TESIS

INTRUSION DETECTION SYSTEM (IDS) BERBASIS DEEP LEARNING DENGAN METODE RECURRENT NEURAL NETWORK (RNN)

Disusun oleh:

Nama : **Omar Muhammad Altoumi Alsyaibani**
NIM : **20.77.1250**
Konsentrasi : **Business Intelligence**

PROGRAM STUDI S2 TEKNIK INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2021

TESIS

INTRUSION DETECTION SYSTEM (IDS) BERBASIS DEEP LEARNING DENGAN METODE RECURRENT NEURAL NETWORK (RNN)

DEEP LEARNING BASED INTRUSION DETECTION SYSTEM (IDS) WITH RECURRENT NEURAL NETWORK (RNN) METHOD

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Magister

Disusun oleh:

Nama	**: Omar Muhammad Altoumi Alsyaibani**
NIM	**: 20.77.1250**
Konsentrasi	**: Business Intelligence**

PROGRAM STUDI S2 TEKNIK INFORMATIKA

PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA

YOGYAKARTA

2021

HALAMAN PENGESAHAN

INTRUSION DETECTION SYSTEM (IDS) BERBASIS DEEP LEARNING DENGAN METODE RECURRENT NEURAL NETWORK (RNN)

DEEP LEARNING BASED INTRUSION DETECTION SYSTEM (IDS) WITH RECURRENT NEURAL NETWORK (RNN) METHOD

Dipersiapkan dan Disusun oleh

Omar Muhammad Altoumi Alsyaibani

20.77.1250

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Teknik Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Selasa, 2 November 2021

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 2 November 2021
Rektor

Prof. Dr. M. Suyanto, M.M.
NIK. 190302001

HALAMAN PERSETUJUAN

INTRUSION DETECTION SYSTEM (IDS) BERBASIS DEEP LEARNING DENGAN METODE RECURRENT NEURAL NETWORK (RNN)

DEEP LEARNING BASED INTRUSION DETECTION SYSTEM (IDS) WITH RECURRENT NEURAL NETWORK (RNN) METHOD

Dipersiapkan dan Disusun oleh

Omar Muhammad Altoumi Alsyaibani

20.77.1250

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Teknik Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Selasa, 2 November 2021

Pembimbing Utama	**Anggota Tim Penguji**
Prof. Dr. Ema Utami, S.Si., M.Kom.	**Dr. Andi Sunyoto, M.Kom.**
NIK. 190302037	**NIK. 190302052**
	Alva Hendi Muhammad, S.T., M.Eng., Ph.D.
Pembimbing Pendamping	**NIK. 190302493**
Anggit Dwi Hartanto, M.Kom.	**Prof. Dr. Ema Utami, S.Si., M.Kom.**
NIK. 190302163	**NIK. 190302037**

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 2 November 2021
Direktur Program Pascasarjana

Dr. Kusrini, M.Kom.
NIK. 190302106

HALAMAN PERNYATAAN KEASLIAN TESIS

Yang bertandatangan di bawah ini,

Nama mahasiswa : Omar Muhammad Altoumi Alsyaibani
NIM : 20.77.1250
Konsentrasi : Business Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
Intrusion Detection System (IDS) Berbasis Deep Learning Dengan Metode Recurrent Neural Network (RNN)

Dosen Pembimbing Utama : Prof. Dr. Ema Utami, S.Si., M.Kom.
Dosen Pembimbing Pendamping : Anggit Dwi Hartanto, M.Kom.

1. Karya tulis ini adalah benar-benar ASLI dan BELUM PERNAH diajukan untuk mendapatkan gelar akademik, baik di Universitas AMIKOM Yogyakarta maupun di Perguruan Tinggi lainnya
2. Karya tulis ini merupakan gagasan, rumusan dan penelitian SAYA sendiri, tanpa bantuan pihak lain kecuali arahan dari Tim Dosen Pembimbing
3. Dalam karya tulis ini tidak terdapat karya atau pendapat orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan disebutkan dalam Daftar Pustaka pada karya tulis ini
4. Perangkat lunak yang digunakan dalam penelitian ini sepenuhnya menjadi tanggung jawab SAYA, bukan tanggung jawab Universitas AMIKOM Yogyakarta
5. Pernyataan ini SAYA buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka SAYA bersedia menerima SANKSI AKADEMIK dengan pencabutan gelar yang sudah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi

Yogyakarta, 2 November 2021
Yang Menyatakan,

Omar Muhammad Altoumi Alsyaibani

HALAMAN PERSEMBAHAN

Segala puji bagi Allah SWT., Tuhan semesta Alam, karena dengan izin-Nya jualah laporan tesis ini dapat terselesaikan. Laporan tesis ini saya persembahkan kepada:

1. Tika, yang selalu memberikan senyuman meski dalam keadaan sulit, selalu memberikan dukungan dan mau memaklumi.
2. Aisha, yang selalu menjadi peluruh lelah setiap harinya.
3. Abah dan Mama tercinta, terima kasih akan kasih sayangnya, dukungan dan doanya di setiap waktu.
4. Ayah, Mamah, Bima dan Mira yang turut selalu mendukung proses penulis selama studi.
5. Seluruh Dosen dan Staf Tata Usaha Universitas AMIKOM Yogyakarta khususnya di Program Pascasarjana.
6. Teman-teman PJJ Angkatan 3 yang selalu mau bekerja sama dalam proses perkuliahan.
7. Seluruh rekan guru di TKJ yang mau memaklumi saat tugas kuliah saya sedang *peak*.

HALAMAN MOTTO

Man jadda wajada

Belilah masa depan dengan harga sekarang

A candle loses nothing by lighting another candle

إِنَّ مَعَ الْعُسْرِ يُسْرًا

KATA PENGANTAR

Alhamdulillaahirrabbilaalaamiin. Dengan izin Allah SWT, Tuhan semesta alamlah maka laporan tesis ini dapat terselesaikan. Banyak pihak yang telah membantu penulis dalam menyelesaikan penelitian ini, baik secara langsung maupun tidak langsung. Oleh karena itu, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Dr. M. Suyanto, M.M. selaku Rektor Universitas AMIKOM Yogyakarta.
2. Ibu Prof. Dr. Kusrini, M.Kom. selaku Direktur Program Pascasarjana Universitas AMIKOM Yogyakarta.
3. Ibu Prof. Dr. Ema Utami, S.Si., M.Kom. selaku Wakil Direktur Program Pascasarjana Universitas AMIKOM Yogyakarta sekaligus selaku Pembimbing Utama.
4. Bapak Anggit Dwi Hartanto, M.Kom. selaku dosen Pembimbing Pendamping.
5. Seluruh Dewan Penguji yang telah memberikan banyak masukan baik pada SPT, SHPT maupun UT untuk memperbaiki persiapan, proses dan pelaporan hasil penelitian ini.
6. Semua pihak yang telah membantu penulis selama penelitian ini.

Penulis menyadari bahwa penelitian ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang sifatnya membangun.

Banjarbaru, 2 November 2021

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR ISTILAH

Dataset: Kumpulan data yang umumnya digunakan dalam penelitian di bidang *data science*, *machine learning* dan *deep learning*. Ada *dataset* yang bersifat *public*, yakni bisa diakses secara bebas dan ada pula yang bersifat *private*.

Neural Network: Dikenal juga sebagai jaringan syaraf tiruan (JST) di dalam Bahasa, yaitu teknik komputasi yang mengikuti sistem syaraf manusia. *Neural network* biasanya berisi banyak *neuron*.

Neuron: Merupakan satuan unit kerja dalam sistem syaraf. Di dalam JST, istilah *neuron* juga diadopsi untuk merepresentasikan bagian-bagian terkecil di dalam JST. *Neuron* berisi nilai yang diperbaharui secara berkala selama proses *training* model

Model: Istilah umum yang digunakan untuk merepresentasikan JST yang sudah dilatih. Secara matematis, model biasanya digambarkan dengan sebuah fungsi atau rumus.

Layer (lapisan): Di dalam JST, banyak *neuron* disusun ke dalam sebuah layer atau lapisan. JST dapat mengandung lebih dari 1 layer. Di dalam ilmu JST, dikenal juga istilah *input layer* yang berarti susunan *neuron* yang menjadi jalan untuk data atau input masuk ke dalam JST dan *output layer* yang merupakan satu atau lebih *neuron* yang mengeluarkan hasil akhir komputasi JST yang umumnya berupa angka.

Hidden layer: JST dapat mempunyai banyak layer. Layer-layer yang berada diantara *input layer* dan *output layer* disebut *hidden layer*.

Epoch: Sebuah JST dapat dilatih dengan data yang sama berkali-kali. Setiap menyelesaikan satu kali latihan (*epoch*), nilai *weight* atau bobot pada setiap neuron diperbaharui agar dapat nilai keluarannya dapat mendekati output yang seharusnya berdasarkan label data.

INTISARI

Dalam penelitian ini, LSTM dan GRU *layer* digunakan dalam pengembangan model *Intrusion Detection System*. *Bidirectional layer* juga diimplementasikan dalam beberapa percobaan untuk menguji peningkatan akurasi dalam kinerja model.

Dataset yang digunakan dalam penelitian ini adalah CIC IDS 2017. *Dataset* dibagi menjadi 3 bagian, untuk keperluan *training*, validasi dan pengujian. Model dilatih dengan pendekatan *binary classification*. Adam dan SGD diimplementasikan sebagai optimasi model secara bergantian. Fungsi aktivasi pada model digunakan Softmax, Tanh, Relu, Lrelu dan Prelu secara bergantian, sedangkan nilai 0.001 dan 0.0001 digunakan sebagai nilai *learning rate*. Performa tertinggi dihasilkan oleh model kombinasi LSTM-GRU dengan menggunakan fungsi optimasi Adam. Sementara itu, durasi pelatihan dan validasi terpendek dihasilkan oleh model kombinasi LSTM-LSTM yang juga menggunakan fungsi optimasi Adam. Perbedaan akurasi serta durasi *training* dan validasi antara kombinasi model LSTM-GRU dan LSTM-LSTM sangat kecil.

Oleh karena itu, direkomendasikan kedua model ini untuk digunakan oleh peneliti selanjutnya. Selanjutnya, penggunaan *Bidirectional layer* pada *neural network* menghasilkan kinerja model yang juga tinggi, namun durasi pelatihan jaringan saraf menjadi lebih lama.

Kata kunci: GRU; IDS; CIC IDS 2017; *Bidirectional layer*

ABSTRACT

In this experiment, LSTM and GRU layers were used in developing Intrusion Detection System model. Bidirectional layer was also implemented in several experiments to examine the increasing in accuracy in model performance.

Dataset used in the study was CIC IDS 2017. The dataset was divided into 3 parts, for training, validation and testing purposes. The training models were done in binary classification approach. Adam and SGD were implemented as optimizer of models interchangeably. For the activation function in the model, Softmax, Tanh, Relu, Leaky Relu and Prelu were used alternately, while the values of 0.001 and 0.0001 were used as learning rates. The highest performance was produced by the LSTM-GRU combination model using Adam optimizer. Meanwhile, the shortest training and validation durations was generated by the LSTM-LSTM combination model which also used the Adam optimizer. The difference in accuracy and duration of training and validation between the combination of the LSTM-GRU and LSTM-LSTM models is very small.

Thus, we recommend these two models to be used by future researchers. Furthermore, Bidirectional layer usage on the neural network resulted high model performance as well, yet the training duration of the neural network became longer.

Keyword: LSTM; GRU; IDS; CIC IDS 2017; Bidirectional layer

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Saat ini, internet telah menjadi tulang punggung aktivitas dalam berbagai aspek kehidupan. Pandemi COVID-19 memaksa kita untuk melakukan aktivitas secara online, sehingga kebutuhan akan koneksi Internet semakin meningkat. Dengan semakin meningkatnya aktivitas masyarakat di internet tentunya menjadikan aspek keamanan jaringan menjadi penting untuk diperhatikan. Berdasarkan data yang dirilis oleh (CyberEdge Group, 2021) bahwa 86,2% organisasi setidaknya pernah sekali diserang secara berhasil oleh *hacker*. Jumlah ini meningkat signifikan jika dibandingkan dengan tahun-tahun sebelumnya. Misalnya pada tahun 2020 hanya mencapai 80,7% dan tahun 2019 mencapai 78,0%.

Intrusion Detection System (IDS) merupakan sistem yang bisa berupa *hardware* maupun *software* yang dapat mendeteksi keberadaan trafik jaringan yang mencurigakan di jaringan. Menurut (Bace & Mell, 2001) *intrusion detection* adalah teknik pemantauan (*monitoring*) aktivitas dan *event* yang terjadi di sistem atau jaringan komputer dan menganalisisnya untuk mengetahui tanda-tanda gangguan, yang didefinisikan sebagai upaya untuk membahayakan kerahasiaan, integritas, ketersediaan, atau untuk melewati mekanisme keamanan komputer atau jaringan. Beberapa perusahaan menggunakan IDS sebagai sistem tambahan untuk mendukung *firewall* di jaringan mereka (Mazini et al., 2019).

IDS tradisional menggunakan *database signature* untuk dapat membedakan antara *traffic* normal dan serangan (Bhuyan et al., 2014). Jika jenis *traffic* tidak ada dalam *database signature*, IDS tidak akan dapat mengenali *traffic*. Hal ini akan menimbulkan tingginya deteksi yang tidak tepat sasaran atau sering disebut *false alarm* (Khraisat et al., 2019), seperti mengirimkan peringatan (*alert*) saat tidak ada trafik yang berbahaya. Hal ini akan menambah beban kerja analis keamanan jaringan. Jika analis keamanan jaringan terus menerus mendapatkan *false alarm alert*, maka ini memungkinkan serangan yang sebenarnya menyusup pada salah satu *false alarm* tersebut. Perubahan kondisi dan lingkungan jaringan yang sangat cepat dan kemunculan berbagai teknologi baru di jaringan juga memunculkan berbagai jenis serangan baru. Oleh karena itu, menjadi *urgent* untuk mengembangkan IDS yang dapat mendeteksi serangan yang bahkan tidak dikenalinya (*unknown attack*).

Untuk menangani hal tersebut, banyak peneliti yang mulai menggunakan *Machine learning* sebagai metode untuk membangun IDS. *Machine learning* merupakan salah satu teknik kecerdasan buatan yang mampu menggali dan menemukan informasi yang bermanfaat dari sebuah *dataset* yang besar (Fulkerson et al., 1995). *Deep learning* yang merupakan salah satu cabang ilmu *Machine learning*, juga mulai banyak digunakan untuk mengimplementasikan IDS. Hal ini menjadi menarik bagi banyak peneliti karena algoritma *Deep learning* yang mencoba meniru cara berpikir manusia. Penulis (Liu & Lang, 2019) juga menyatakan bahwa *deep learning* cenderung berkinerja lebih baik daripada *machine learning* saat memproses *dataset* yang besar.

Beberapa penelitian telah menggunakan *deep learning* dalam mengembangkan model IDS. Sebagain contoh, penulis (Bhuvaneswari Amma et al., 2021) menggunakan metode Gated Recurrent Unit (GRU) yang merupakan pengembangan dari metode RNN (Chung, 2015). Metode GRU yang digunakan dikombinasikan dengan Stacked Autoencoder (SA). *Dataset* yang digunakan adalah dataset NSL-KDD. Hasil penelitian (Bhuvaneswari Amma et al., 2021) menunjukkan bahwa metode yang diusulkan dapat meningkatkan akurasi jika dibandingkan dengan metode lainnya. Selanjutnya, metode GRU memiliki durasi pelatihan yang lebih pendek dibandingkan LSTM dan RNN.

