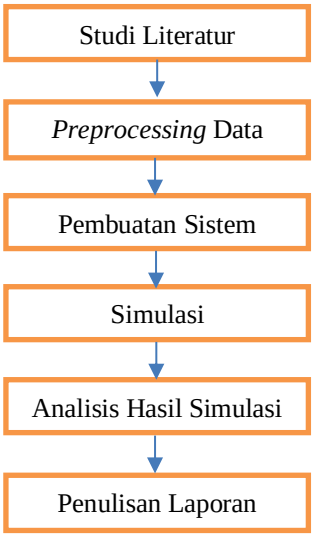
Pada tahap ini, akan ditarik kesimpulan dari penelitian yang telah dikerjakan. Serta memberi sebuah saran untuk penelitian ini dapat dikembangkan menjadi lebih baik lagi.

7. Penulisan Laporan

Pada tahap ini, akan disusun sebuah laporan dari hasil penelitian yang telah dikerjakan secara runtut dan teratur.

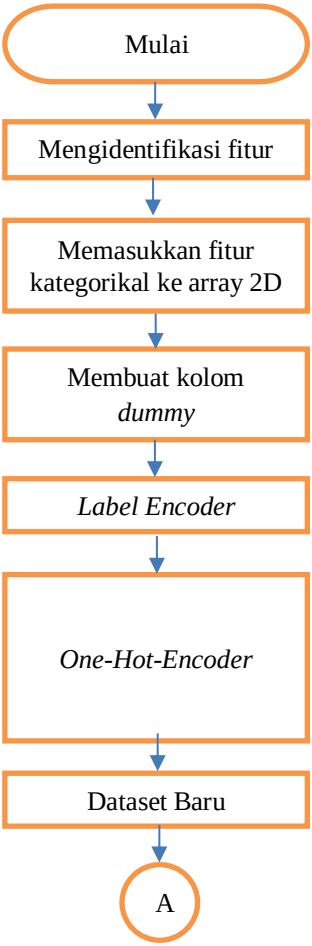
3.2. Diagram Alur Penelitian

Alur penelitian yang dilakukan dalam Tugas Akhir ini disajikan dalam **Gambar 3.1** berikut ini,

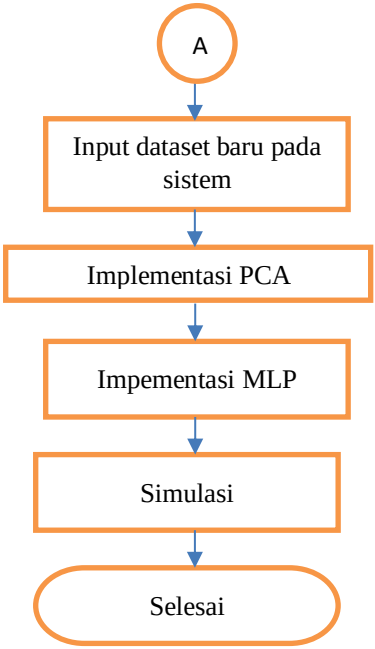


***Gambar 3.1** Diagram Alur Penelitian*

Alur pembuatan sistem akan dijelaskan dalam **Gambar 3.2** berikut ini



Gambar 3.2a *Diagram Alur Sistem*



Gambar 3.2b *Diagram Alir Sistem*

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai hasil dan pembahasan mengenai penilitian secara rinci.

4.1 Analisis Data

Dataset pada penelitian kali ini menggunakan "NSL-KDD" dataset, sebuah dataset standar untuk Intrusion Detection. NSL-KDD adalah dataset yang bersifat memperbaharui dataset KDD-99, sehingga dataset NSL-KDD memiliki fitur dan masalah yang sama, namun dataset ini dirasa merupakan dataset yang efektif untuk melakukan percobaan Intrusion Detection. Selain itu, jumlah catatan dalam NSL-KDD membuatnya bisa menjalankan percobaan pada dataset lengkap tanpa perlu membuat sampel acak kecil. Akibatnya, hasil evaluasi dari pekerjaan penelitian yang berbeda akan konsisten dan dapat dibandingkan.

Dataset ini berisi 41 fitur dari tiap koneksi yang mengidentifikasi tiap status dari koneksi sebagai normal atau tidak [20]. 41 fitur dataset dijelaskan dalam **Tabel 4.1**

Tabel 4.1 Fitur Dataset

No	Nama fitur	Deskripsi
1	Duration	Lama koneksi
2	Proptocol_type	Tipe dari protokol
3	Service	Jaringan tujuan
4	Flag	Status dari koneksi
5	Src_bytes	Jumlah data bytes dari sumber ke tujuan

No	Nama Fitur	Deskripsi
6	Dst_bytes	Jumlah data bytes dari tujuan ke sumber
7	Land	1 jika koneksi berasal dari port yang sama: lainnya 0
8	Wrong_fragment	Jumlah dari bagian yang “salah”
9	Urgent	Jumlah dari paket yang penting
10	Hot	Jumlah dari indikator “hot”
11	Num_failed_logins	Jumlah dari percobaan login gagal
12	Logged_in	1 jika login sukses: lainnya 0
13	Num_compromised	Jumlah kondisi yang “compromised”
14	Root_shell	1 jika root shell didapatkan, lainnya 0
15	Su_attempted	1 jika perintah “su root” dilakukan, lainnya 0
16	Num_root	Jumlah dari akses “root”
17	Num_file_creations	Jumlah dari operasi pembuatan file
18	Num_shells	Jumlah dari shell
19	Num_acces_files	Jumlah dari operasi pada file akses control
20	Num_outbound_cmds	Jumlah dari perintah outbound pada sesi ftp

No	Nama Fitur	Deskripsi
21	Is_hot_logins	1 jika login bias dimasukkan ke daftar "hot", lainnya 0
22	Is_guest_logins	1 jika yang login adalah "guest", lainnya 0
23	Count	Jumlah dari koneksi ke host yang sama pada arus yang sama
24	Srv_count	Jumlah dari layanan yang sama pada koneksi pada periode 2 detik pertama
25	Serror_rate	Presentase dari koneksi memiliki "SYN" error
26	Srv_serror_rate	Presentase dari koneksi memiliki "SYN" error
27	Rerror	Presentase dari koneksi memiliki "REJ" error
28	Srv_rerror_rate	Presentase dari koneksi memiliki "REJ" error
29	Same_srv_rate	Presentase dari koneksi ke layanan yang mirip
30	Diff_srv_rate	Presentase dari koneksi ke layanan yang tidak mirip
31	Srv_diff_host_rate	Presentase koneksi ke host yang tidak mirip
32	Dst_host_count	Jumlah dari host tujuan
33	Dst_host_srv_count	srv_count dari host tujuan
34	Dst_host_same_srv_rate	Same_srv_rate dari host tujuan