Penelitian yang dilakukan oleh (Borisenko et al., 2021) membandingkan metode Multilayer Perceptron (MLP) dan LSTM. Hasil penelitian menunjukkan bahwa LSTM memiliki kinerja yang lebih baik dari MLP dimana LSTM mampu mendeteksi jenis serangan yang tidak mampu dilakukan oleh MLP.

Metode Deep Neural Network (DNN) digunakan oleh penulis (Lohiya & Thakkar, 2021). AntiRectifier layer yang dikombinasikan dengan metode ini terbukti meningkatkan akurasi. Selain itu, penggunaan Dropout layer juga dapat meningkatkan akurasi DNN jika dibandingkan dengan DNN tanpa Dropout layer.

Selanjutnya penelitian (Khan, 2021) menggunakan RNN sebagai *hidden layer* dan Convolutional Neural Network (CNN) sebagai *input layer*. Penulis (Khan, 2021) menerapkan berbagai fungsi optimasi seperti Stochastic Gradient Descent (SGD), Adam, RMSProp dan Adamax, berbagai *learning rate*, dalam upaya mencari model dengan hasil tertinggi. Riset mereka menghasilkan performa yang cukup menjanjikan, yakni mencapai F1 Score sebesar 97,6%.

Penelitian yang dilakukan oleh penulis (Al-Emadi et al., 2020) membandingkan kinerja metode LSTM, GRU dan CNN. Hasil eksperimen (Al-Emadi et al., 2020) menunjukkan bahwa CNN, LSTM dan GRU mencapai kinerja optimal pada jumlah epoch masing-masing 598, 987, dan 879. Akurasi metode CNN, LSTM dan GRU masing-masing mencapai 97,01%, 81,60% dan 50,25%. Dataset yang digunakan dalam penelitian ini adalah NSL-KDD.

Penulis (Nayyar et al., 2020) mengusulkan LSTM dan Dense layer untuk digunakan sebagai *hidden layer*. Mereka menggunakan 5 LSTM layer dan dilanjutkan dengan 7 Dense layer. Tanh diterapkan sebagai fungsi aktivasi pada LSTM layer dan Relu sebagai fungsi aktivasi pada Dense layer. Akurasi model cukup menjanjikan yaitu mencapai 96,7%. Penulis (Nayyar et al., 2020) menyarankan agar peneliti selanjutnya melakukan variasi lapisan LSTM dengan metode lain.

Penulis (Andalib & Vakili, 2020) membandingkan metode GRU, CNN dan Random Forest (RF). Untuk menyelesaikan masalah *vanishing gradient* pada GRU, peneliti mengimplementasikan fungsi aktivasi Leaky Relu dan Exponential Linear Unit (ELU). Model dilatih untuk 100 iterasi. Akurasi yang dihasilkan model sangat menjanjikan yaitu mencapai 99,76% pada metode GRU.

Penelitian yang dilakukan oleh (Kim et al., 2020) menggunakan LSTM sebagai *hidden layer* dan CNN sebagai *input layer*. Pada penelitian ini, metode LSTM juga dikombinasikan dengan Bidirectional layer. Setelah data diolah oleh LSTM layer, data tersebut kemudian dimasukkan ke dalam 8 Dense layer yang juga berfungsi sebagai *hidden layer*. Penulis (Kim et al., 2020) menggunakan Adam

sebagai fungsi optimasi. Akurasi yang dihasilkan oleh model tersebut cukup tinggi, masing-masing mencapai 98,07%, 91,52% dan 93,00% pada dataset KF-ISAC, CSIC-2010 dan CIC IDS 2017.

Studi oleh penulis (Priyadarshini & Barik, 2019) mengimplementasikan LSTM untuk mengembangkan model untuk melakukan mitigasi serangan DDoS pada fog environment. Penulis (Priyadarshini & Barik, 2019) bereksperimen menggunakan LSTM tanpa *hidden layer*, LSTM dengan 1 *hidden layer*, LSTM dengan 2 *hidden layer* dan LSTM dengan 3 *hidden layer*. Akurasi tertinggi dihasilkan oleh model yang menggunakan 2 *hidden layer* dengan aktivasi Tanh dan Sigmoid. Dropout layer pada model diimplementasikan dengan parameter 0.2 dan model mencapai akurasi 98,88%.

Penulis (Le et al., 2019) membandingkan metode RNN, LSTM dan GRU. Penelitian ini menggunakan NSL-KDD dan ISCX sebagai dataset. Hasil penelitian (Le et al., 2019) menunjukkan bahwa akurasi metode LSTM pada dataset NSL-KDD dan ISCX mencapai 92% dan 97,5%, sedangkan metode GRU mencapai 91,8% dan 97,08%. Metode RNN hanya mencapai 89,6% dan 94,75% pada dataset yang sama.

Penulis (Xu et al., 2018) mengeksplorasi metode LSTM dan GRU yang dikombinasikan dengan Bidirectional layer dan MLP. Performa yang dihasilkan oleh setiap kombinasi metode cukup menjanjikan dengan akurasi tertinggi yang dihasilkan oleh metode Bidirectional GRU yang dikombinasikan dengan MLP, baik pada dataset KDD99 maupun pada dataset NSL-KDD.

Terdapat beberapa metode *Deep learning* yang dapat digunakan untuk menyelesaikan permasalahan klasifikasi, diantaranya DBN, DNN, CNN dan RNN (Liu & Lang, 2019). Metode CNN merupakan algoritma yang paling banyak digunakan dalam melakukan tugas-tugas *computer vision* (Xie & Yuille, 2017). Metode DBN telah banyak digunakan untuk para peneliti dalam penelitian *intrusion detection* baik untuk trafik jaringan secara umum (Sohn, 2021) maupun di dalam jaringan *Internet of Things* (Otoum & Nayak, 2021). Begitu juga dengan DNN yang telah banyak diadopsi di ranah kecerdasan buatan (Kwon et al., 2019). Di dalam penelitian ini dipilih metode RNN karena mampu memahami data yang berupa *time-series* seperti trafik data di jaringan dengan baik (Madan & Sarathimangipudi, 2018).

Traditional RNN mempunyai permasalahan *vanishing and exploding gradients*, sehingga metode LSTM diusulkan oleh (Hochreiter & Schmidhuber, 1997) dan GRU diusulkan oleh (Chung et al., 2014). Berdasarkan hasil penelitian (Bhuvaneswari Amma et al., 2021), (Borisenko et al., 2021), (Al-Emadi et al., 2020), (Nayyar et al., 2020), (Andalib & Vakili, 2020), (Kim et al., 2020), (Priyadarshini & Barik, 2019), (Le et al., 2019) dan (Xu et al., 2018) menyatakan bahwa metode LSTM dan GRU mempunyai performa yang sangat baik saat digunakan dalam mengembangkan model IDS. Oleh karena itu, dalam penelitian ini digunakan kedua metode tersebut. Traditional RNN tidak digunakan lagi pada penelitian ini karena mempunyai permasalahan *vanishing and exploding gradient* serta performanya yang lebih rendah jika dibandingkan metode LSTM dan GRU berdasarkan hasil penelitian (Le et al., 2019).

Selanjutnya, penelitian ini juga menguji penggunaan Bidirectional layer terhadap peningkatan performa model yang menggunakan LSTM dan GRU layer. Hasil penelitian (Alex Graves & Schmidhuber, 2005) menyatakan bahwa metode Bidirectional LSTM lebih efektif dibandingkan metode *neural network* yang lain.

Dalam mengembangkan model *deep learning* terdapat banyak parameter yang dapat diatur, diantaranya adalah fungsi aktivasi, fungsi optimasi, dan *learning rate*. Oleh karena itu, dalam mengembangkan metode LSTM dan GRU juga dilakukan variasi parameter-parameter tersebut sehingga akan dibuat beberapa skenario eksperimen pada masing-masing metode dengan harapan bisa mendapatkan akurasi terbaik dari masing-masing metode. Akurasi tertinggi pada masing-masing metode ini kemudian dibandingkan. Metode dengan akurasi yang paling tinggi akan menjadi usulan dalam penelitian ini untuk meningkatkan akurasi *intrusion detection system* dan kemudian dibandingkan dengan penelitian sebelumnya.

1.2. Rumusan Masalah

Latar belakang di atas menghasilkan beberapa rumusan di bawah ini:

a. Metode apa dan dengan parameter bagaimana yang menghasilkan akurasi paling tinggi?
b. Apakah terjadi peningkatan akurasi dari metode yang diusulkan jika dibandingkan dengan penelitian sebelumnya?

1.3. Batasan Masalah

a. Dataset yang digunakan adalah CIC IDS 2017 secara keseluruhan yang mempunyai ukuran 841.3 MB setelah digabungkan. Dataset ini mempunyai 2.830.743 baris data.
b. Tahapan *preprocessing* yang dikenai pada dataset meliputi data *cleaning*.
c. Pengaruh tahapan *preprocessing* tidak diukur terhadap performa model.
d. Jumlah *hidden layer* menggunakan 1 layer.
e. Maksimal nilai *epoch* pada *training* tahapan pertama adalah 100 iterasi dan pada tahapan kedua adalah 1000 iterasi.
f. Tipe layer diuji adalah LSTM dan GRU.
g. Penggunaan Bidirectional layer pada model diuji untuk melihat pengaruhnya terhadap performa model.
h. *Learning rate* yang akan digunakan yaitu 0.001 dan 0.0001.
i. Fungsi aktivasi yang akan diujikan yaitu Relu, PRelu, LRelu, Softmax dan Tanh.
j. Optimasi pada saat training akan diuji menggunakan Adam dan Stochastic Gradient Descent (SGD).
k. Regularization dilakukan menggunakan gabungan antara Dropout dan L2.
l. Platform penelitian menggunakan Google Colaboratory.

1.4. Tujuan Penelitian

a. Untuk mengetahui metode beserta parameternya untuk menghasilkan akurasi paling tinggi.

b. Untuk mengetahui apakah terjadi peningkatan akurasi dari metode yang diusulkan jika dibandingkan dengan penelitian sebelumnya.

1.5. Manfaat Penelitian

a. Menjadi referensi bagi berbagai praktisi jaringan yang ingin mengimplementasikan IDS berbasis *deep learning* untuk mengamankan jaringan di lingkungan kerjanya.
b. Berkontribusi secara ilmiah terhadap pengembangan penelitian di bidang keamanan jaringan, khususnya IDS berbasis *deep learning*.
c. Menjadi referensi dalam penelitian IDS berbasis *deep learning*.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Beberapa penelitian telah menggunakan *deep learning* dalam mengembangkan model IDS. Sebagain contoh, penulis (Bhuvaneswari Amma et al., 2021) menggunakan metode Gated Recurrent Unit (GRU) yang merupakan pengembangan dari metode RNN (Chung, 2015). Metode GRU yang digunakan dikombinasikan dengan Stacked Autoencoder (SA). Ekstraksi fitur dari data latih dilakukan oleh SA, yang kemudian diproses oleh GRU yang menjadi lapisan tersembunyi. *Dataset* yang digunakan adalah *dataset* NSL-KDD. Sebagai perbandingan, penulis (Bhuvaneswari Amma et al., 2021) juga menggunakan metode Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) dan GRU. Hasil penelitian mereka menunjukkan bahwa metode yang diusulkan dapat meningkatkan akurasi jika dibandingkan dengan ketiga metode lainnya. Selanjutnya, metode GRU memiliki durasi pelatihan yang lebih pendek dibandingkan LSTM dan RNN.

Penelitian yang dilakukan oleh (Borisenko et al., 2021) membandingkan metode Multilayer Perceptron (MLP) dan LSTM. Hasil penelitian menunjukkan bahwa LSTM memiliki kinerja yang lebih baik dari MLP dimana LSTM mampu mendeteksi jenis serangan yang tidak mampu dilakukan oleh MLP. Penulis (Borisenko et al., 2021) kemudian menggabungkan kedua metode yang memiliki akurasi lebih tinggi dari metode LSTM.

Metode Deep Neural Network (DNN) digunakan oleh penulis (Lohiya & Thakkar, 2021). AntiRectifier layer yang dikombinasikan dengan metode yang diusulkan terbukti meningkatkan akurasi. Selain itu, penggunaan layer Dropout juga dapat meningkatkan akurasi DNN jika dibandingkan dengan DNN tanpa layer Dropout. Namun penelitian (Lohiya & Thakkar, 2021) tidak menggunakan L2 Regularization yang juga memiliki fungsi yang sama dengan layer Dropout, yaitu untuk mencegah model dari *overfitting*. Penelitian tersebut dilakukan dengan menggunakan pendekatan *binary classification*.

Selanjutnya penelitian (Khan, 2021) menggunakan RNN sebagai *hidden layer* dan Convolutional Neural Network (CNN) sebagai *input layer*. Penelitian (Khan, 2021) dilakukan dengan menggunakan *binary classification*. Penulis (Khan, 2021) menerapkan berbagai fungsi optimasi seperti Stochastic Gradient Descent (SGD), Adam, RMSProp dan Adamax, berbagai learning rate, dalam upaya mencari model dengan hasil tertinggi. Riset mereka menghasilkan performa yang cukup menjanjikan, yakni mencapai F1 Score sebesar 97,6%.

Penelitian yang dilakukan oleh penulis (Al-Emadi et al., 2020) membandingkan kinerja metode LSTM, GRU dan CNN. Setiap metode dilatih hingga 1000 epoch. Sama halnya dengan penelitian (Lohiya & Thakkar, 2021) dan (Khan, 2021), penelitian (Al-Emadi et al., 2020) juga menggunakan pendekatan *binary classification*. Hasil eksperimen (Al-Emadi et al., 2020) menunjukkan bahwa CNN, LSTM dan GRU mencapai kinerja optimal pada jumlah epoch masing-masing 598, 987, dan 879. Akurasi metode CNN, LSTM dan GRU masing-masing mencapai 97,01%, 81,60% dan 50,25%. Dataset yang digunakan dalam

penelitian ini adalah NSL-KDD. Penulis (Al-Emadi et al., 2020) menyebutkan bahwa peningkatan jumlah *hidden layer* akan meningkatkan durasi pelatihan dan penggunaan *resource* mesin, tetapi tidak akan meningkatkan kinerja model. Pada bagian akhir penulis (Al-Emadi et al., 2020) mengungkapkan bahwa metode CNN membutuhkan durasi pelatihan yang lebih lama dibandingkan dengan kedua model lainnya.

Penulis (Nayyar et al., 2020) mengusulkan LSTM dan Dense layer untuk digunakan sebagai *hidden layer*. Mereka menggunakan 5 LSTM layer dan dilanjutkan dengan 7 Dense layer. Tanh diterapkan sebagai fungsi aktivasi pada LSTM layer dan Relu sebagai fungsi aktivasi Dense layer. Penelitian (Nayyar et al., 2020) dilakukan dengan menggunakan pendekatan *binary classification* sehingga fungsi aktivasi Sigmoid diimplementasikan pada *output layer*. Akurasi model cukup menjanjikan yaitu mencapai 96,7%. Penulis (Nayyar et al., 2020) menyarankan agar peneliti selanjutnya melakukan variasi lapisan LSTM dengan metode lain.

Penulis (Andalib & Vakili, 2020) membandingkan metode GRU, CNN dan Random Forest (RF). *Output* dari ketiga metode tersebut menjadi masukan bagi *output layer* dalam menentukan apakah suatu trafik merupakan trafik *benign* atau trafik serangan dengan *loss function binary cross-entropy*. Untuk menyelesaikan masalah *vanishing gradient* pada GRU, peneliti (Andalib & Vakili, 2020) mengimplementasikan fungsi aktivasi Leaky Relu dan Exponential Linear Unit (ELU). Model dilatih untuk 100 epoch. Penulis (Andalib & Vakili, 2020) menyatakan bahwa fungsi optimasi Adam menghasilkan kinerja yang lebih baik

dibandingkan dengan SGD, RMSProp, Adagrad dan Adadelta. Akurasi yang dihasilkan model sangat menjanjikan yaitu mencapai 99,76% pada metode GRU.

Penelitian yang dilakukan oleh (Kim et al., 2020) menggunakan LSTM sebagai *hidden layer* dan CNN sebagai *input layer*. Metode ini mirip dengan yang diusulkan oleh (Khan, 2021) dimana LSTM merupakan perbaikan dari metode RNN (Hochreiter & Urgen Schmidhuber, 1997). Pada penelitian ini, metode LSTM juga dikombinasikan dengan Bidirectional layer. Setelah data diolah oleh LSTM layer, data tersebut kemudian dimasukkan ke dalam 8 Dense layer yang juga berfungsi sebagai *hidden layer*. Untuk menghindari *overfitting*, penulis (Kim et al., 2020) menambahkan Dropout layer setelah setiap Dense layer yang diikuti oleh Leaky Relu layer. Pada layer output, Dense layer diimplementasikan dengan fungsi aktivasi Sigmoid karena studi dilakukan dengan *binary classification*. Penulis (Kim et al., 2020) menggunakan Adam sebagai fungsi optimasi. Akurasi yang dihasilkan oleh model tersebut cukup tinggi, masing-masing mencapai 98,07%, 91,52% dan 93,00% pada dataset KF-ISAC, CSIC-2010 dan CIC IDS 2017.

Studi oleh penulis (Priyadarshini & Barik, 2019) mengimplementasikan LSTM untuk mengembangkan model untuk melakukan mitigasi serangan DDoS pada *fog environment*. Penulis (Priyadarshini & Barik, 2019) bereksperimen menggunakan LSTM tanpa *hidden layer*, LSTM dengan 1 *hidden layer*, LSTM dengan 2 *hidden layer* dan LSTM dengan 3 *hidden layer*. Dropout layer juga dikonfigurasi di dalam arsitektur. Akurasi tertinggi dihasilkan oleh model yang menggunakan 2 *hidden layer* dengan aktivasi Tanh dan Sigmoid. Dropout layer

pada model diimplementasikan dengan parameter 0.2 dan model mencapai akurasi 98,88%.

Penulis (Le et al., 2019) membandingkan metode RNN, LSTM dan GRU. Penelitian ini menggunakan NSL-KDD dan ISCX sebagai dataset. Hasil penelitian (Le et al., 2019) menunjukkan bahwa akurasi metode LSTM pada dataset NSL-KDD dan ISCX mencapai 92% dan 97,5%, sedangkan metode GRU mencapai 91,8% dan 97,08%. Metode RNN hanya mencapai 89,6% dan 94,75% pada dataset yang sama. Metode LSTM menghasilkan hasil tertinggi dibandingkan metode lainnya dan metode RNN menghasilkan hasil terendah. Metode yang diusulkan memiliki hasil yang lebih tinggi dari hasil sebelumnya yang diusulkan oleh penulis lain. Hasil penelitian ini cukup menjanjikan dan perlu dilanjutkan pada dataset IDS terbaru khususnya dataset yang berukuran lebih besar seperti yang dinyatakan oleh penulis (Le et al., 2019) pada bagian kesimpulan.

Penulis (Xu et al., 2018) mengeksplorasi metode LSTM dan GRU yang dikombinasikan dengan Bidirectional layer dan MLP. Performa yang dihasilkan oleh setiap kombinasi metode cukup menjanjikan dengan akurasi tertinggi yang dihasilkan oleh metode Bidirectional GRU yang dikombinasikan dengan MLP, baik pada dataset KDD99 maupun pada dataset NSL-KDD. Konfigurasi model untuk *batch size* menggunakan nilai default, yaitu 32, dan nilai *learning rate* 0,01, yaitu 10 kali lebih besar dari nilai *default*. Besarnya nilai *learning rate* ini tentunya akan mempercepat proses pelatihan dengan resiko *overfit* yang cukup tinggi.

2.2. Keaslian Penelitian

Tabel 2.1 Matriks literatur review dan posisi penelitian

Intrusion Detection System Berbasis Deep learning Dengan Metode Recurrent Neural Network

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	*SAGRU: A Stacked Autoencoder-Based Gated Recurrent Unit Approach to Intrusion Detection*	G. Bhuvaneswari Amma, S. Selvakumar, R. Leela Velusamy, Intelligent Data Engineering and Analytics, Advances in Intelligent Systems and Computing, 2021.	Menerapkan metode GRU dikombinasikan dengan Autoencoder menggunakan *dataset* NSL-KDD. Performa metode yang diusulkan dibandingkan dengan metode LSTM dan RNN.	Metode yang diusulkan dapat meningkatkan akurasi jika dibandingkan dengan ketiga metode lainnya. Selanjutnya, metode GRU memiliki durasi pelatihan yang lebih pendek dibandingkan LSTM dan RNN.	Peneliti menyarankan agar peneliti berikutnya menggunakan *dataset* yang lebih lengkap dan mendekati *real-world traffic*.	Metode GRU diusulkan dalam penelitian ini, namun tidak dikombinasikan dengan Stack Autoencoder. Selain itu, metode pembanding, yaitu LSTM juga digunakan. Dataset yang digunakan berbeda, dimana pada penelitian ini digunakan *dataset* CIC IDS 2017 yang mempunyai ukuran yang lebih besar dan tipe serangan yang lebih lengkap.
2	*Intrusion Detection Using Multilayer Perceptron and Neural Networks with Long Short-Term Memory*	B. B. Borisenko, S. D. Erokhin, A. S. Fadeev, I. D. Martishin, Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO), 2021	Membandingkan metode MLP dan LSTM menggunakan *dataset* CIC IDS 2018. LSTM yang digunakan mempunyai 128 *neuron* dan dilatih menggunakan fungsi optimasi RMSprop.	Metode MLP mempunyai performa yang lebih tinggi dari pada LSTM, dimana akurasi MLP mencapai 96% sedangkan akurasi LSTM mencapai 94%. Namun di sisi lain, LSTM mampu mendeteksi trafik yang tidak mampu dideteksi oleh MLP.	Akurasi model masih bisa ditingkatkan lagi. Penulis (Borisenko et al., 2021) tidak menjelaskan alasan atau dasar pemilihan fungsi aktivasi, optimasi dan *learning rate*.	Metode LSTM akan digunakan pada penelitian ini, namun dengan beberapa variasi. Penelitian ini akan mencoba fungsi aktivasi Adam yang menurut penelitian sebelumnya mempunyai performa yang lebih baik dari fungsi optimasi yang lain. Selain itu, pada penelitian ini dilakukan beberapa skenario eksperimen untuk menemukan konfigurasi terbaik pada model.

Tabel 2.1 Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
3	*Intrusion Detection Using Deep Neural Network with AntiRectifier Layer*	Ritika Lohiya dan Ankit Thakkar, Applied Soft Computing and Communication Networks, 2021.	Mengkombinasikan metode DNN dengan AntiRectifier layer untuk meningkatkan akurasi model IDS.	Penggunaan AntiRectifier layer yang dikombinasikan dengan DNN dapat menghasilkan akurasi yang lebih tinggi dibandingkan dengan model yang menggunakan metode yang sama tapi dikombinasikan dengan standar Dropout atau Gaussian Dropout ataupun saat dibandingkan dengan akurasi metode *machine learning*.	Penelitian tersebut tidak menggunakan L2 regularizer yang juga dapat menghindarkan model dari *overfit*. Peneliti juga tidak menjelaskan jumlah iterasi training model sehingga tidak diketahui apakah model sudah dilatih sampai optimal atau belum.	Penelitian ini menggunakan Dropout layer yang dikombinasikan dengan L2 regularizer untuk menjaga model dari overfit. Salah satu *dataset* yang digunakan oleh penelitian tersebut adalah CIC IDS 2017 yang merupakan dataset yang juga digunakan pada penelitian ini. Akurasi model sudah cukup tinggi, namun masih bisa ditingkatkan lagi.
4	*HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System*	Muhammad Ashfaq Khan, Processes, 2021	Menggunakan RNN sebagai hidden layer dan CNN sebagai input layer untuk membuat IDS model.	Metode yang diusulkan mampu mencapai performa yang menjanjikan, yaitu f1 score 97.75% pada *dataset* CIC IDS 2018. Penulis (Khan, 2021) menerapkan berbagai fungsi optimasi seperti Stochastic Gradient Descent (SGD), Adam, RMSProp dan Adamax, berbagai *learning rate*, dalam upaya mencari model dengan hasil tertinggi.	Penulis (Khan, 2021) tidak menjelaskan secara detail semua hasil penelitian yang dicoba dengan berbagai fungsi optimasi dan *learning rate*. Penulis (Khan, 2021) menyarankan untuk melakukan pengujian pada dataset yang lain.	Penelitian ini mirip dengan penelitian (Khan, 2021) dimana sama-sama melakukan *binary classification*, namun metode yang diusulkan berbeda. Fungsi optimasi yang digunakan pada penelitian ini adalah Adam dan SGD saja.

Tabel 2.1 Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
5	*Using Deep Learning Techniques for Network Intrusion Detection*	Sara Al-Emadi, Aisha Al-Mohannadi, Felwa Al-Senaid, IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), 2020	Membandingkan kinerja metode LSTM, GRU dan CNN. Setiap metode dilatih hingga 1000 *epoch*. *Dataset* yang digunakan dalam penelitian ini adalah NSL-KDD.	Hasil eksperimen mereka menunjukkan bahwa CNN, LSTM dan GRU mencapai kinerja optimal pada jumlah *epoch* masing-masing 598, 987, dan 879. Akurasi metode CNN, LSTM dan GRU masing-masing mencapai 97,01%, 81,60% dan 50,25%. Akurasi tertinggi dicapai oleh metode CNN.	Menguji performa metode *deep learning* dengan berbagai kombinasi metode *deep learning* dan berbagai *hyperparameter*.	Penelitian ini menggunakan jumlah iterasi training yang sama dengan penelitian (Al-Emadi et al., 2020), yaitu 1000 *epoch*, namun menggunakan *dataset* yang berbeda. Penelitian ini menguji beberapa kombinasi metode LSTM, GRU dan Bidirectional layer beserta berbagai *hyperparameter* seperti yang disarankan oleh penulis (Al-Emadi et al., 2020) pada bagian *Future Work*.
6	*Recurrent Neural Network Based Intrusion Detection System*	Sanchit Nayyar, Sneha Arora and Maninder Singh, International Conference on Communication and Signal Processing, 2020	Menggunakan LSTM dan Dense untuk *hidden layer* dalam mengembangkan model IDS. Tanh diterapkan sebagai fungsi aktivasi pada LSTM layer dan Relu sebagai fungsi aktivasi Dense layer.	Akurasi model yang dilatih menggunakan dataset CIC IDS 2017 cukup menjanjikan yaitu mencapai 96,7%.	Penulis (Nayyar et al., 2020) menyarankan agar peneliti selanjutnya melakukan variasi lapisan LSTM dengan metode lain.	Persamaan penelitian ini dengan penelitian (Nayyar et al., 2020) adalah sama-sama menggunakan Sigmoid pada *output layer* karena sama-sama melakukan *binary classification*. Jumlah LSTM layer yang digunakan oleh penulis (Nayyar et al., 2020) cukup banyak, yaitu 5 layer dan dilanjutkan dengan 7 Dense layer. Hal ini bertentangan dengan pernyataan (Al-Emadi et al., 2020). Pada penelitian ini tidak hanya fungsi aktivasi Tanh yang digunakan, tetapi ada juga fungsi aktivasi Softmax, Relu, Prelu, dan Leaky Relu yang performanya akan dibandingkan.

Tabel 2.1 Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
7	*An Autonomous Intrusion Detection System Using an Ensemble of Advanced Learners*	Amir Andalib, Vahid Tabataba Vakili, Iranian Conference on Electrical Engineering (ICEE), 2020	Membandingkan metode GRU, CNN dan Random Forest (RF) pada *dataset* NSL-KDD. Model dilatih hingga 100 *epoch*.	Akurasi yang dihasilkan model sangat menjanjikan yaitu mencapai 99,76% pada metode GRU.	Penulis (Andalib & Vakili, 2020) menyatakan bahwa fungsi optimasi Adam menghasilkan kinerja yang lebih baik dibandingkan dengan SGD, RMSProp, Adagrad dan Adadelta.	Pada penelitian ini digunakan fungsi optimasi Adam seperti yang disarankan oleh penulis (Andalib & Vakili, 2020). Jumlah iterasi yang sama diterapkan pada semua skenario penelitian untuk menentukan model terbaik dari setiap kombinasi metode. Model-model tersebut selanjutnya dilatih lagi hingga 1000 epoch. Penulis (Andalib & Vakili, 2020) menggunakan fungsi aktivasi Leaky Relu ELU sedangkan pada penelitian ini menggunakan Relu, Prelu, Leaky Relu, Tanh dan Softmax.
8	*AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection*	Aechan Kim, Mohyun Park, Dong Hoon Lee, Special Section on Scalable Deep Learning For Big Data, 2020	Menggunakan LSTM yang dikombinasikan dengan Bidirectional layer sebagai *hidden layer* dan CNN sebagai *input layer*. Selanjutnya data dimasukkan ke dalam 8 Dense layer yang juga berfungsi sebagai *hidden layer*.	Akurasi yang dihasilkan oleh model tersebut cukup tinggi, masing-masing mencapai 98,07%, 91,52% dan 93,00% pada dataset KF-ISAC, CSIC-2010 dan CIC IDS 2017.	Akurasi model pada dataset CIC IDS 2017 masih dapat ditingkatkan lagi.	Penelitian ini dan penelitian penulis (Kim et al., 2020) sama-sama menggunakan fungsi optimasi Adam dan fungsi aktivasi Sigmoid pada output layer (Dense). Namun penulis (Kim et al., 2020) hanya menguji fungsi aktivasi Leaky Relu pada model, sedangkan pada penelitian ini digunakan juga fungsi aktivasi lain.