No	Nama Fitur	Deskripsi
35	Dst_host_diff_srv_rate	Diff_srv_rate dari host tujuan
36	Dst_host_same_src_port_rate	Same_src_port_rate dari host tujuan
37	Dst_host_srv_diff_host_rate	Diff_host_rate dari host tujuan
38	Dst_host_serror_rate	Serror_rate dari host tujuan
39	Dst_host_serror_rate	Srv_serror rate dari host tujuan
40	Dst_host_rerror_rate	Rerror_rate dari host tujuan
41	Dst_host_srv_rerror_rate	Srv_serror_rate dari host tujuan

Dari hasil pennjabaran fitur pada **Tabel 4.1**, fitur-fitur tersebut dapat dibagi menjadi 4 berdasarkan jenis inputnya, yaitu:

1. Kategorikal
2. Biner
3. Diskrit
4. Kontinu

4.2 *Preprocessing* Data

Pada tahap ini dilakukan *preprocessing* untuk mengubah dataset yang berbentuk numerical dan kategorikal menjadi biner, proses ini dibutuhkan agar sistem nantinya dapat menggunakan metode PCA dengan lebih efektif. Berikut beberapa tahap dari *Preprocessing* data

4.2.1 Version Check

Langkah ini merupakan inisiasi untuk melihat kemampuan *software* yang diperlukan unutk melakukan *preprocessing* dan pembuatan sistem, beberapa hal yang perlu dilihat adalah modul seperti *pandas* dan *numpy*, modul *pandas* adalah modul yang berguna untuk melakukan manipulasi data, sedangkan modul *numpy* adalah modul

yang berguna untuk melakukan perhitungan *scientific*. Kemudian juga akan dilihat versi dari python dan sklearn.

4.2.2 Mengidentifikasi Dataset

Pada langkah ini dataset akan mulai di *input*, karena dataset memiliki format .csv, maka langkah pertama adalah memasukkan nama kolom untuk memisahkan setiap fitur yang ada dalam dataset, kemudian dilakukan pengecekan pada dimensi dataset, berapa input yang dimiliki pada tiap dataset, dan berapa fitur yang dimiliki. Dari langkah ini diperoleh bahwa dataset *train* memiliki 125.973 data dengan 42 fitur, dan dataset *test* memiliki 22.544 data dengan 42 fitur

Selanjutnya akan dilakukan pendataan, fitur mana saja yang merupakan fitur kategorikal, fitur kategorikal adalah fitur yang memiliki isi non-numerik, contoh seperti fitur protocol_type yang berisi icmp, tcp, dan ump, pendataan ini dilakukan untuk dataset train dan test, pendataan ini dilakukan unutk melihat apakah ada perbedaan jumlah kategori pada kedua dataset. Pada langkah ini diperoleh bahwa fitur yang bersifat kategorikal adalah *protocol_type*, *service*, dan *flag*. Dimana pada dataset *train* terdapat 84 fitur kategorikal dan pada dataset *test* terdapat 78 fitur kategorikal. Karena terdapat perbedaan fitur pada dataset *train* dan *test*, maka nantinya akan diberikan fitur kosong untuk dataset *test*.

4.2.3 Label Encoder

Label Encoder berfungsi untuk mengubah sebuah fitur yang bersifat kategorikal menjadi numerik, misal sebuah dataset memiliki fitur kategorikal “tinggi” dengan kategori “tinggi”, “sedang”, dan “rendah”, maka dengan *Label Encoder* fitur terseut akan menjadi “0”,”1”,”2”. *Label Encoder* selalu memulai bilangan dari 0. Sebelum melakukan *Label Encoder* pada dataset train dan test, yang perlu dilakukan adalah mencari setiap kategori dari fitur kategorikal. Jadi, pada tahap ini langkah yang pertama dilakukan adalah menyatukan semua fitur kategorikal. Untuk penelitian ini, menyatukan fitur

kategorikal dilakukan dengan memasukkan semua fitur kedalam bentuk table. 5 data teratas dari tabel tersebut adalah seperti pada **Tabel 4.2** berikut.

Tabel 4.2 Tabel fitur kategorikal

Index	*Protocol_type*	*Service*	*Flag*
0	*tcp*	*ftp_data*	*SF*
1	*udp*	*other*	*SF*
2	*tcp*	*private*	*S0*
3	*tcp*	*http*	*SF*
4	*tcp*	http	*SF*

Kemudian dilakukan identifikasi kategori pada seluruh fitur kategorikal sebelum digunakan *label encoder*. Sistem akan mengidentifikasi setiap kategori pada *protocol_type*, *service*, dan *flag* Kategori pada fitur diberikan dalam **Tabel 4.3** berikut

Tabel 4.3 Kategori dari fitur kategorikal

Index	Nama fitur
0	Protocol_type_icmp
1	Protocol_type_tcp
2	Protocol_type_udp
3	Service_IRC
4	Servce_X11
5	Service_Z39_50
6	Service_aol
7	Service_auth
8	Service_bgp
9	Service_courier
10	Service_csnet_ns
11	Service_ctf
12	Service_daytime
13	Service_discard
14	Service_domain

Index	Nama Fitur
15	Service_domain_u
16	Service_echo
17	Service_eco_i
18	Service_ecr_i
19	Service_efs
20	Service_exec
21	Service_finger
22	Service_ftp
23	Service_ftp_data
24	Service_gopher
25	Service_harvest
26	Service_hostnames
27	Service_http
28	Service_http_2784
29	Service_http_443
30	Service_http_8001
31	Service_imap4
32	Service_iso_tsap
33	Service_klogin
34	Service_kshell
35	Service_ldap
36	Service_link
37	Service_login
38	Service_mtp
39	Service_name
40	Service_netbios_dgm
41	Service_netbios_ns
42	Service_netbios_ssn
43	Service_netstat
44	Service_nnsp
45	Service_nntp
46	Service_nntp_u
47	Service_other

Index	Nama Fitur
48	Service_pm_dump
49	Service_pop_2
50	Service_pop_3
51	Service_printer
52	Service_private
53	Service_red_i
54	Service_remote_job
55	Service_rje
56	Service_shell
57	Service_smrp
58	Service_sql_net
59	Service_ssh
60	Service_sunrpc
61	Service_supdup
62	Service_systat
63	Service_telnet
64	Service_tftp_u
65	Service_tim_i
66	Service_time
67	Service_urh_i
68	Service_urp_i
69	Service_uucp
70	Service_uucp_path
71	Service_vmnet
72	Service_whois
73	Flag_OTH
74	Flah_REJ
75	Flag_RSTO
76	Flag_RSTOS0
77	Flag_RSTR
78	Flag_S0
79	Flag_S1
80	Flag_S2