Tabel 2.1 Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
9	*A deep learning based intelligent framework to mitigate DDoS attack in fog environment*	Rojalina Priyadarshini, Rabindra Kumar Barik, Journal of King Saud University - Computer and Information Sciences, 2019	Membandingkan performa LSTM tanpa *hidden layer*, LSTM dengan 1 *hidden layer*, LSTM dengan 2 *hidden layer* dan LSTM dengan 3 *hidden layer*.	Training dilakukan pada dataset ISCX 2012 IDS dan diuji menggunakan *dataset* Hogzilla. Akurasi tertinggi dihasilkan oleh model yang menggunakan 2 *hidden layer* dengan aktivasi Tanh dan Sigmoid. Dropout layer pada model diimplementasikan dengan parameter 0.2 dan model mencapai akurasi 98,88%.	Studi dilakukan pada *dataset* yang sudah lama dan beberapa tipe serangan tidak ada di dalam *dataset* tersebut.	Penelitian ini menggunakan *dataset* CIC IDS 2017 yang merupakan dataset yang lebih baru. Persamaannya adalah penelitian ini juga mengeksplorasi metode LSTM dan menggunakan Dropout layer, namun dengan nilai parameter yang berbeda.
10	*Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks*	Thi-Thu-Huong Le, Yongsu Kim, Howon Kim, Applied Sciences, 2019	Membandingkan metode RNN, LSTM dan GRU dalam membuat IDS model menggunakan *dataset* NSL-KDD dan ISCX.	Hasil penelitian mereka menunjukkan bahwa akurasi metode LSTM pada dataset NSL-KDD dan ISCX mencapai 92% dan 97,5%, sedangkan metode GRU mencapai 91,8% dan 97,08%. Metode RNN hanya mencapai 89,6% dan 94,75% pada dataset yang sama.	Penulis (Le et al., 2019) menyatakan bahwa metode ini menjanjikan untuk dilatih pada data yang lebih besar.	Pada penelitian ini metode dengan akurasi tertinggi, yaitu LSTM juga dilatih pada dataset CIC IDS 2017, yang berukuran lebih besar dari dataset NSL-KDD yang digunakan pada penelitian (Le et al., 2019). Namun pada implementasi metode LSTM pada penelitian ini dilakukan variasi.
11	*An Intrusion Detection System Using a Deep Neural Network with Gated Recurrent Units*	Congyuan Xu, Jizhong Shen, Xin Du, Fan Zhang, IEEE Access, 2018	Mengeksplorasi metode LSTM dan GRU yang dikombinasikan dengan Bidirectional layer dan MLP.	Performa yang dihasilkan oleh setiap kombinasi metode cukup menjanjikan dengan akurasi tertinggi yang dihasilkan oleh metode Bidirectional GRU yang dikombinasikan dengan MLP, baik pada *dataset* KDD99 maupun pada *dataset* NSL-KDD.	*Learning rate* yang digunakan cukup besar, yaitu 0.01 sehingga training hanya dilakukan sampai 20 iterasi.	Pada penelitian ini nilai *learning rate* yang lebih kecil digunakan. Penulis (Xu et al., 2018) menggunakan nilai *default batch size* dalam melatih model yang juga dilakukan pada penelitian ini.

2.3. Landasan Teori

2.3.1. Intrusion Detection System

Intrusion Detection System (IDS) pertama kali diusulkan pada tahun 1980 (Anderson, 1980). Selama kurang lebih 40 tahun perkembangan IDS memunculkan berbagai produk IDS. Selama perkembangannya, ada berbagai macam masalah yang muncul, yaitu tingginya deteksi yang tidak tepat sasaran (*false alarm*), seperti mengirimkan peringatan (*alert*) saat tidak ada trafik yang berbahaya. Hal ini akan menambah beban kerja analis keamanan jaringan. Jika analis keamanan jaringan terus menerus mendapatkan *false alarm alert*, maka ini memungkinkan serangan yang sebenarnya menyusup pada salah satu *false alarm* tersebut. Oleh karena itu, banyak penelitian mengenai IDS fokus untuk mengurangi jumlah false alarm dan meningkatkan kemampuan deteksi trafik yang berbahaya. Di sisi lain, IDS tradisional tidak mampu mendeteksi serangan yang tidak dikenali. Perubahan kondisi dan lingkungan jaringan yang sangat cepat dan kemunculan berbagai teknologi baru di jaringan juga memunculkan berbagai jenis serangan baru. Oleh karena itu, menjadi urgent untuk mengembangkan IDS yang dapat mendeteksi serangan yang bahkan tidak dikenalinya (*unknown attack*).

Untuk menangani hal tersebut, banyak peneliti yang mulai menggunakan Machine Learning sebagai metode untuk membangun IDS. Machine Learning merupakan salah satu teknik kecerdasan buatan yang mampu menggali dan menemukan informasi yang bermanfaat dari sebuah dataset yang besar (Michie, dkk, 2009). Sebuah IDS yang yang berbasis Machine Learning dapat menjadi IDS yang baik dengan tingkat akurasi deteksi yang tinggi jika tersedia dataset yang

cukup dan model Machine Learning yang dibuat menggunakan metode yang tepat. Machine Learning mudah dipelajari banyak orang karena tidak tergantung pada satu bidang ilmu secara utuh. Selanjutnya, Deep Learning mempunyai performa yang lebih dibandingkan dengan metode Machine Learning lain dalam mengolah bigdata. Selain itu, Deep Learning dapat mempelajari fitur-fitur data langsung dari data yang berbentuk *raw* (data mentah). Perbedaan karakteristik antara Machine Learning dan Deep Learning adalah mengenai bagaimana keduanya melakukan kalkulasi. Deep Learning akan membuat beberapa *layer* yang tersembunyi dalam pemrosesannya. Sedangkan model Machine Learning tradisional seperti Support Vector Machine (SVM) ataupun k-nearest neighbors (KNN) hanya mempunyai satu *layer* pemrosesan saja. Oleh karena itu, model Machine Learning tradisional sering juga disebut *shallow model* (dangkal).

Secara umum, berdasarkan metode deteksinya, IDS diklasifikasikan menjadi dua macam, yaitu *Anomaly-based* IDS dan *Misuse/Signature-based* IDS (Aghdam & Kabiri, 2016). *Anomaly-based* IDS akan melakukan deteksi dengan cara membandingkan sebuah trafik dengan trafik-trafik normal sebelumnya. Jika sebuah trafik tampak berbeda dari biasanya (*anomaly*), maka IDS akan mengirimkan peringatan (*alert*) kepada administrator jaringan. Sedangkan *Signature-based* IDS akan membandingkan setiap trafik dengan *database signature* serangan yang sudah dipersiapkan pada IDS tersebut. Jika sebuah trafik mempunyai ciri-ciri yang mirip dengan salah satu atau lebih *entry database signature*, maka IDS akan mengirimkan *alert* kepada administrator. Kelebihan dari *Signature-based IDS* adalah rendahnya false alarm yang dimunculkan dan detailnya laporan yang

diberikan, jika pola trafik ada di dalam database. Jika pola *malicious traffic* tidak ada di dalam database IDS, maka IDS tidak akan melaporkan apapun kepada administrator. IDS jenis ini juga tidak bisa mendeteksi jenis/pola *malicious traffic* baru sehingga administrator harus dapat terus mengupdate database IDS. Kekurangan ini coba ditutupi oleh *anomaly-based* IDS. Yang paling penting dalam menyiapkan *anomaly-based* IDS adalah dengan mendefinisikan database trafik normal dengan detail, sehingga *anomaly-based* IDS akan dapat mendeteksi pola-pola trafik yang anomali. Jika muncul komunikasi di jaringan dengan berbagai protokol dan layanan yang baru dan tidak ada di dalam database IDS, maka hal ini akan memunculkan *false alarm*. Selain itu, IDS jenis ini tidak dapat memberikan laporan secara tepat dan detail kepada administrator karena memang pola trafik ini tidak di kenali oleh IDS.

Berdasarkan sumber datanya (*data source*), IDS juga dapat diklasifikasikan menjadi 2, yaitu *Host-based* IDS dan *Network-based* IDS (Heberlein et al., 1989). *Host-based* IDS diimplementasikan pada sebuah *host* tertentu yang fokus mendeteksi trafik berbahaya yang masuk ke *host* tersebut saja. IDS jenis ini dapat mendeteksi dengan detail karena IDS jenis ini dapat memonitor setiap aktifitas yang mencurigakan di dalam sistem baik pada *files*, *programs* dan *ports*. Berbeda dengan *Network-based* IDS yang diletakkan di perangkat jaringan atau pada *host* yang diposisikan agar dapat mengakses setiap aktifitas yang ada di jaringan. IDS jenis ini fokus melakukan pengecekan pada setiap aktivitas-aktivitas yang ada di jaringan, biasanya dengan mempertimbangkan identitas dalam komunikasi jaringan. *Host-based* IDS akan mengakses Log sebagai sumber data. Dalam

pemrosesannya menggunakan Deep Learning, *Host-based* IDS akan menggunakan fitur-fitur pada Log dan kemudian menggunakan teks analisis untuk mengolahnya. Pada *Network-based* IDS, data biasanya dibedakan menjadi 3, yaitu *packet data*, *flow data* dan *session data*.

2.3.2. Data Preprocessing

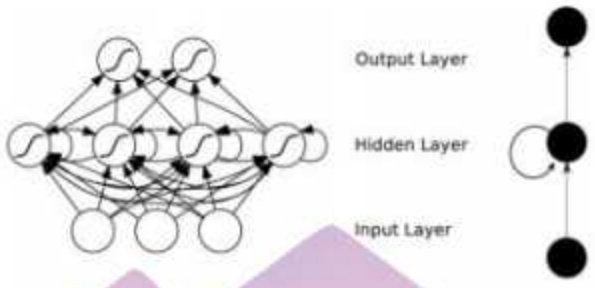
Data preprosessing merupakan tahap yang penting, tidak hanya untuk *deep learning*, tetapi juga pada berbagai kegiatan yang melibatkan data yang banyak. Ada beberapa hal yang biasanya melatarbelakangi upaya *data preprocessing*, diantaranya data yang tidak lengkap, data yang tidak konsisten, nilai yang kosong atau rusak dan data yang tidak seimbang *(imbalanced data)*. Teknik *data preprosessing* seperti *data cleaning*, data *transformastion*, dan *data reduction* merupakan teknik yang terbukti untuk menyelesaikan masalah yang disebutkan di atas. *Data cleaning* mendeteksi, memperbaiki atau bahkan menghapus *record* yang rusak atau tidak akurat dalam *dataset*. Hal-hal yang dilakukan dalam *data transformation* meliputi *smoothing*, *normalization*, *aggregation*, dan *data generalization* (Potluri et al., 2020).

Teknik *Data preprocessing* yang (Almalaq & Zhang, 2020) lakukan untuk metode *deep learning* meliputi tahap *data cleaning* dan *Data normalization*. *Data cleaning* penting dilakukan karena model berbasis *deep learning* sensitif terhadap sampel yang rusak dalam *dataset* agar performa *deep learning* yang lebih baik. Jika *dataset* tidak bersih, maka akan menghasilkan model yang tidak dapat diandalkan (Garcia et al, 2015). Teknik yang dilakukan dalam *data cleaning* termasuk

menghapus atau memperbaiki data dan entri yang hilang. *Data normalization* dilakukan untuk menghindari masalah ketidakseimbangan data dan membantu model untuk bekerja secara akurat.

2.3.3. RNN, LSTM dan GRU

Recurrent Neural Network (RNN) banyak digunakan karena kemampuannya untuk memproses dan memperoleh pengetahuan dari data sekuensial. Analisis video, pembuatan teks gambar, pemrosesan bahasa alami (NLP), dan analisis musik semuanya bergantung pada kemampuan *sequential neural network* (Hosseini et al., 2020). Tidak seperti *neural network* standar yang mengasumsikan independensi di antara titik data, RNN secara aktif menangkap dependensi sekuensial dan waktu antar data. Menurut (Yi & Tan, 2004) *neural network* dapat dibagi menjadi dua kelas besar. Kelas pertama mengandung Feedforward Neural Network (FNN) dan yang kedua mengandung Recurrent Neural Network (RNN). Perbedaan mendasarnya adalah adanya mekanisme *feedback* diantara neural pada RNN. Pada RNN minimal ada satu koneksi *feedback* antar neuron, baik dari satu neuron ke neuron yang lain atau terkoneksi ke dirinya sendiri. Memungkinkan bahwa setiap unit di dalam RNN menerima status sekarang dan status sebelumnya dari unit lain. RNN terdiri dari *Input Layer*, *Hidden Layer* dan *Output Layer*.



Gambar 2.1. Struktur Recurrent Neural *Network*

Gambar diadaptasi dari (Yin et al., 2017)

Menurut (El-Amir & Hamdy, 2020) RNN adalah termasuk model *neural network* yang sekuensial. RNN mengingat keputusannya yang dipengaruhi oleh apa yang telah dipelajari dari masa lalu. RNN terlihat sangat mirip dengan *neural network* tradisional kecuali status memori ditambahkan ke *neuron*. Dalam *neural network* tradisional, model menghasilkan keluaran dengan mengalikan masukan dengan bobot (*weight*) dan fungsi aktivasi. Dengan RNN, keluaran ini dikirim kembali ke dirinya sendiri beberapa kali. RNN dapat digambarkan dengan fungsi di bawah ini.

$$h_t = \sigma\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right) \quad (2)$$

$$y_t = W_{hy}h_t + b_y \quad (3)$$

Dimana:

x = Input Sequence

h = Hidder Vector Sequence

y = Output Vector Sequence

t = Sequence dari 1 sampai T

σ = Nonlinearity Function

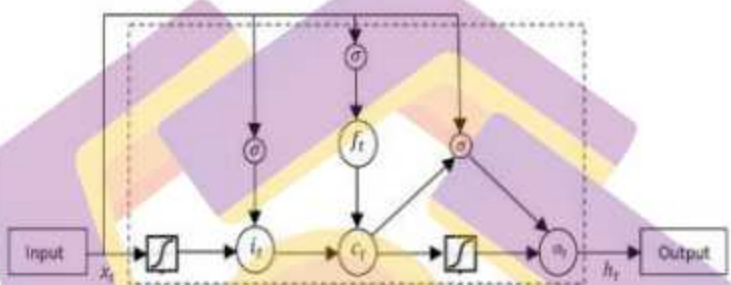
W = Weight Matrix

b = Bias

Recurrent Neural *Network* (RNN) banyak digunakan untuk pemodelan *time-series data*. Namun, RNN konvensional sering mengalami masalah *gradient vanishing/exploding* selama *training* model. Hal ini dapat mempengaruhi kinerja RNN pada pemodelan *long-term dependencies* dalam *time-series data*. Untuk mengatasi masalah ini, (Hochreiter & Schmidhuber, 1997) mengusulkan arsitektur baru, bernama Long Short-Term Memory (LSTM), yang mencoba menggunakan beberapa gerbang untuk mengontrol informasi untuk menyimpan atau membuang, sehingga dapat menangkap *long-term dependencies* dari urutan tersebut. Gerbang-gerbang ini diimplementasikan dengan Sigmoids yang berada pada range 0 sampai 1 (Vinita, 2020).

Komponen utama LSTM adalah *sequence input layer* dan LSTM layer. *Input layer* melakukan *import* urutan data ke dalam *network* sedangkan LSTM layer dapat mengingat *state network* (Yan, 2020). Sebuah *cell* pada LSTM dapat direpresentasikan dengan Gambar 2.2. LSTM konvensional hanya memperhatikan

informasi sekuensial dalam satu arah, yaitu arah maju. LSTM yang dapat mempertimbangkan konteks masa depan dan masa lalu disebut BiLSTM (Bidirectional LSTM) yang berisi lapisan maju dan lapisan mundur untuk memproses data sekuensial ke arah maju dan mundur (Chen et al., 2020).



Gambar 2.2. LSTM Cell (Yin et al., 2017)

Selain LSTM, ada juga bentuk modifikasi lain dari RNN yaitu Gated Recurrent Unit (GRU). GRU mempunyai struktur yang lebih sederhana dan lebih sedikit parameter jika dibandingkan dengan LSTM (Bansal et al., 2016). GRU pertama kali diusulkan oleh (Chung, 2015). *Memory cell* pada GRU tidak terpisah seperti pada LSTM (Zhang & Kabuka, 2018).