Index	Nama Fitur
81	Flag_S3
82	Flag_SF
83	Flag_SH

Setelah didapatkan seluruh kategori dari fitur kategorikal, maka langkah selanjutnya adalah menggunakan *label encoder* pada fitur kategorikal tersebut, setiap kategori akan diubah sesuai indeks pada table. Menggunakan tabel sebagai contoh, maka *label encoder* akan menghasilkan

Tabel 4.4 Implementasi *Label Encoder*

Index	*Protocol_type*	*Service*	*Flag*
0	1	23	82
1	2	47	82
2	1	52	78
3	1	27	82
4	1	27	82

4.2.4 One-Hot Encoder

Langkah selanjutnya adalah mengubah fitur numerik pada **4.2.3** menjadi biner dengan *One-Hot Encoder*. *One Hot Encoder* merupakan sebuah metode dimana sebuah fitur diubah menjadi fitur binerik, dengan contoh apabila digunakan fitur kategorikal “tinggi”, maka dengan *One-Hot Encoder* bentuk tersebut akan menjadi

Tabel 4.5 Contoh Penerapan *One-Hot Encoder*

	Tinggi	Sedang	Rendah
1	1	0	0
2	0	1	0
3	0	0	1

Tabel diatas menunjukkan bahwa data 1 bernilai tinggi, data 2 bernilai sedang, dan data 3 bernilai rendah. *One-Hot Encoder* sendiri

akan membuat dataset memiliki fitur *dummy* sehingga harus diketahui dulu fitur *dummy* dari dataset, pada dataset NSL KDD, fitur *dummy* merupakan semua kategori dari fitur kategorikal. Menggunakan bagian *protocol_type* pada tabel sebagai contoh, maka penggunaan *one-hot encoder* akan menghasilkan

Tabel 4.6 Implementasi *One-Hot Encoder*

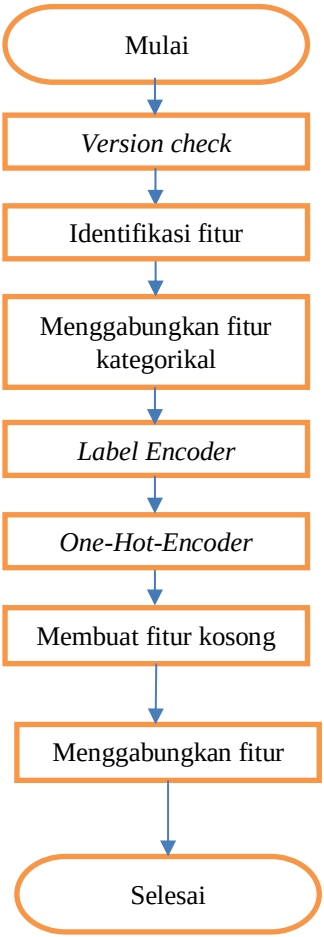
Index	*Protocol_type _icmp*	*Protocol_type_tcp*	*Protocol_type _udp*
0	0	1	0
1	0	0	1
2	0	1	0
3	0	1	0
4	0	1	0

4.2.5 Menggabungkan fitur

Dikarenakan pada dataset *train* dan *test* terdapat perbedaan jumlah kategori, maka pada dataset *test* diberikan fitur kosong untuk menyamakan jumlah fitur pada kedua dataset. Fitur kosong tersebut adalah *Service_http_2784, Service_red_i, Service_http_8001, Service_urh_i, Service _harvest, Service_aol*. Kemudian fitur yang telah menjadi diubah menjadi biner dengan *one-hot encoding* akan digabungkan dengan fitur pada dataset yang telah bernilai binerik. Sehingga dataset *train* dan *test* kini memiliki 123 fitur. Selanjutnya dataset terbaru akan disimpan dengan format .mat.

4.2.6 Diagram Alur *Preprocessing*

Alur *Preprocessing* dalam penelitian ini disajikan dalam **Gambar 4.1** berikut



Gambar 4.1 *Diagram Alir Preprocessing*

4.3 Pembuatan Sistem

Proses ini adalah langkah dimana sistem mulai dibuat untuk nantinya disimulasikan, berikut adalah beberapa langkah pada pembuatan sistem.

4.3.1. Struktur *Neural Networks*

Pada pembuatan sistem, akan ditentukan struktur neural networks yang dibutuhkan pada penelitian berikut ini

1. *Input Neuron*

   Input neuron merupakan jumlah dari fitur yang akan digunakan untuk membuat prediksi. *Input* nantinya akan berupa fitur yang telah direduksi dengan PCA.

2. *Output Neuron*

   Output Neuron merupakan jumlah prediksi yang diinginkan, dikarenakan data yang di *preprocessing* telah berbentuk binerik, maka dapat digunakan satu *output neuron* untuk setiap kelas positif. *Output neuron* juga akan menggunakan fungsi aktivasi *softmax* untuk memastikan jumlah dari tiap probabilitas bernilai 1.

3. *Hidden Layer*

   Jumlah *hidden layer* sangat bergantung pada masalah dan struktur dari *neural network*, jumlah dari hidden layer harus berada di niali yang tepat, tidak terlalu besar, dan tidak terlalu kecil. Dikarenakan dataset telah berbentuk numerik, maka pada penelitian berikut dapat digunakan 2 lapis *hidden layer* saja

4. *Loss Function*

   Dikarenakan output berjenis *classification*, maka *loss function* yang digunakan adalah *Cross-Entropy*

5. Jumlah *epoch*

Jumlah *epoch* yang akan digunakan adalah 3000

6. Learning Rate

Untuk menemukan *learning rate* terbaik, mulanya akan digunakan *learning rate* yang bernilai 1 x 10^{-6}, dan akan di multiplikasi sampai ditemukan nilai yang optimal. Nilai optimal yang ditemukan adalah 1 x 10^{-2}

7. Fungsi Aktivasi

Fungsi aktivasi yang akan digunakan pada *input layer* dan *hidden layer* adalah fungsi aktivasi ReLU, fungsi aktivasi ini merupakan fungsi aktivasi paling popular saat ini.

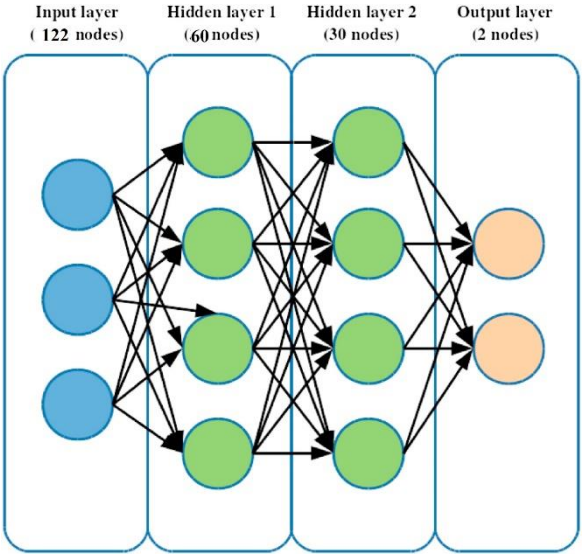
8. Dropout

Dropout adalah teknik regularisasi yang memberikan sistem dorongan kinerja besar. *Dropout* mematikan beberapa persen neuron dari tiap *layer* secara acak. Ini membuat jaringan lebih kuat karena tidak dapat bergantung pada set neuron input tertentu untuk membuat prediksi. Pengetahuan didistribusikan di antara seluruh jaringan. Dikarenakan *output* berbentuk klasifikasi, maka *dropout rate* yang digunakan adalah 0,5

9. Optimizer

Optimizer yang digunakan pada penelitian ini adalah algoritma ADAM. Algoritma ADAM merupakan algoritma yang baik sebagai langkah awal pada penelitian.

Dari struktur diatas, maka desain final dari *neural network* pada penilitian ini adalah seperti gambar 4.2 berikut

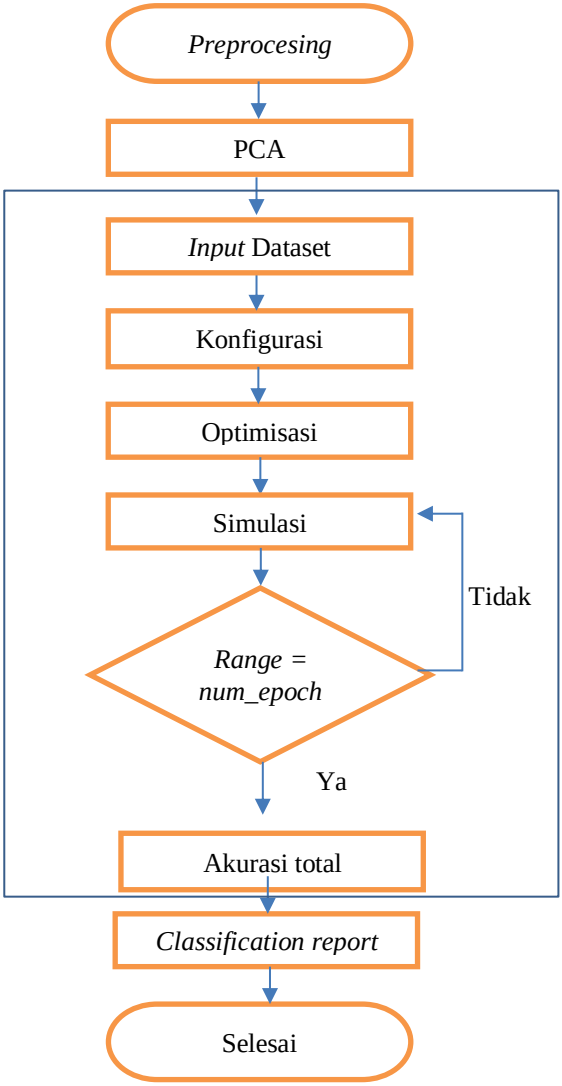


Gambar 4.2 *Desain final Neural Netwok*

Pada *input layer*, dikarenakan fitur setelah hasil *preprocessing* berjumlah 123, maka digunakan 122 *nodes* sebagai input layer. Pada *output layer*, dikarenakan fungsi aktivasi *softmax* pada *output*, maka dapat digunakan 2 *nodes* yang mewakili setiap kelas output, yaitu "0" dan "1". Untuk *hidden layer*, jumlah node ditentukan berdasarkan "*thumb-rule*" neural network, yakni nilai optimal pada *hidden layer* umumnya berada di antara jumlah *input* dan *output*. Sehingga pada *layer* pertama diberikan 60 *nodes*, dan *layer* kedua diberikan 30 *nodes*.

4.3.2. Diagram Alir Pembuatan Sistem

Diagram alur sistem disajikan dalam **Gambar 4.3** berikut



Gambar 4.3 *Diagram Alir Sistem*

4.3.3. Proses Pembuatan Sistem

Pada pembuatan sistem, akan dilakukan beberapa langkah awal, sistem akan menggunakan *numpy* karena system akan menyajikan hasil dalam bentuk array n-dimensi, dan sistem akan menggunakan komputasi ilmiah. Kemudian sistem akan menggunakan *pandas* untuk melakukan manipulasi dan analisis data. Digunakan juga modul seperti *copy* yang berguna untuk membuat Salinan dan *time* yang berguna untuk mencatatkan waktu. Selanjutnya digunakan modul *os* untuk berinteraksi dengan sistem operasi, dan digunakan juga modul *confusion matrix* dan *classification report* untuk mendapatkan hasil yang di inginkan.

Selanjutnya, akan digunakan modul *torch* yang memiliki array n-dimensional yang fleksibel atau tensor, yang berguna untuk *indexing*, *resizing*, dan *cloning*. Kemudian digunakan modul *nn* karena pada system akan digunkan *neural networks*.

Pada langkah selanjutnya diciptakan kelas *load dataset*, dimana x adalah *input* untuk model, dan y set label pada semua data di x. Fitur dipisahkan untuk memperkecil kemungkinan terjadi *overfitting* dan *underfitting*. Kemudian apabila x dan y pada test bernilai *false*, maka akan dilakukan perluasan bentuk array dan melakukan *horizontal stack* pada y, baik pada test maupun train. Kemudian x dan y akan dimuat dalam bentuk tensor yang memiliki isi berupa float, dan juga tensor berisi float.

Pada langkah selanjutnya, sistem akan menggunakan *Principal Component Analysis* (PCA) untuk visualisasi data. Beberapa hal yang dilakukan dalam implementasi PCA antara lain.

1. Substraksi mean

Langkah ini dilakukan dengan menemukan mean dari dataset, lalu dataset akan dikurangi dengan nilai mean tersebut, sehingga menciptakan sebuah dataset yang telah direduksi.