Dalam membangun sebuah *neural network*, hal berikutnya yang harus ditentukan setelah menentukan metode atau algoritma yang akan digunakan adalah jumlah layer dan jumlah *neuron* pada masing-masing layer. Sebuah *hidden layer* pada *neural network* mampu melakukan fungsi *universal approximation* dan melakukan tugas-tugas sederhana lain. Banyak *hidden layer* dibutuhkan pada kasus-kasus yang menggunakan data tidak terstruktur. Jumlah neuron pada *neural*

network merepresentasikan kemampunyai sebuah *neural network* untuk mengingat hasil *training*. Semakin banyak jumlah *neuron* pada *neural network*, maka semakin baik juga hasil akurasinya terhadap data *training*, namun akan memungkinkan terjadinya *overfitting* dimana model tidak mampu melakukan generalisasi terhadap *validation data*. Oleh karena itu, sebisa mungkin untuk menimimalisir jumlah *neuron* yang digunakan namun tanpa mengurangi hasil akurasi. Jumlah *neuron* pada masing-masing layer serta jumlah *hidden layer* juga berpengaruh pada lamanya durasi *training*. Semakin banyak jumlah layer dan neuron pada masing-masing layer akan menambah durasi *training* sebuah model.

Penulis (Vinita, 2020) menjelaskan bahwa terdapat beberapa layer yang dapat membantu untuk mendesain sebuah LSTM *network* yang efektif, diantaranya: Embedding Layer sebagai *input layer*, Dropout Layer sebagai teknik *regularization*, Dense Layer sebagai *output layer* dan *activation layer* sebagian fungsi untuk menghitung *output* dari sebuah *neuron*.

2.3.4. Regularization

Umumnya, model *deep learning* yang baik mempunyai tingkat kesalahan yang rendah pada data *training*. Namun terkadang sebuah model bisa menjadi sangat kompleks karena harus membuat fungsi yang dapat mengakomodir relasi semua fitur yang diekspresikan oleh data *training*. Peningkatan kompleksitas ini mungkin memiliki dua konsekuensi negatif. Pertama, model yang kompleks mungkin membutuhkan banyak waktu untuk dieksekusi. Kedua, model yang kompleks dapat mencapai performa yang sangat baik pada data *training*, tetapi

berperforma cukup buruk pada data validasi. Ini karena model tersebut mampu membuat hubungan antara banyak parameter dalam konteks pelatihan tertentu, tetapi hubungan ini sebenarnya tidak ada dalam konteks yang lebih umum. Hal ini menyebabkan model kehilangan kemampuannya untuk melakukan generalisasi. Gejala ini dikenal dengan istilah *overfitting* (Gulli, dkk, 2019).

Dalam bukunya (Calin, 2020) menguraikan bahwa model akan melakukan *overfitting* terhadap data *training* jika parameter diizinkan untuk menggunakan nilai yang besar. Namun, jika parameter dibatasi untuk memiliki nilai yang ditentukan, maka hal ini akan menghindarkan model untuk melakukan *overfitting* pada data pelatihan. Oleh karena itu, nilai parameter harus dijaga agar tetap kecil di sekitar nilai nol. Untuk melakukan hal ini *regularization* tipe L1 atau L2 biasanya digunakan. Selain teknik *regularization*, teknik Dropout juga digunakan untuk menghindari *overfitting* (El-Amir & Hamdy, 2020).

2.3.5. Activation

Activation function merupakan hal yang krusial dalam melakukan *training* pada *deep neural network*. Dengan *activation function*, sebuah *neural network* dapat mempelajari data yang kompleks dan merepresentasikan pemetaan fungsi *non-linear* antara *input* dan *output*. Beberapa jenis *activation function* yaitu Sigmoid, Tanh, Relu, Leaky Relu, Parametric Relu, Maxout dan ELU (Narayanan et al., 2020). Sedangkan menurut (Smith, 2020) fungsi aktivasi pada *Deep learning* meliputi: Tanh, Softsign, Softplus, Sofmax, Sigmoid, Relu, Prelu, RationalTanh, LeakyRelu, Identity, HardTanh, Hardsigmoid, Elu dan Cube. Hal ini sejalan dengan

dokumentasi yang dibuat oleh Keras pada halaman ini (https://keras.io/api/layers/activations/). Untuk membatasi banyaknya jumlah skenario penelitian, maka pada penelitian ini hanya akan digunakan fungsi aktivasi yang umum digunakan saja yaitu Softmax, Tanh, Relu, Prelu dan Lrelu.

Softmax merupakan fungsi matematis yang melakukan pemrosesan *input* dan menghasilkan *output* pada *range* 0 sampai 1 dimana semua jumlah *output* dari Softmax adalah 1 (Wani, dkk, 2020). Fungsi Softmax direpresentasikan oleh penulis (Wani, dkk, 2020) dengan persamaan 4. Sedangkan, dokumentasi Keras merepresentasikan fungsi Softmax dengan persamaan 5 dimana x adalah *input tensor*.

$$H(y) = \sum_i y'_i \log(y_i) \quad (4)$$

$$\text{Output} = \exp(x) / \text{tf.reduce\_sum}(\exp(x)) \quad (5)$$

Hyperbolic Tangent (Tanh) adalah fungsi yang outputnya berapa pada *range* nilai -1 sampai 1. Kelebihan Tanh adalah dapat membedakan antara data negatif dan positif dan tidak sama dengan Softmax dan Relu yang *output*-nya berada pada range nilai 0 sampai 1. Dokumentasi Keras merepresentasikan fungsi Tanh dengan persamaan 6 dan persamaan 7 dimana x adalah *input* tensor.

$$\tanh(x) = \sinh(x)/\cosh(x) \quad (6)$$

$$\tanh(x) = ((\exp(x) - \exp(-x))/(\exp(x) + \exp(-x))) \quad (7)$$

(Wani, dkk, 2020) menguraikan bahwa Rectified Linear Units (Relu) merupakan fungsi yang dapat menghindarkan model dari permasalahan *vanishing gradient descent* karena turunannya (*derivative*) selalu 1 untuk positif. Output Relu menjadi 0 untuk input yang mempunyai nilai kurang dari 0. Fungsi aktivasi Relu dapat direpresentasikan dengan persamaan 8.

$$f(x) = x \text{ dan } f'(x) = 1 \text{ untuk } x > 0 \quad (8)$$

2.3.6. Optimization

Sebuah model pada *deep learning* tidak dilatih hanya sekali pada sebuah data training. Model akan dilatih berkali-kali agar dapat memperbaiki *weight* dan *bias* pada neuron di tiap-tiap layer. Perbaikan atau *update* ini dilakukan oleh fungsi optimisasi. Ada banyak fungsi optimisasi yang dapat digunakan, diantaranya yang diuraikan pada dokumentasi Keras (https://keras.io/api/optimizers/) adalah: Stochastic Gradient Descent (SGD), RMSprop, Adam, Adadelta, Adagrad, Adamax, Nadam dan Ftrl. Dalam penelitian ini akan digunakan dua fungsi optimisasi yang umum digunakan yaitu Adam dan SGD.

SGD melakukan iterasi pada setiap sampel *training* dan kemudian mengkalkulasi *loss* untuk meng-*update weight* (Moolayil, 2019). Menurut (Goodfellow, dkk, 2016) SGD merupakan pengembangan dari algoritma Gradient-Based Optimization. Dalam perspektif SGD, gradien dianggap sebagai ekspektasi. SGD mempunyai banyak kegunaan di luar konteks *deep learning* karena SGD merupakan cara yang *general* untuk melatih sebuah *model linear* dengan *dataset*

yang sangat besar. Estimasi dari sebuah *gradient* dapat diformulasikan pada persamaan 8. Berdasarkan dokumentasi Keras, aturan untuk *update weight* jika momentum = 0 pada sebuah *neuron* adalah w = w – (learning_rate * g) dimana g adalah *gradient*, sedangkan jika momentum > 0, maka w = w + velocity dimana velocity = momentum * velocity – learning_rate * g.

$$g = \frac{1}{m'} \nabla\theta \sum_{i=1}^{m'} L\left(x^{(i)}, y^{(i)}, \theta\right) \quad (9)$$

Untuk melakukan *update* pada setiap sampel *training*, parameter use_batch harus = 1 pada fungsi model training. Karena *weight* akan di-*update* pada setiap sampel, maka SGD akan sangat sibuk. Untuk mengurangi fluktuasi optimasi SGD yang sangat tinggi, pendekatan yang dapat digunakan adalah dengan mengurangi jumlah iterasi dengan membuat *minibatch* yang dapat menghitung rata-rata *loss* dari semua sampel dalam sebuah *batch* dan kemudian meng-*update batch* di akhir data pada setiap *batch*. Pendekatan ini banyak berhasil untuk memperhalus proses *training* (Moolayil, 2019).

Menurut (Goodfellow, dkk, 2016) Adam merupakan salah satu algoritma optimasi yang mengadaptasi *learning rate*. Adam merupakan singkatan dari Adaptive Moments. (Moolayil, 2019) mengutarakan bahwa Adam merupakan algoritma optimasi yang paling terkenal dan banyak digunakan dalam *deep learning*. Pada banyak kasus, para praktisi dapat memilih algoritma Adam untuk optimasi dan melupakan algoritma optimasi alternatif lain. Estimasi gradien pada Adam diformulasikan pada persamaan 10.

$$g = \frac{1}{m} \nabla\theta \quad \sum_{i=1} L\left(f(x^{(i)}; \theta) y^{(i)}\right) \tag{10}$$

Lebih lanjut (Goolfellow, dkk, 2016) menguraikan bahwa dalam Adam terjadi penggabungan momentum sebagai perkiraan momen orde pertama dari gradien (dengan pembobotan eksponensial). Penggunaan momentum yang dikombinasikan dengan penskalaan ulang tidak memiliki motivasi teoritis yang jelas. Koreksi bias akan dimasukkan oleh Adam ke perkiraan momen orde pertama dan momen orde dua untuk menjelaskan inisialisasi mereka di awal.

2.3.7. Learning Rate dan Epoch

Menurut (Gulli, dkk, 2019) dalam membuat sebuah model, ada beberapa parameter yang dapat dioptimasi seperti jumlah *hidden neuron, batch size* dan jumlah epoch. Parameter-parameter ini disebut *hyperparameter*. Selain itu, diidentifikasi juga *hyperparameter learning rate* (El-Amir & Hamdy, 2020). *Learning rate* pada neural *network* umumnya berada pada *range* 0.0 dan 1.0. Variasi nilai *learning rate* yang umum digunakan adalah 0.1, 0.01, 0.001 dan 0.0001. Penyesuaian *hyperparameter* adalah proses menemukan kombinasi optimal dari *hyperparameter* tersebut untuk meminimalkan *cost function*. Jika n jumlah *hyperparameter* ditentukan secara manual, maka pengembang dapat dikatakan menggunakan metode *brute force* pada semua kemungkinan dari n parameter.

Learning rate mendefinisikan seberapa besar langkah yang dapat dilakukan oleh algoritma optimasi untuk melakukan *update weight* (Moolayil, 2019). Nilai *default learning rate* pada *framework* Keras adalah 0.001 dimana nilai ini dianggap

nilai yang paling tepat untuk kasus kebanyakan. Semakin besar nilai *learning rate* maka akan semakin cepat pula nilai *weight* mendekati nilai sebenarnya. Namun, jika nilai *weight* sudah mencapai nilai optimum tetapi masih ada beberapa iterasi *training* (*epoch*) yang harus dilewati oleh model maka penyesuaian akan melewati batas yang seharusnya (nilai optimum). Jika nilai *learning rate* terlalu kecil, maka memungkinkan bahwa nilai w belum mencapai nilai terbaik, namun iterasi model pada data *training* telah habis (Perrotta, 2020). Oleh karena itu, pemilihan nilai pada *learning rate* dan epoch menjadi salah satu variabel penting dalam penelitian ini.

BAB III

METODE PENELITIAN

3.1. Jenis, Sifat, dan Pendekatan Penelitian

Penelitian akan dilaksanakan dengan pendekatan kuantitatif. Peneliti akan mendapatkan angka-angka hasil eksperimen menggunakan fungsi-fungsi yang sudah tersedia di dalam *framework*. Penelitian ini termasuk penelitian eksperimental dimana peneliti akan melakukan berkali-kali percobaan dalam penerapan algoritma. Dalam penelitian ada beberapa variabel yang dapat dikontrol sehingga penelitian ini bersifat kausal.

3.2. Metode Pengumpulan Data

Data yang akan digunakan pada penelitian ini adalah *dataset* CIC IDS 2017. *Dataset* merupakan hasil penelitian (Sharafaldin et al., 2018) dari Canadian Institute for Cybersecurity (CIC), New Brunswick University. Untuk dapat mengunduh dataset, maka pengguna harus mengisi beberapa form pendataan terlebih dahulu. Dataset dapat diunduh secara bebas pada link berikut : https://www.unb.ca/cic/datasets/ids-2017.html.

3.3. Metode Analisis Data

Framework Tensor Flow dan Keras digunakan dalam melakukan analisa data karena fungsi-fungsi yang terdapat dalam *framework* tersebut sangat lengkap

dan cocok untuk melaksanakan penelitian yang menggunakan *deep learning*. Selain itu, dalam melakukan penghitungan metric digunakan framework Scikit-learn. Platform eksperimen yang akan digunakan adalah Google Colaboratory. *Dataset* akan dikenai *preprocessing* berupa *data cleaning*.

3.4. Dataset

Dataset CIC IDS 2017 merupakan *dataset* yang dihasilkan dari penelitian (Sharafaldin et al., 2018) di Canadian Institute for Cybersecurity, University of New Brunswick. *Dataset* CICIDS2017 berisi data normal (*benign*) dan serangan paling mutakhir, yang berbentuk *real-world data* (berbentuk file PCAP). *Dataset* ini juga berisi hasil analisis trafik jaringan menggunakan CICFlowMeter dengan *flow-based* berlabel pada *timestamp, source IP, destination IP, source port, destination port, protocol* dan jenis serangan (berbentuk file CSV). *Dataset* ini dibuat dari trafik 25 user pada protocol HTTP, HTTPS, FTP, SSH dan email.

Data capturing dilakukan selama 5 hari (Senin sampai Jumat), yaitu pada tanggal 3-7 Juli 2019 sejak pukul 9.00 AM hingga 17.00 PM. *Dataset* pada hari Senin berisi data trafik benign. Trafik serangan di-*generate* pada pagi dan sore hari di hari Selasa sampai Jumat.

Dataset CIC IDS 2017 merupakan *dataset* terstruktur dan berekstensi CSV. *Dataset* ini dibuat dari data yang diemulasi yang terbagi menjadi 8 file yang dengan label seperti pada Tabel 3.1. Berdasarkan Tabel 3.1 dapat dilihat bahwa dataset CIC IDS 2017 mempunyai 15 class. Komposisi jumlah data berdasarkan label juga dapat dilihat pada Tabel 3.1 dimana data berlabel BENIGN menjadi yang terbanyak.

Tabel 3.1 Komposisi *Dataset* CIC IDS 2017

Nama File	**Ukuran File**	**Komposisi Data (baris)**
Monday_WorkingHours pcap_ISCX	168 MB	BENIGN (529918)
Tuesday_WorkingHours pcap_ISCX	128 MB	BENIGN (432074) FTP-Patator (7938) SSH-Patator (5897)
Wednesday_workingHours pcap_ISCX	214 MB	BENIGN (440031) DoS Hulk (231073) DoS GoldenEye (10293) DoS slowloris (5796) DoS Slowhttptest (5499) Heartbleed (11)
Thursday_WorkingHours Morning_WebAttacks pcap_ISCX	49,7 MB	BENIGN (168186) Web Attack - Brute Force (1507) Web Attack – XSS (652) Web Attack - Sql Injection (21)
Thursday_WorkingHours_Afternoon Infilteration pcap_ISCX	79,2 MB	BENIGN (288566) Infiltration (36)
Friday_WorkingHours_Morning pcap_ISCX	55,6 MB	BENIGN (189067) Bot (1966)
Friday_WorkingHours_Afternoon DDos pcap_ISCX	73,5 MB	DDoS (128027) BENIGN (97718)
Friday_WorkingHours_Afternoon PortScan pcap_ISCX	73,3 MB	PortScan (158930) BENIGN (127537)
Hasil Gabungan Semua File	841.3 MB	BENIGN (2273097) SSH-Patator (5897) FTP-Patator (7938) DoS Slowhttptest (5499) DoS slowloris (5796) DoS GoldenEye (10293) DoS Hulk (231073) DDoS (128027) Heartbleed (11) Web Attack - Sql Injection (21) Web Attack – XSS (652) Web Attack - Brute Force (1507) Infiltration (36) Bot (1966) PortScan (158930)

Cuplikan isi dataset CIC IDS 2017 dapat dilihat pada Gambar 3.1. Pada gambar tersebut tidak dapat ditampilkan semua kolom karena keterbatasan area

tulisan. Semua kolom dataset berisi angka kecuali pada kolom Label yang berisi nama jenis serangan. Gambar 3.2 merupakan cuplikan sebagian kolom dataset beserta labelnya.

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min	Bwd Packet Length Mean	Bwd Packet Length Std	Flow Bytes/s	Flow Packets/s
0	80	640	7	4	440	358	220	0	[illegible]	[illegible]	179	0	89.5	[illegible]	1.246875e+06	17187.500000
1	80	900	9	4	600	2944	300	0	66.666667	[illegible]	1472	0	736.0	[illegible]	3.937778e+06	14444.444440
2	80	1205	7	4	2776	2830	1388	0	396.571429	[illegible]	1415	0	707.5	[illegible]	4.652282e+06	9128.630705
3	80	511	7	4	452	370	226	0	64.571429	[illegible]	185	0	92.5	[illegible]	[illegible]	21526.418790
4	80	773	9	4	612	2944	306	0	68.000000	134.000017	1472	0	736.0	[illegible]	[illegible]	[illegible]

Gambar 3.1. Cuplikan isi dataset CIC IDS 2017

```
subset.head()
```

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Label
0	49188	4	2	0	12	0	6	BENIGN
1	49188	1	2	0	12	0	6	BENIGN
2	49188	1	2	0	12	0	6	BENIGN
3	49188	1	2	0	12	0	6	BENIGN
4	49486	3	2	0	12	0	6	BENIGN

Gambar 3.2. Cuplikan sebagian kolom dataset dengan Label

Dataset ini dipilih karena memiliki tipe serangan yang lengkap jika dibandingkan dengan *dataset* publik lain yang sudah banyak digunakan seperti KDD Cup 99 dan NSL-KDD (Thapa et al., 2020). Selain itu, dataset ini juga sesuai dengan *framework dataset* IDS yang diusulkan oleh (Gharib et al., 2017). Baik CIC-IDS-2017 dan CIC-IDS-2018 diterbitkan oleh penulis yang sama, tetapi CIC-IDS-2018 lebih khusus digunakan untuk lingkungan *cloud computing* (Leevy & Khoshgoftaar, 2020).

Data preprocessing yang dilakukan pada penelitian ini adalah *cleaning* terhadap *dataset* dengan cara menghapus baris-baris data yang memiliki nilai yang *null* (kosong). Selain itu juga dilakukan penghapusan pada kolom-kolom yang mempunyai nilai terlalu divergen. Dalam penelitian ini, dilakukan eksperimen dengan pendekatan *binary classification* seperti yang dilakukan oleh penulis (Lohiya & Thakkar, 2021), (Khan, 2021) dan (Al-Emadi et al., 2020). Semua label *benign* diubah menjadi 1 dan semua label jenis serangan diubah menjadi 0. Setelah proses cleaning dan pengubahan label pada dataset, komposisi dataset menjadi 2.272.688 baris data berlabel 1 dan 556.697 baris data berlabel 0.

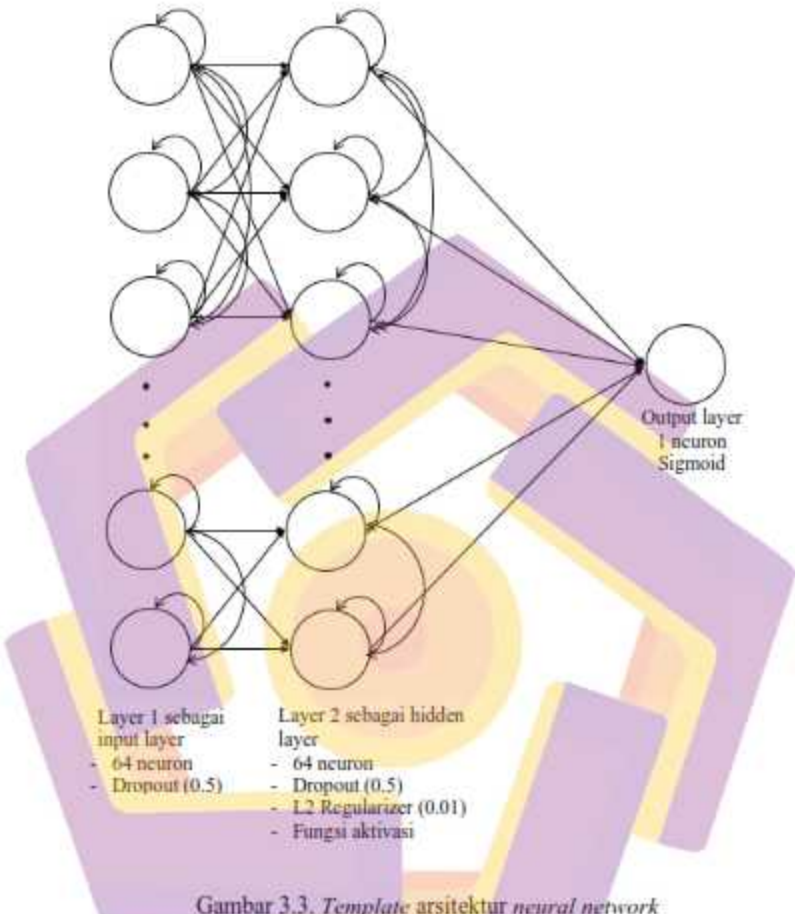
Dataset KDD-99 dan NSL-KDD dibagi oleh pembuat *dataset* menjadi data *training* dan data uji (Gamage & Samarabandu, 2020). Dataset CIC-IDS-2017 dibagi menjadi beberapa hari pengambilan data sehingga pemisahan data dilakukan oleh peneliti atau pengguna dataset. Model diuji menggunakan data yang berbeda dengan data yang digunakan untuk pelatihan dan validasi untuk menghindari *score bias* (Ripley, 2008) dengan pembagian persentase 60% untuk *training*, 20% untuk validasi dan 20% untuk pengujian. Validasi data digunakan untuk menguji model setiap kali model menyelesaikan satu iterasi *training*. Keluaran dari pengujian validasi data ini akan digunakan oleh model untuk meng-*update* nilai dari setiap *neuron*. Dalam penelitian ini, beberapa skenario eksperimen diuji sehingga agar distribusi data menghasilkan komposisi yang sama, ditentukan nilai *random_state* yang sama pada setiap proses distribusi data, yaitu 92.

3.5. Alur Penelitian

Penelitian ini terdiri dari beberapa tahap, yaitu: persiapan penelitian, data preprocessing, persiapan model, training dan validasi model, serta pengujian model. Pada tahap persiapan dilakukan *literature review* dan analisis terhadap dataset. Pada tahap preprocessing, dilakukan penggabungan semua file CSV dan kemudian dilakukan juga *data cleaning*.

Pada penelitian ini akan diuji berbagai skenario yang menggunakan variasi metode layer LSTM, GRU dan Bidirectional. LSTM dan GRU merupakan penyempurnaan jaringan syaraf tiruan dari RNN (Abdella et al., 2021). Langkah pertama yang dilakukan adalah menentukan template awal yang akan menjadi baseline untuk setiap skenario eksperimen nantinya. Gambar 3.3 mengilustrasikan template arsitektur *neural network* untuk setiap skenario penelitian. Setiap *neural network* yang diuji mengandung dua layer utama dari metode yang diuji dengan 64 *neuron* untuk setiap layer dan satu *layer output*. *Output layer* menggunakan Dense layer dengan fungsi aktivasi Sigmoid.

Penulis (Al-Emadi et al., 2020) menyatakan bahwa dalam hasil penelitiannya, peningkatan jumlah *hidden layer* pada arsitektur *neural network* untuk membuat model IDS tidak meningkatkan kinerja model yang mereka uji. Hasil penelitian penulis (Vinayakumar et al., 2019) juga menunjukkan bahwa 1 layer DNN menghasilkan akurasi yang lebih tinggi daripada DNN dengan lebih banyak layer yang diuji pada beberapa dataset. Oleh karena itu, pada penelitian ini hanya digunakan 1 hidden layer.



Gambar 3.3. *Template* arsitektur *neural network*

Dropout layer digunakan dengan nilai parameter 0,5 setelah setiap layer utama metode dengan harapan model yang dilatih tidak akan *overfit dataset* sesuai dengan rekomendasi penulis (Lohiya & Thakkar, 2021) dan penulis (Kim et al., 2020). Dalam arsitektur ini, L2 regularizer juga digunakan seperti yang direkomendasikan oleh penulis (El-Amir & Hamdy, 2020). Implementasi ini

dilakukan dengan menggunakan variabel recurrent_regularizer sebagai parameter pada neural network dengan nilai 0,01 sebagai parameter.

Pada output layer digunakan fungsi aktivasi Sigmoid karena penelitian ini dilakukan dengan pendekatan binary classification. Jumlah neuron yang dibutuhkan pada output layer hanya 1 karena fungsi aktivasi Sigmoid dapat menghasilkan 2 kemungkinan, yaitu 0 dan 1. Output ini sesuai dengan label dataset yang sudah diubah.

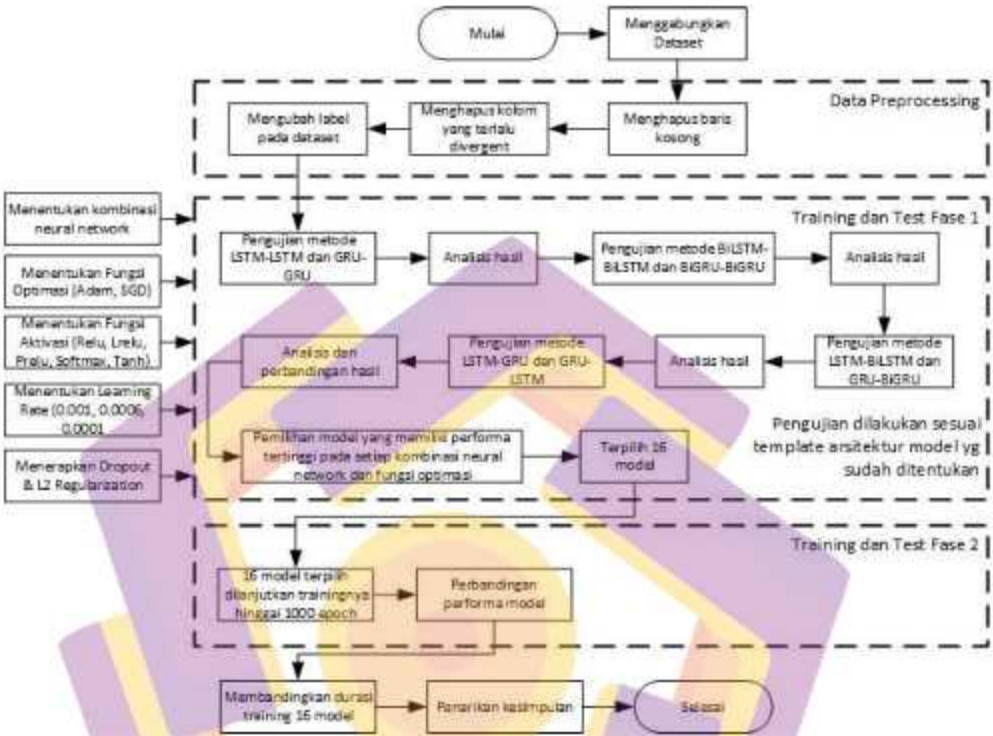
Setelah menentukan template arsitektur *neural network*, diteentukan variabel penelitian beserta nilai untuk setiap variabel. Variabel pertama adalah kombinasi *neural network* yang digunakan untuk layer pertama dan layer kedua. Ada 8 kombinasi jaringan saraf yang diuji seperti yang disajikan pada Tabel 3.2.

Dalam penelitian yang dilakukan oleh penulis (Khan, 2021) and (Kim et al., 2020), fungsi aktivasi Adam digunakan. Algoritma Adam diperkenalkan oleh penulis (Kingma & Ba, 2015). Penulis (Khan, 2021) menyatakan bahwa Adam mudah diimplementasikan dan implementasinya sendiri membutuhkan sedikit memori karena teknik komputasinya efisien. Penulis (Khan, 2021) juga menyebutkan bahwa algoritma ini cocok untuk menangani data yang besar. Selanjutnya penulis (Andalib & Vakili, 2020) menyatakan bahwa fungsi optimasi Adam memiliki kinerja yang lebih baik dibandingkan dengan fungsi optimasi lainnya, sehingga dalam penelitian ini digunakan fungsi optimasi Adam. Sebagai perbandingan, fungsi optimasi SGD yang juga digunakan. Kedua fungsi aktivasi tersebut banyak digunakan oleh para peneliti (Kingma & Ba, 2015).

Fungsi aktivasi merupakan komponen penting dalam melatih *neural network* (Goodfellow, dkk, 2016) sehingga fungsi aktivasi juga digunakan sebagai salah satu variabel dalam penelitian ini. Ada 5 fungsi aktivasi yang digunakan yaitu Relu, Leaky Relu, Prelu, Softmax dan Tanh. Variabel penelitian yang terakhir adalah *learning rate*. Dalam melatih model *deep learning*, *learning rate* menentukan seberapa besar perubahan nilai *weight* pada setiap *neuron* dalam model (Moolayil, 2019). *Learning rate* default pada *framework* Keras adalah 0,001. Nilai ini dianggap paling tepat karena tidak terlalu besar tetapi cukup cepat bagi model untuk mencapai kinerja yang optimal. Dengan demikian, nilai ini digunakan untuk variabel *learning rate*. Mengingat model dengan *learning rate* yang kecil akan memiliki peluang untuk mencapai performa yang lebih tinggi dalam jumlah iterasi *training* yang banyak, maka pada penelitian ini juga diterapkan *learning rate* sebesar 0,0001. Tabel 3.2 merangkum semua variabel eksperimen dengan nilainya. Alur penelitian ini dapat dilihat pada Gambar 3.4.