2. Dekomposisi Nilai Singular

Selanjutnya dataset yang telah direduksi akan dimplementasikan persamaan (3.1) untuk mendapatkan sebuah matriks σ.

$$\sigma = \frac{\begin{bmatrix} x_{1,1} & \cdots & x_{1,125973} \\ \vdots & \ddots & \vdots \\ x_{123,1} & \cdots & x_{123,125973} \end{bmatrix} \times \begin{bmatrix} x_{1,1} & \cdots & x_{1,123} \\ \vdots & \ddots & \vdots \\ x_{125973,1} & \cdots & x_{125973,123} \end{bmatrix}}{123}$$

$$= \frac{\begin{bmatrix} x_{1,1}x_{1,1} + \cdots + x_{1,125973}x_{125973,1} & \cdots & x_{1,1}x_{1,123} + \cdots \\ \vdots & \ddots & \vdots \\ x_{123,1}x_{1,1} + \cdots + x_{123,125973}x_{125973,1} & \cdots & x_{123,1}x_{1,123} + \cdots \end{bmatrix}}{123}$$

$$= \begin{bmatrix} \frac{x_{1,1}x_{1,1} + \cdots + x_{1,125973}x_{125973,1}}{123} & \cdots & \frac{x_{1,1}x_{1,123} + \cdots}{123} \\ \vdots & \ddots & \vdots \\ \frac{x_{123,1}x_{1,1} + \cdots + x_{123,125973}x_{125973,1}}{123} & \cdots & \frac{x_{123,1}x_{1,123} + \cdots}{123} \end{bmatrix}$$

Matriks σ kemudian diimplementasikan dekomposisi nilai singular seperti pada persamaan (3.2).

$$\sigma = USV^t$$

$$\begin{bmatrix} \frac{x_{1,1}x_{1,1} + \cdots + x_{1,125973}x_{125973,1}}{123} & \cdots & \frac{x_{1,1}x_{1,123} + \cdots}{123} \\ \vdots & \ddots & \vdots \\ \frac{x_{123,1}x_{1,1} + \cdots + x_{123,125973}x_{125973,1}}{123} & \cdots & \frac{x_{123,1}x_{1,123} + \cdots}{123} \end{bmatrix} = USV^t$$

$$\sigma = \begin{bmatrix} u_{1,1} & \cdots & u_{125973,1} \\ \vdots & \ddots & \vdots \\ u_{1,125973} & \cdots & u_{125973,125973} \end{bmatrix} \begin{bmatrix} s_1 & \cdots & 0 \\ \vdots & s_r & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} v_{1,1} & \cdots & v_{1,123} \\ \vdots & \ddots & \vdots \\ v_{123,1} & \cdots & v_{123,123} \end{bmatrix}$$

Dengan U adalah matriks $m \times m$, S adalah matriks $m \times n$, dan V adalah matriks $n \times n$, apabila σ adalah matriks $m \times n$. Matriks orthogonal yang dihasilkan dari operasi ini akan menjadi komponen utama untuk menghasilkan sebuah dataset yang telah direduksi.

4. Mengurangi dimensi data

Matriks orthogonal yang dihasilkan dari dekomposisi nilai singular akan dikalikan dengan dataset yang telah direduksi, untuk

menghasilkan sebuah data tereduksi yang terproyeksi pada komponen utama.

$$C = \sigma \times U$$

$$C = \begin{bmatrix} x_{1,1} & \cdots & x_{1,123} \\ \vdots & \ddots & \vdots \\ x_{125973,1} & \cdots & x_{125973,123} \end{bmatrix} \times \begin{bmatrix} u_{1,1} & \cdots & u_{125973,1} \\ \vdots & \ddots & \vdots \\ u_{1,125973} & \cdots & u_{125973,125973} \end{bmatrix}$$

$$C = \begin{bmatrix} x_{1,1}u_{1,1} + \cdots + x_{1,123}u_{1,125973} & \cdots & x_{1,1}u_{125973,1} + \cdots \\ \vdots & \ddots & \vdots \\ x_{1,123}u_{1,1} + \cdots + x_{125973,123}u_{1,125973} & \cdots & x_{125973,1}u_{125973,1} + \cdots \end{bmatrix}$$

Dengan C adalah matriks yang telah ditransformasikan dengan PCA. Kemudian class *get accuracy* yang mengimplementasikan persamaan (3.4) akan dibuat untuk menentukan akurasi dari *output*, awalnya akan dibuat nilai n_correct yang merupakan jumlah data yang benar pada data, setelah itu akan dicari akurasi.

Selanjutnya akan dirancang MLP pada sistem, mulanya dilakukan proses inisisasi untuk mengetahui layer linier, dan kemudian diciptakan 3 layer transformasi linier di proses inisiasi. Pada tahap ini akan digunakan 1 layer input, 2 layer hidden, dan 1 layer output, seperti pada rancangan neural network. Setiap layer akan menggunakan mungsi aktivasi RELu, kecuali pada bagian output. Layer output akan menggunakan fungsi softmax, fungsi ini berfungsi dengan baik pada sistem yang menggunakan data tipe binerik dan memiliki output klasifikasi.

Langkah selanjutnya adalah membangun kelas *train model* yang berfungsi untuk melakukan konfigurasi ketika system melakukan *train*. Pada langkah ini juga diimplementasikan nilai-nilai yang telah dirancang pada sub-subbab 4.3.1 seperti ukuran layer input, hidden, dan output. Selanjutnya pada kelas tersebut dilakukan optimisasi, optimisasi akan menggunakan algoritma optimisasi ADAM seperti pada persamaan (3.6),

Kemudian Sistem akan menghitung akurasi, dan menghilangkan verbose, yakni situasi dimana sistem melakukan simulasi lebih dari

seharusnya, pada training juga akan mengubah akurasi sesuai dengan akurasi rata-rata. Langkah ini dilakukan untuk membuat peluang sistem mengalami *underfitting* dan *overfitting* menjadi kecil.

Selanjutnya diberikan elapsed time untuk mengetahui berapa lama waktu yang dibutuhkan untuk simulasi. Pada langkah ini juaga akan diberikan *classification report* yang berisi akurasi, *precision*, *recall*, dan *f1-score* dari setiap iterasi.

4.4. Simulasi Sistem

Subbab ini akan membahas saat sistem melakukan simulasi, sistem melakukan 174 iterasi, dengan hasil setiap iterasi akan berbentuk seperti pada **Gambar 4.4** berikut

```
------------------------------pca:   9------------------------------
epoch:    0 | loss: 0.6643 | Train accuracy: 68.1% | Test accuracy: 55.1%
epoch:  500 | loss: 0.1405 | Train accuracy: 93.4% | Test accuracy: 81.0%
epoch: 1000 | loss: 0.1378 | Train accuracy: 93.4% | Test accuracy: 81.1%
epoch: 1500 | loss: 0.1364 | Train accuracy: 93.5% | Test accuracy: 81.0%
epoch: 2000 | loss: 0.1354 | Train accuracy: 93.6% | Test accuracy: 80.9%
epoch: 2500 | loss: 0.1349 | Train accuracy: 93.6% | Test accuracy: 80.8%
Training complete in 0m 20s
train acc: 93.4%, test acc: 81.1%
              precision    recall  f1-score   support

           0       1.00      0.58      0.73     37000
           1       0.74      1.00      0.85     45332

   micro avg       0.81      0.81      0.81     82332
   macro avg       0.87      0.79      0.79     82332
weighted avg       0.86      0.81      0.80     82332
```

***Gambar 4.4** Iterasi dari Hasil Simulasi*

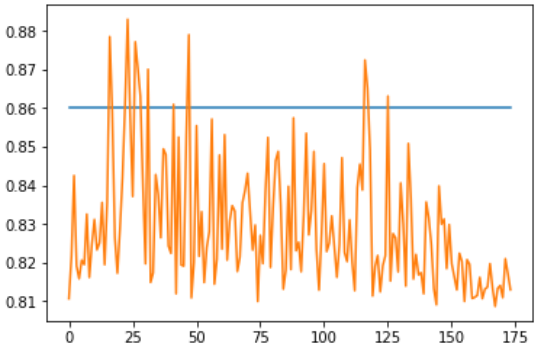
Setiap nilai akurasi *test*, *precision*, *recall*, dan *score* pada tiap iterasi akan disajikan dalam bentuk tensor seperti pada *lampiran 1*, 5 hasil simulasi pertama akan disajikan dalam bentuk tabel seperti pada **Tabel 4.7** berikut

Tabel 4.7 Sampel simulasi

Iterasi	Akurasi	Precision	Recall	Score
1	81%	0.74	1	0.85
2	82%	0.76	0.99	0.86
3	84%	0.80	0.95	0.87
4	82%	0.76	0.98	0.86
5	82%	0.75	1	0.86

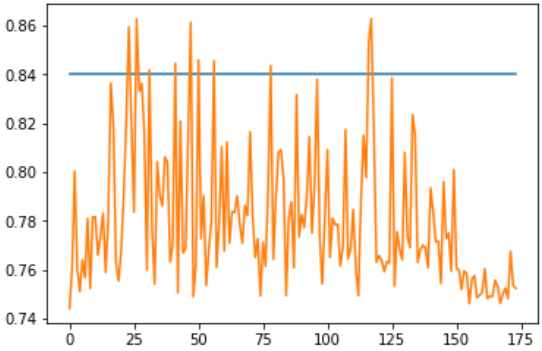
4.5. Analisis Hasil Simulasi

Pada tahap ini, akan dibahas mengenai hasil yang didapatkan dari simulasi. Hasil yang telah disajikan dalam bentuk *tensor* akan di analisis setiap hasil. Kemudian akan diberikan grafik untuk melihat kemampuan sistem dalam mendeteksi dan mengklasifikasikan intrusi.



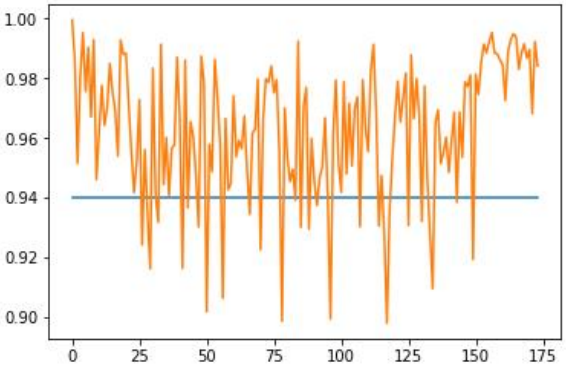
***Gambar 4.5** Grafik Akurasi*

Pada **Gambar 4.5**, didapatkan grafik dari akurasi *test*, dimana akurasi tertinggi ada pada angka 88%, dari grafik juga dapat dilihat bahwa sistem memiliki akurasi diatas 81%. Pada iterasi awal, sistem mengalami penaikan, sebelum pada iterasi bagian akhir mengalami penurunan.



Gambar 4.6 *Grafik Precision*

Pada gambar 4.5, *precision* mengalami nilai terendah pada nilai 74%, dan tertinggi 86%, pada grafik dapat dilihat bahwa sistem mengalami kenaikan pada iterasi awal, namun mengalami penurunan pada iterasi akhir.



Gambar 4.7 *Grafik Recall*

Pada nilai *recall*, kebanyakan iterasi memiliki nilai diatas rata-rata, dengan nilai rata sebesar 94%, nilai recall tertinggi ada pada 100%, dan terendah terdapat pada 90%.

BAB V
PENUTUP

Bab ini berisi tentang kesimpulan yang dihasilkan berdasarkan penelitian yang telah dilakukan serta saran yang diberikan jika penelitian ini ingin dikembangkan.

5.1 Kesimpulan

1. Berdasarkan hasil simulasi, IDS dengan metode *neural networks* memiliki performa yang baik dalam mendeteksi intrusi, dengan akurasi diatas 80%, dan mengalami penurunan kemampuan di iterasi ke-150.
2. Berdasarkan hasil rasio *recall* dan *precision*, sistem mampu untuk mengklasifikasi salah positif dan salah negatif dengan baik, dimana nilai *precision* dan *recall* selalu berada diatas 0.50.

5.2 Saran

Pada penelitian selanjutnya dapat dibuat sistem untuk membandingkan IDS dengan berbagai metode *machine learning* seperti *feature selection, decision tree, support vector machine,* dll.

DAFTAR PUSTAKA

[1] Schatz, D., Bashroush, R., Wall, J. (2017). “Towards a More Representative Definition of Cyber Security”. Journal of Digital Forensics, Security and Law vol. 12: No. 2, Article 8. DOI: 10.12394/jdfl.2017.1476.

[2] Steven, Tim. (2018). “Global Cybersecurity: New Directions in Theory and Methods”. Politics and Governance 6(2): 1. DOI: 10.17645/pag.v6i2.1569.

[3] Axelsson, S. (2000). "Intrusion Detection Systems: A Survey and Taxonomy”. Sweden: Department of Computer Engineering Chalmers University of Technology Goteborg.

[4] Zhenwei Yu, Tsai, Jeffrey J.P. (2011). “Intrusion Detection: A Machine Learning Approach (3)”. London: Imperial College Press.

[5] Bethge, Matthias; Ecker, Alexander S.; Gatys, Leon A. (2015). "A Neural Algorithm of Artistic Style". arXiv. DOI: 10.1167/16.12.326.

[6] Hajimirzaei, Bahram, Navimipour, Nima Jafari. (2018). “Intrusion Detection for Cloud Computing Using Neural Networks and Artificial Bee Colony Optimization Algorithm”. ICT Express 5(1), 56-59.