Tabel 3.1. Nilai untuk setiap variabel penelitian

NN Combination	Optimization Function	Activation Function	Learning Rate
LSTM-LSTM GRU-GRU BiLSTM-BiLSTM BiGRU-BiGRU LSTM-BiLSTM GRU-BiGRU GRU-LSTM LSTM-GRU	Adam SGD	Softmax Tanh Relu Prelu Lrelu	0.001 0.0001



Gambar 3.4. Alur penelitian

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1. Proses Pelaksanaan Training

Dalam penelitian ini, digunakan framework Keras, Tensor Flow dan Scikit Learn. File notebook dibuat untuk setiap skenario sehingga proses training dapat dijalankan secara paralel di beberapa komputer. Google Collaboratory digunakan sebagai *environment* untuk melakukan eksperimen. Training model dilakukan dalam 2 tahap. Pada tahap pertama, model dilatih dalam 100 iterasi seperti yang dilakukan dalam penelitian (Andalib & Vakili, 2020). Dari hasil *training* model pada setiap kombinasi *neural network* akan dipilih 1 model yang memiliki akurasi tertinggi dan menggunakan fungsi optimasi Adam dan 1 model yang menggunakan fungsi optimasi SGD. Pada dasarnya performa model yang menggunakan fungsi optimasi SGD lebih rendah dibandingkan yang menggunakan fungsi optimasi Adam di semua skenario *neural network*. Namun tetap dipilih 1 model yang memiliki akurasi tertinggi untuk dilatih lagi di tahap kedua. Pada tahap kedua, model dilatih hingga 1000 epoch seperti yang dilakukan oleh penulis (Al-Emadi et al., 2020) sehingga model dapat mencapai kinerja yang optimal.

Untuk setiap 100 iterasi *training*, model untuk setiap skenario disimpan menggunakan fungsi dari *framework* Keras. Selain itu, model dengan akurasi validasi tertinggi akan disimpan oleh fungsi ModelCheckPoint karena ada kemungkinan akurasi validasi model pada iterasi ke-100 lebih rendah dari akurasi validasi model pada beberapa epoch sebelumnya. Model dengan akurasi validasi

tertinggi diuji menggunakan data uji dan kemudian hasilnya disajikan dalam penelitian ini. Prosedur yang sama dilakukan pada pelatihan tahap kedua. Model yang telah dilatih hingga 1000 iterasi dan model dengan akurasi validasi tertinggi masing-masing disimpan dengan fungsi *framework* Keras dan ModelCheckPoint. Selama proses pelatihan tahap kedua, juga disimpan semua model untuk setiap 1000 iterasi.

Penulis (Lohiya & Thakkar, 2021), (Khan, 2021), (Al-Emadi et al., 2020), (Nayyar et al., 2020), (Andalib & Vakili, 2020), dan (Kim et al., 2020) melatih model mereka menggunakan pendekatan *binary classification*. Hal yang sama dilakukan juga dalam penelitian ini, semua model dilatih dengan pendekatan *binary classification*. Nilai *default* yang digunakan untuk parameter *batch size* seperti yang dilakukan oleh penulis (Xu et al., 2018). Dalam penelitian ini, digunakan akurasi dan f1 score sebagai *metric* pengukuran karena kedua *metric* ini merupakan *metric* yang paling sering digunakan oleh para peneliti sehingga nantinya hasil penelitian ini dapat dibandingkan dengan hasil penelitian lainnya.

Salah satu kendala terbesar dalam menggunakan Google Collaboratory sebagai *environment training* model adalah batasan waktu. Ada dua macam batasan waktu di sini, yang pertama adalah batasan waktu *idle*. Sampai laporan penelitian ini ditulis, belum ditemukan informasi resmi yang dikeluarkan oleh pihak Google mengenai hal tersebut. Namun berdasarkan informasi yang ada diberbagai forum, disebutkan bahwa jika semua notebook *idle* selama 90 menit, maka *notebook* akan terputus dari *runtime*. Padahal, proses *training* model yang dijalankan paling sedikit memakan waktu 12 menit periterasi (durasi *training* akan dibahas lebih detail

nanti). Sehingga untuk menjaga agar *notebook* tetap terhubung dengan *runtime*, maka pada penelitian secara berkala dibuka semua tab yang sedang proses menjalankan *training* dan menekan tombol Ctrl + S agar file *notebook* tidak dianggap *idle*. Batasan waktu kedua adalah batasan waktu penggunaan *runtime*. Berdasarkan informasi yang ada pada halaman FAQ Google Colaboratory, batasan waktu penggunaan sebuah *Virtual Machine (VM)* untuk menjalankan *runtime* adalah 12 jam. Untuk menangani hal tersebut, model hasil training dalam setiap 10 iterasi juga disimpan. Sehingga jika *training* model terputus pada 10 iterasi berikutnya karena sudah mencapai batasan waktu penggunaan *runtime*, maka proses training hanya mengulang beberapa iterasi saja dan kemudian dilanjutkan ke 10 iterasi berikutnya. Cara ini cukup manjur dilakukan dengan konsekuensi model yang disimpan akan menjadi sangat banyak. Oleh karena itu, penamaan file model yang disimpan menjadi sangat penting dan krusial.

4.2. Training dan Test Fase 1

Pada tahap pertama, percobaan dimulai dengan melatih kombinasi dasar layer LSTM dan GRU dimana layer LSTM digabungkan dengan layer LSTM dan layer GRU digabungkan dengan layer GRU. Kedua *neural network* ini dilatih dengan berbagai fungsi optimasi, fungsi aktivasi dan *learning rate* yang telah ditentukan pada Tabel 3.1. Performa kedua *neural network* ini disajikan pada Tabel 4.1. Pada model dengan kombinasi layer LSTM-LSTM menggunakan optimasi Adam fungsi, akurasi tertinggi dan f1 score dicapai oleh model yang menggunakan fungsi aktivasi Tanh dan *learning rate* 0,0001 sedangkan yang menggunakan SGD

dicapai oleh model yang menggunakan fungsi aktivasi Prelu dan *learning rate* 0,001. Pada kombinasi layer GRU-GRU dengan fungsi optimasi Adam, performa tertinggi dicapai oleh model yang menggunakan fungsi aktivasi Lrelu dan *learning rate* 0,0001. Sedangkan pada model dengan fungsi optimasi SGD, kinerja tertinggi dicapai oleh model yang menggunakan aktivasi Lrelu dan *learning rate* sebesar 0,001. Hasil keempat model tersebut ditandai dengan huruf tebal pada Tabel 4.1.

Tabel 4.1. Hasil pengujian kombinasi dasar layer LSTM dan GRU

Kombinasi NN layer	Optimasi	Fungsi Aktivasi	LR = 0.001		LR = 0.0001	
			Accuracy (%)	F1 Score (%)	Accuracy (%)	F1 Score (%)
LSTM-LSTM	Adam	Softmax	97.6562	97.6818	97.6904	97.7158
		Tanh	97.6780	97.7030	**97.8893**	**97.9194**
		Relu	97.6762	97.7015	97.7194	97.7446
		Lrelu	97.6677	97.6934	97.7146	97.7406
		Prelu	97.6659	97.6914	97.7146	97.7406
	SGD	Tanh	96.6509	96.6756	95.1813	95.1809
		Relu	96.8086	96.8373	96.0001	96.0360
		Lrelu	96.7121	96.7372	96.3642	96.4026
		Prelu	**96.8225**	**96.8417**	96.0240	96.0654
GRU-GRU	Adam	Softmax	97.5402	97.5666	97.7237	97.7486
		Tanh	97.5636	97.5887	97.7164	97.7417
		Lrelu	97.5711	97.5955	**97.7312**	**97.7563**
		Prelu	97.5206	97.5452	97.7243	97.7497
	SGD	Softmax	96.8695	96.8915	96.5038	96.5451
		Tanh	96.6939	96.7204	96.5002	96.5454
		Lrelu	**96.9899**	**97.0180**	96.5671	96.6035
		Prelu	96.7505	96.7832	96.2398	96.2785

Selanjutnya, layer LSTM dan GRU digabung dengan Bidirectional layer sehingga gabungan *neural network*-nya menjadi BiLSTM-BiLSTM dan BiGRU-BiGRU. Hasil training model pada kombinasi neural network ini disajikan pada Tabel 4.2. Pada kombinasi BiLSTM-BiLSTM dengan fungsi optimasi Adam, model yang mencapai performa tertinggi menggunakan fungsi aktivasi Tanh dan

learning rate 0,0001, sedangkan pada model dengan fungsi optimasi SGD kinerja tertinggi dicapai oleh model dengan fungsi aktivasi Lrelu dan *learning rate* 0,0001 juga. Pada kombinasi BiGRU-BiGRU dengan fungsi optimasi Adam, model dengan kinerja tertinggi dicapai oleh model dengan fungsi aktivasi Prelu dengan *learning rate* 0,0001, sedangkan pada kombinasi dengan optimasi SGD, kinerja tertinggi dicapai oleh model dengan fungsi aktivasi Tanh dan *learning rate* 0,001.

Tabel 4.2. Hasil pengujian kombinasi LSTM dan GRU dengan Bidirectional layer

Kombinasi NN layer	Optimasi	Fungsi Aktivasi	LR = 0.001		LR = 0.0001	
			Accuracy (%)	F1 Score (%)	Accuracy (%)	F1 Score (%)
BiLSTM-BiLSTM	Adam	Softmax	97.5959	97.6224	97.7210	97.7460
		Tanh	97.5324	97.5579	**97.7461**	**97.7701**
		Lrelu	97.6701	97.6957	97.7113	97.7366
		Prelu	97.6556	97.6810	97.7282	97.7524
	SGD	Softmax	96.8283	96.8623	96.2114	96.2502
		Tanh	**96.9597**	**96.9883**	96.4829	96.5248
		Lrelu	96.8913	96.9142	96.6442	96.6574
		Prelu	96.5925	96.6177	96.5380	96.5604
BiGRU-BiGRU	Adam	Softmax	97.5608	97.5847	**97.7579**	**97.7817**
		Tanh	97.5717	97.5973	97.7276	97.7530
		Lrelu	97.5157	97.5412	97.7264	97.7515
		Prelu	97.5554	97.5820	97.7101	97.7355
	SGD	Softmax	96.9309	96.9597	96.1990	96.2438
		Tanh	96.7895	96.8115	96.3776	96.4212
		Lrelu	**96.9391**	**96.9681**	96.3485	96.3801
		Prelu	96.8277	96.8537	95.8954	95.9219

Setelah menggabungkan LSTM dan GRU dengan Bidirectional layer pada layer pertama dan layer kedua, tidak didapatkan peningkatan kinerja model. Akurasi tertinggi yang dicapai oleh kombinasi LSTM-LSTM adalah 97,8893% sedangkan akurasi tertinggi yang dicapai oleh kombinasi BiLSTM-BiLSTM hanya mencapai 97,7264%. Di sisi lain, kombinasi GRU-GRU mencapai akurasi

97,7312%. Kombinasi BiGRU-BiGRU memberikan hasil yang sedikit lebih baik yaitu 97,8051 %.

Karena sifat Bidirectional layer yang akan mencoba mempelajari urutan data dalam dua arah (Schuster & Paliwal, 1997), model yang menggunakan Bidirectional layer akan memiliki durasi *training* yang lebih lama daripada model yang tidak. Penulis (A Graves & Schmidhuber, 2005) menyatakan hal yang sama dalam makalah mereka. Hasil eksperimen penulis (Imrana et al., 2021) juga mengungkapkan bahwa metode Bidirectional LSTM memiliki durasi *training* yang lebih lama daripada LSTM karena kompleksitas komputasinya.

Dengan mempertimbangkan hal ini, berikutnya dilatih kombinasi LSTM-BiLSTM dan GRU-BiGRU dengan harapan dapat mengurangi waktu training karena layer pertama tidak menggunakan Bidirectional layer namun masih mendapatkan kompleksitas komputasi dari Bidirectional layer pada model layer kedua. Hasilnya disajikan pada Tabel 4.3. Performa tertinggi dari kombinasi LSTM-BiLSTM yang menggunakan optimasi Adam dihasilkan oleh model dengan fungsi aktivasi Tanh dan learning rate 0,0001 sedangkan hasil tertinggi dari kombinasi yang menggunakan optimasi SGD dihasilkan oleh model dengan fungsi aktivasi Tanh dan *learning rate* 0,001. Hasil tertinggi dari kombinasi GRU-BiGRU yang menggunakan optimasi Adam dihasilkan oleh model dengan aktivasi Softmax dan *learning rate* 0,0001 sedangkan hasil tertinggi dari kombinasi yang menggunakan optimasi SGD dihasilkan oleh model dengan fungsi aktivasi Laeky Relu dan *learning rate* 0,001.

Tabel 4.3. Hasil pengujian neural network LSTM-BiLSTM dan GRU-BiGRU

Kombinasi NN layer	Optimasi	Fungsi Aktivasi	LR = 0.001		LR = 0.0001	
			Accuracy (%)	F1 Score (%)	Accuracy (%)	F1 Score (%)
LSTM-BiLSTM	Adam	Softmax	97.5959	97.6224	97.7210	97.7460
		Tanh	97.5324	97.5579	**97.7461**	**97.7701**
		Lrelu	97.6701	97.6957	97.7113	97.7366
		Prelu	97.6556	97.6810	97.7282	97.7524
	SGD	Softmax	96.8283	96.8623	96.2114	96.2502
		Tanh	**96.9597**	**96.9883**	96.4829	96.5248
		Lrelu	96.8913	96.9142	96.6442	96.6574
		Prelu	96.5925	96.6177	96.5380	96.5604
GRU-BiGRU	Adam	Softmax	97.5608	97.5847	**97.7579**	**97.7817**
		Tanh	97.5717	97.5973	97.7276	97.7530
		Lrelu	97.5157	97.5412	97.7264	97.7515
		Prelu	97.5554	97.5820	97.7101	97.7355
	SGD	Softmax	96.9309	96.9597	96.1990	96.2438
		Tanh	96.7895	96.8115	96.3776	96.4212
		Lrelu	**96.9391**	**96.9681**	96.3485	96.3801
		Prelu	96.8277	96.8537	95.8954	95.9219

Dari Tabel 4.3 terlihat bahwa kombinasi LSTM-BiLSTM menghasilkan akurasi yang sedikit lebih tinggi daripada kombinasi BiLSTM-BiLSTM, namun hasilnya masih di bawah hasil kombinasi LSTM-LSTM. Hal ini berbeda dengan hasil penelitian penulis (Imrana et al., 2021) yang menggunakan optimasi yang sama, bahkan pada hasil dengan *learning rate* yang sama pada Tabel 4.1 dan Tabel 4.2. Hal ini dapat terjadi karena ada perbedaan arsitektur *neural network* dan *dataset* yang digunakan. Di sisi lain, hasil yang berbeda dihasilkan oleh kombinasi GRU. Kombinasi GRU-BiGRU menghasilkan akurasi yang lebih rendah dibandingkan kombinasi BiGRU-BiGRU, namun masih lebih tinggi dari hasil kombinasi GRU-GRU.