[7] Wang, Cheng-Ru. (2018). “Network Intrusion Detection Using Equality Constrained-Optimization-Based Extreme Machine Learning”. Knowledge-Based System 147, 68-80.

[8] Shenfield A., Day D., Ayesh, A. (2018). “Intelligent Intrusion Detection System Using Artificial Neural Network”. ICT Express 4(2), 95-99. DOI: 10.1016/j.cte.2018.04.003.

[9] Martellini, Maurizio, Malizia, Andrea. (2017) “Cyber and Chemical, Biological, Radiological, Nuclear, Explosives

Challenges: Threats and Counter Efforts". New York: Springer.

[10] Mohammed, Mohssen, Rehman, Habib-ur (2012). "Honeypots and Routers: Collecting Internet Attacks". Florida: CRC Press.

[11] Reddy, Ekambaram Kesavulu. (2013). "Neural Networks for Intrusion Detection and Its Application" Lecture Notes in Engineering and Computer Science.

[12] Abdi. H., Williams, L.J. (2010). "Principal component analysis". Wiley Interdisciplinary Reviews: Computational Statistics. 2.

[13] Jolliffe I.T. (2002). "Principal Component Analysis (2)". New York: Springer.

[14] Shaw, P.J.A. (2003). "Multivariate statistics for the Environmental Sciences". Hodder-Arnold.

[15] A. A. Miranda, Y. A. Le Borgne, and G. Bontempi. (2008). "New Routes from Minimal Approximation Error to Principal Components". Neural Processing Letters 27, 197-207.

[16] Johnson, Richard A & Wichern, Dean W. (1998). "Applied Multivariate Statistical Analysis". New Jersey: Prentice-Hall International Inc.

[17] Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. (2009). "The Elements of Statistical Learning: Data Mining, Inference, and Prediction". New Yok: Springer.

[18] Rumelhart, David E., Geoffrey E. Hinton, dan R. J. Williams. (1986). "Learning Internal Representations by Error Propagation". MIT Press.

[19] Fausett, Laurene. (1994). "Fundamentals of Neural Networks" Prentice-Hall.

[20] Hassan, Asma A., Sheta, Alaa F., Wahbi, Talaat M. (2015). "Intrusion Detection Using Neural Network: A Literature Review". IJSR.

LAMPIRAN

Lampiran A Tensor Hasil Simulasi

```
tensor([[0.8107, 0.7443, 0.9996, 0.8532],
        [0.8217, 0.7609, 0.9859, 0.8589],
        [0.8426, 0.8003, 0.9515, 0.8694],
        [0.8191, 0.7604, 0.9805, 0.8565],
        [0.8158, 0.7511, 0.9953, 0.8561],
        [0.8206, 0.7640, 0.9756, 0.8569],
        [0.8195, 0.7568, 0.9904, 0.8580],
        [0.8326, 0.7810, 0.9671, 0.8641],
        [0.8162, 0.7524, 0.9929, 0.8561],
        [0.8247, 0.7815, 0.9461, 0.8560],
        [0.8312, 0.7816, 0.9622, 0.8626],
        [0.8233, 0.7661, 0.9777, 0.8590],
        [0.8251, 0.7738, 0.9643, 0.8586],
        [0.8356, 0.7831, 0.9700, 0.8666],
        [0.8195, 0.7589, 0.9850, 0.8573],
        [0.8357, 0.7808, 0.9754, 0.8674],
        [0.8785, 0.8363, 0.9691, 0.8978],
        [0.8587, 0.8191, 0.9540, 0.8814],
        [0.8265, 0.7633, 0.9929, 0.8630],
        [0.8173, 0.7554, 0.9880, 0.8562],
        [0.8286, 0.7674, 0.9884, 0.8640],
        [0.8421, 0.7901, 0.9712, 0.8713],
        [0.8611, 0.8214, 0.9555, 0.8834],
        [0.8830, 0.8593, 0.9417, 0.8986],
        [0.8581, 0.8195, 0.9521, 0.8808],
        [0.8371, 0.7836, 0.9729, 0.8680],
        [0.8772, 0.8626, 0.9242, 0.8923],
        [0.8704, 0.8332, 0.9561, 0.8904],
        [0.8631, 0.8361, 0.9345, 0.8826],
        [0.8402, 0.8161, 0.9162, 0.8632],
        [0.8197, 0.7598, 0.9834, 0.8573],
        [0.8701, 0.8418, 0.9408, 0.8886],
        [0.8149, 0.7766, 0.9318, 0.8472],
        [0.8173, 0.7542, 0.9914, 0.8567],
        [0.8428, 0.8042, 0.9444, 0.8687],
        [0.8377, 0.7902, 0.9602, 0.8669],
        [0.8264, 0.7861, 0.9407, 0.8565],
```

```
[0.8495, 0.8061, 0.9567, 0.8750],
[0.8480, 0.8038, 0.9577, 0.8740],
[0.8243, 0.7633, 0.9871, 0.8609],
[0.8224, 0.7698, 0.9664, 0.8570],
[0.8610, 0.8444, 0.9163, 0.8789],
[0.8119, 0.7506, 0.9861, 0.8524],
[0.8525, 0.8207, 0.9367, 0.8749],
[0.8194, 0.7669, 0.9654, 0.8548],
[0.8191, 0.7687, 0.9605, 0.8540],
[0.8530, 0.8154, 0.9475, 0.8765],
[0.8790, 0.8612, 0.9302, 0.8944],
[0.8109, 0.7490, 0.9874, 0.8518],
[0.8201, 0.7622, 0.9784, 0.8569],
[0.8555, 0.8459, 0.9018, 0.8729],
[0.8216, 0.7727, 0.9578, 0.8554],
[0.8332, 0.7903, 0.9487, 0.8623],
[0.8148, 0.7535, 0.9863, 0.8544],
[0.8238, 0.7684, 0.9736, 0.8589],
[0.8280, 0.7798, 0.9582, 0.8599],
[0.8572, 0.8454, 0.9063, 0.8748],
[0.8144, 0.7610, 0.9665, 0.8515],
[0.8209, 0.7787, 0.9427, 0.8529],
[0.8479, 0.8103, 0.9449, 0.8724],
[0.8235, 0.7677, 0.9741, 0.8587],
[0.8531, 0.8122, 0.9538, 0.8773],
[0.8207, 0.7710, 0.9592, 0.8549],
[0.8308, 0.7838, 0.9565, 0.8616],
[0.8347, 0.7833, 0.9674, 0.8657],
[0.8333, 0.7902, 0.9494, 0.8625],
[0.8177, 0.7787, 0.9345, 0.8495],
[0.8214, 0.7708, 0.9616, 0.8557],
[0.8355, 0.7862, 0.9631, 0.8657],
[0.8387, 0.7822, 0.9797, 0.8699],
[0.8431, 0.8164, 0.9225, 0.8662],
[0.8335, 0.7821, 0.9669, 0.8648],
[0.8233, 0.7651, 0.9799, 0.8593],
[0.8297, 0.7727, 0.9786, 0.8636],
[0.8099, 0.7493, 0.9842, 0.8508],
[0.8271, 0.7712, 0.9752, 0.8613],
```

```
[0.8198, 0.7614, 0.9795, 0.8568],
[0.8385, 0.7933, 0.9555, 0.8669],
[0.8524, 0.8436, 0.8986, 0.8702],
[0.8188, 0.7642, 0.9701, 0.8549],
[0.8343, 0.7896, 0.9531, 0.8637],
[0.8464, 0.8081, 0.9454, 0.8714],
[0.8488, 0.8090, 0.9495, 0.8737],
[0.8349, 0.7971, 0.9392, 0.8623],
[0.8131, 0.7494, 0.9924, 0.8540],
[0.8180, 0.7811, 0.9302, 0.8491],
[0.8398, 0.7878, 0.9704, 0.8696],
[0.8182, 0.7608, 0.9770, 0.8555],
[0.8575, 0.8315, 0.9295, 0.8778],
[0.8231, 0.7734, 0.9599, 0.8566],
[0.8253, 0.7825, 0.9455, 0.8563],
[0.8176, 0.7773, 0.9375, 0.8499],
[0.8324, 0.7904, 0.9468, 0.8615],
[0.8535, 0.8144, 0.9506, 0.8772],
[0.8272, 0.7750, 0.9668, 0.8603],
[0.8334, 0.7950, 0.9398, 0.8614],
[0.8488, 0.8379, 0.8994, 0.8676],
[0.8233, 0.7732, 0.9610, 0.8569],
[0.8129, 0.7542, 0.9794, 0.8521],
[0.8275, 0.7823, 0.9517, 0.8587],
[0.8456, 0.8091, 0.9419, 0.8704],
[0.8229, 0.7651, 0.9788, 0.8589],
[0.8250, 0.7811, 0.9480, 0.8565],
[0.8321, 0.7784, 0.9716, 0.8643],
[0.8240, 0.7786, 0.9507, 0.8561],
[0.8162, 0.7616, 0.9698, 0.8532],
[0.8246, 0.7691, 0.9737, 0.8594],
[0.8472, 0.8174, 0.9303, 0.8702],
[0.8226, 0.7645, 0.9795, 0.8588],
[0.8203, 0.7687, 0.9636, 0.8552],
[0.8311, 0.7846, 0.9555, 0.8617],
[0.8200, 0.7604, 0.9828, 0.8574],
[0.8127, 0.7494, 0.9914, 0.8536],
[0.8394, 0.7883, 0.9684, 0.8691],
[0.8455, 0.8150, 0.9306, 0.8690],
```

```
[0.8388, 0.7978, 0.9474, 0.8662],
[0.8725, 0.8533, 0.9280, 0.8891],
[0.8652, 0.8627, 0.8980, 0.8800],
[0.8491, 0.8167, 0.9359, 0.8723],
[0.8113, 0.7630, 0.9535, 0.8477],
[0.8191, 0.7658, 0.9673, 0.8548],
[0.8218, 0.7639, 0.9791, 0.8582],
[0.8124, 0.7593, 0.9654, 0.8500],
[0.8192, 0.7633, 0.9735, 0.8557],
[0.8218, 0.7627, 0.9817, 0.8585],
[0.8631, 0.8385, 0.9307, 0.8822],
[0.8152, 0.7533, 0.9878, 0.8548],
[0.8275, 0.7755, 0.9666, 0.8605],
[0.8264, 0.7684, 0.9799, 0.8614],
[0.8176, 0.7641, 0.9675, 0.8538],
[0.8406, 0.8079, 0.9321, 0.8656],
[0.8300, 0.7735, 0.9775, 0.8636],
[0.8139, 0.7691, 0.9461, 0.8485],
[0.8509, 0.8234, 0.9283, 0.8727],
[0.8366, 0.8151, 0.9096, 0.8597],
[0.8157, 0.7630, 0.9652, 0.8523],
[0.8221, 0.7682, 0.9694, 0.8572],
[0.8168, 0.7701, 0.9514, 0.8512],
[0.8174, 0.7690, 0.9551, 0.8520],
[0.8119, 0.7609, 0.9602, 0.8490],
[0.8357, 0.7935, 0.9485, 0.8641],
[0.8316, 0.7836, 0.9590, 0.8625],
[0.8248, 0.7715, 0.9686, 0.8589],
[0.8132, 0.7716, 0.9385, 0.8469],
[0.8091, 0.7544, 0.9687, 0.8482],
[0.8399, 0.7960, 0.9535, 0.8677],
[0.8300, 0.7729, 0.9788, 0.8637],
[0.8313, 0.7750, 0.9773, 0.8645],
[0.8184, 0.7594, 0.9811, 0.8561],
[0.8298, 0.8010, 0.9193, 0.8561],
[0.8197, 0.7607, 0.9813, 0.8570],
[0.8164, 0.7598, 0.9747, 0.8539],
[0.8129, 0.7520, 0.9852, 0.8529],
[0.8224, 0.7595, 0.9915, 0.8601],
```

```
[0.8203, 0.7584, 0.9885, 0.8583],
[0.8100, 0.7462, 0.9923, 0.8519],
[0.8208, 0.7562, 0.9953, 0.8594],
[0.8195, 0.7576, 0.9885, 0.8578],
[0.8107, 0.7486, 0.9882, 0.8518],
[0.8110, 0.7496, 0.9862, 0.8518],
[0.8115, 0.7507, 0.9847, 0.8519],
[0.8162, 0.7604, 0.9726, 0.8535],
[0.8107, 0.7481, 0.9891, 0.8519],
[0.8131, 0.7493, 0.9929, 0.8540],
[0.8137, 0.7490, 0.9949, 0.8546],
[0.8197, 0.7557, 0.9939, 0.8586],
[0.8134, 0.7533, 0.9831, 0.8530],
[0.8087, 0.7463, 0.9885, 0.8505],
[0.8133, 0.7499, 0.9916, 0.8540],
[0.8141, 0.7525, 0.9868, 0.8539],
[0.8109, 0.7482, 0.9896, 0.8522],
[0.8210, 0.7675, 0.9681, 0.8562],
[0.8172, 0.7537, 0.9923, 0.8567],
[0.8130, 0.7524, 0.9841, 0.8528]])
```