LSTM layer memiliki akurasi yang lebih baik pada *dataset* ini ketika diimplementasikan tanpa Bidirectional layer. Sebaliknya, GRU layer memiliki akurasi yang lebih tinggi ketika diimplementasikan dengan Bidirectional layer.

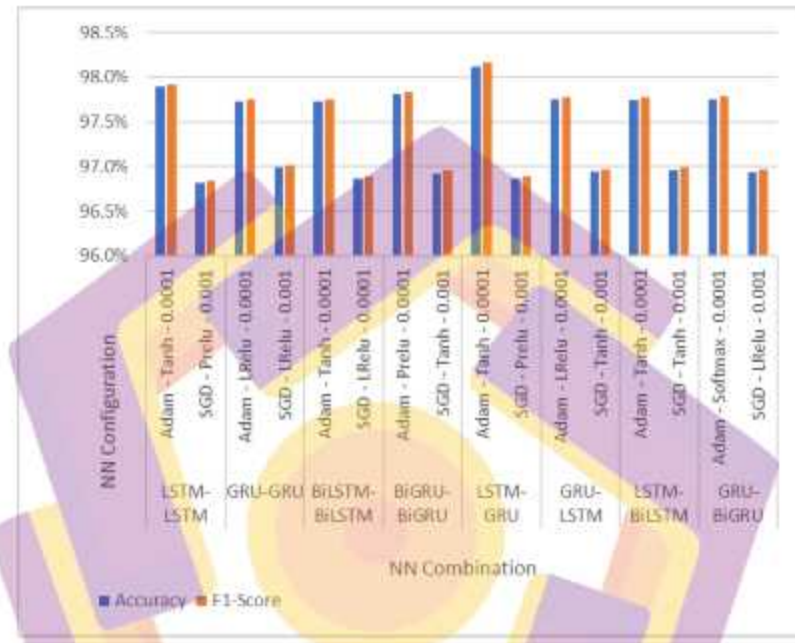
Fakta tentang GRU yang diperoleh dalam percobaan ini sesuai dengan hasil penelitian (Xu et al., 2018) yang mengusulkan Bidirectional GRU untuk membangun arsitektur *neural network*. Hasil ini masih akan dievaluasi pada tahap kedua nanti.

Tabel 4.4. Hasil pengujian neural network LSTM-GRU dan GRU-LSTM

Kombinasi NN layer	Optimasi	Fungsi Aktivasi	LR = 0.001		LR = 0.0001	
			Accuracy (%)	F1 Score (%)	Accuracy (%)	F1 Score (%)
LSTM-GRU	Adam	Softmax	97.6822	97.7075	97.7527	97.7771
		Tanh	97.6686	97.6941	**98.1087**	**98.1586**
		Lrelu	97.6737	97.6995	97.7497	97.7737
		Prelu	97.6895	97.7148	97.7309	97.7582
	SGD	Softmax	96.8026	96.8310	96.4705	96.5091
		Tanh	96.7687	96.7948	96.7926	96.8139
		Lrelu	96.8401	96.8680	96.5098	96.5497
		Prelu	**96.8619**	**96.8831**	96.3524	96.3866
GRU-LSTM	Adam	Softmax	97.5318	97.5570	97.7267	97.7506
		Tanh	97.5393	97.5633	97.7243	97.7494
		Relu	97.6014	97.6268	97.7297	97.7550
		Lrelu	97.5775	97.6046	**97.7527**	**97.7768**
		Prelu	97.5702	97.5949	97.7207	97.7458
	SGD	Tanh	**96.9451**	**96.9713**	95.7540	95.7727
		Relu	95.4746	95.5013	95.7519	95.7502
		Lrelu	96.8365	96.8639	95.8817	95.9199
		Prelu	96.9106	96.9430	95.7637	95.8075

Mengingat peningkatan kinerja masih kecil, penelitian dilanjutkan dengan menggabungkan lapisan dasar LSTM dan GRU sebagai metode gabungan *(hybrid)*. Kombinasi LSTM-GRU dan GRU-LSTM dilatih dan hasilnya disajikan pada Tabel 4.4. Kombinasi LSTM-GRU mencapai akurasi 98,1087%, lebih tinggi dari hasil semua kombinasi *neural network* yang disajikan sebelumnya. Hasil ini dicapai dengan model yang menggunakan fungsi optimasi Adam, aktivasi Tanh dan *learning rate* 0,0001. Sayangnya, hal yang sama tidak terjadi pada model dengan kombinasi GRU-LSTM. Akurasi tertinggi yang dihasilkan oleh model ini dengan

optimasi Adam, fungsi aktivasi Lrelu dan *learning rate* 0,0001 hanya mencapai 97,7527%.



Gambar 4.1. Perbandingan kinerja model terpilih dalam 100 iterasi training

Gambar 4.1 menunjukkan perbandingan kinerja antara model-model terbaik pada setiap kombinasi *neural network* baik menggunakan optimasi Adam maupun optimasi SGD. Sumbu X pada Gambar 4.1 menjelaskan kombinasi *neural network*, optimasi, fungsi aktivasi dan *learning rate* dari masing-masing model. Sumbu Y menampilkan performa model baik dalam akurasi maupun f1 score yang masing-masing berwarna biru dan jingga. Dari hasil 8 kombinasi *neural network* dapat disimpulkan bahwa model dengan optimasi Adam menghasilkan akurasi yang lebih tinggi dibandingkan model dengan optimasi SGD. Hasil ini sesuai dengan hasil

penelitian penulis (Andalib & Vakili, 2020). Dari gambar tersebut juga dapat diketahui bahwa model yang menghasilkan hasil tertinggi pada kelompok Adam dilatih dengan *learning rate* 0,0001 di semua kombinasi *neural network*. Sedangkan model yang menghasilkan akurasi tertinggi pada kelompok SGD dilatih dengan *learning rate* 0,001, kecuali pada kombinasi BiLSTM-BiLSTM. Penggunaan fungsi aktivasi tampaknya bervariasi pada setiap kombinasi *neural network*. *Neural network* yang menggunakan LSTM atau BiLSTM sebagai layer pertama dan optimasi Adam kebanyakan menggunakan fungsi aktivasi Tanh dan *learning rate* 0,0001. Selain itu, hasil *training* dari semua skenario menunjukkan bahwa nilai akurasi dan f1 score selalu berjalan beriringan dan f1 score selalu sedikit lebih tinggi dari akurasi.

4.3. Training dan Test Fase 2

Setelah mendapatkan semua hasil yang dijelaskan sebelumnya pada fase pertama, penelitian dilanjutkan ke tahap kedua di mana *training* model dilanjutkan hingga 1000 iterasi. Karena keterbatasan sumber daya, pada penelitian ini hanya dipilih dua model yang menghasilkan kinerja tertinggi dari setiap kombinasi *neural network*. Model pertama adalah model yang dilatih dengan optimasi Adam dan model kedua adalah model yang dilatih dengan optimasi SGD. Dengan demikian, ada 16 model yang dilatih pada tahap kedua.

Gambar 4.2 menggambarkan hasil training tahap kedua dari 16 model terpilih. Sumbu X pada Gambar 4.2 mewakili kombinasi *neural network*, optimizer, fungsi aktivasi dan *learning rate* dari masing-masing model, dan sumbu Y

menunjukkan kinerja model baik dalam akurasi maupun f1 score. Dari Gambar 4.2 dapat dilihat bahwa semua model yang dilatih dengan optimasi Adam, kecuali kombinasi GRU-LSTM, menghasilkan akurasi lebih tinggi dari 98%. Di sisi lain, hasil model yang dilatih menggunakan pengoptimal SGD masih di bawah 97,6%.

Gambar 4.2. Perbandingan kinerja model terpilih dalam 1000 iterasi training

Rincian hasil *training* tahap kedua disajikan pada Tabel 4.5. Hasil *training* tahap pertama juga disajikan dalam tabel sebagai bahan perbandingan. Akurasi dan f1 score kombinasi LSTM-GRU dengan optimasi Adam sedikit meningkat dan masih merupakan hasil tertinggi dari semua kombinasi *neural network* lainnya. Model mencapai akurasi 98,3651% dan f1 score 98,3993%. Di sisi lain, performa

LSTM-LSTM dengan optimasi Adam hampir melampaui performa kombinasi LSTM-GRU dengan akurasi 98,3609% dan f1 score 98,3952%.

Tabel 4.5. Perbandingan performa model terpilih

NN Layer	Optimasi	Aktivasi	LR	Training Fase 1			Training Fase 2		
				Epoch	Accuracy (%)	F1 Score (%)	Epoch	Accuracy (%)	F1 Score (%)
LSTM-LSTM	Adam	Tanh	0.0001	95	97.8893	97.9194	948	98.3609	98.3952
LSTM-LSTM	SGD	Prelu	0.001	89	96.8225	96.8417	727	97.2433	97.2656
GRU-GRU	Adam	LRelu	0.0001	98	97.7312	97.7563	932	98.1862	98.2335
GRU-GRU	SGD	LRelu	0.001	93	96.9899	97.0180	521	97.4043	97.4305
BiLSTM- BiLSTM	Adam	Tanh	0.0001	93	97.7264	97.7516	924	98.3448	98.3793
BiLSTM- BiLSTM	SGD	LRelu	0.0001	98	96.8622	96.8902	994	97.3066	97.3323
BiGRU- BiGRU	Adam	Prelu	0.0001	75	97.8051	97.8319	994	98.2882	98.3227
BiGRU- BiGRU	SGD	Tanh	0.001	50	96.9267	96.9586	439	97.3347	97.3590
LSTM-BiLSTM	Adam	Tanh	0.0001	81	97.7461	97.7701	949	98.3569	98.3879
LSTM-BiLSTM	SGD	Tanh	0.001	76	96.9597	96.9883	676	97.3211	97.3451
GRU-BiGRU	Adam	Softmax	0.0001	97	97.7579	97.7817	861	98.1578	98.2018
GRU-BiGRU	SGD	LRelu	0.001	97	96.9391	96.9681	914	97.4407	97.4633
LSTM-GRU	Adam	Tanh	0.0001	88	**98.1087**	**98.1586**	934	**98.3651**	**98.3993**
LSTM-GRU	SGD	Prelu	0.001	100	96.8619	96.8831	798	97.2406	97.2649
GRU-LSTM	Adam	LRelu	0.0001	91	97.7527	97.7768	351	97.7906	97.7905
GRU-LSTM	SGD	Tanh	0.001	83	96.9451	96.9713	751	97.5188	97.5444

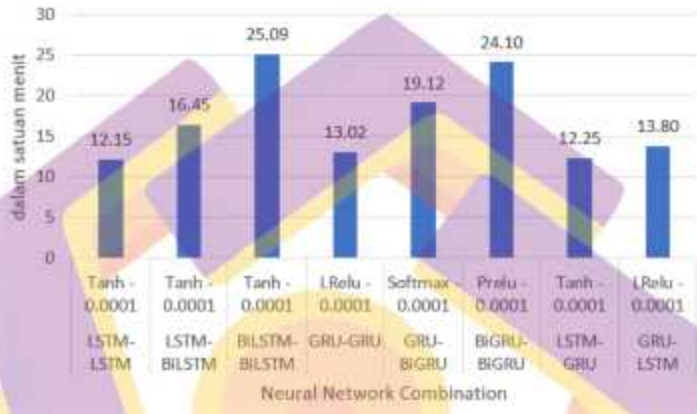
Model terbaik pada tahap pertama dan kedua ditentukan menggunakan validasi akurasi dan disimpan menggunakan ModelCheckPoint. Jumlah iterasi di mana akurasi validasi tertinggi dicapai pada kedua tahap dapat ditemukan di kolom epoch pada Tabel 4.5. Model dari kombinasi GRU-LSTM dengan optimasi Adam berhenti meningkat pada iterasi 351 sehingga model hanya mencapai akurasi 97,7906%. Kecuali model dengan kombinasi GRU-LSTM dan GRU-BiGRU, validasi akurasi model yang menggunakan optimasi Adam terus meningkat selama training hingga jumlah iterasi lebih dari 900. Kondisi berbeda terjadi pada model dengan optimasi SGD dimana jumlah iterasi untuk mencapai validasi akurasi yang optimal berbeda-beda.

Gambar 4.3. Perbandingan kinerja model yang menggunakan optimasi Adam

Pada Gambar 4.3 disajikan data kinerja model yang dilatih menggunakan optimasi Adam yang dikelompokkan berdasarkan *neural network*. Gambar tersebut menunjukkan bahwa penggunaan Bidirectional layer pada LSTM tidak meningkatkan kinerja model. Hal yang sama juga terjadi pada model model GRU yang digabungkan dengan Bidirectional layer hanya pada layer pertama. Meskipun demikian, model GRU yang dikombinasikan dengan Bidirectional layer pada kedua layer menghasilkan hasil yang paling tinggi dibandingkan dengan kombinasi *neural network* GRU lainnya. Meskipun demikian, kinerja yang dihasilkan oleh kombinasi *neural network* GRU masih berada di bawah kombinasi *neural network* LSTM. Hasil yang sama juga diperoleh pada penelitian oleh penulis (Al-Emadi et al., 2020) yang melakukan studi menggunakan dataset NSL-KDD. Hasil penelitian (Le et al., 2019) yang melatih model LSTM, GRU dan RNN pada dataset NSL-KDD dan

ISCX juga mengungkapkan bahwa model LSTM memiliki akurasi yang lebih tinggi dibandingkan dengan metode GRU dan RNN.

4.4. Durasi Training Model



Gambar 4.4. Perbandingan durasi traning dan validasi model

Selama proses *training*, durasi *training* dan validasi setiap model dicatat menggunakan fungsi Time dan Datetime pada Python. Gambar 4.4 menampilkan perbandingan durasi *training* dan validasi periterasi untuk setiap model yang dipilih. Pada gambar tersebut hanya disajikan data model yang menggunakan optimasi Adam dengan tujuan untuk menyederhanakan karena selisih durasi *training* dan validasi antara model yang menggunakan optimasi Adam dan optimasi SGD pada kombinasi *neural network* yang sama sangat kecil. Data yang ditampilkan adalah rata-rata durasi *training* dan validasi setiap model selama 100 epoch yang dilakukan pada tahap pertama. Sumbu Y mewakili durasi *training* dalam satuan menit. Berdasarkan Gambar 4.4 dapat diketahui bahwa model dengan

durasi terpanjang adalah model kombinasi BiLSTM-BiLSTM dengan optimasi Adam. Model yang menggunakan Bidirectional layer pada *neural network* layer pertama dan layer memiliki durasi *training* yang lebih tinggi dibandingkan dengan model yang hanya menggunakan layer dasar LSTM dan GRU. Model yang menggunakan Bidirectional layer hanya pada layer kedua, seperti kombinasi LSTM-BiLSTM dan GRU-BiGRU, memiliki durasi *training* dan validasi yang sedikit lebih lama daripada model yang menggunakan layer LSTM dan GRU dasar.

Tabel 4.6. Perbandingan durasi traning model yang terpilih

Kombinasi neural network	Optimasi	Aktivasi	LR	Durasi training dalam menit
LSTM-LSTM	Adam	Tanh	0.0001	12.14661523
LSTM-LSTM	SGD	Prelu	0.001	13.74149228
GRU-GRU	Adam	LRelu	0.0001	13.0155901
GRU-GRU	SGD	LRelu	0.001	13.08156655
BiLSTM- BiLSTM	Adam	Tanh	0.0001	25.09201881
BiLSTM- BiLSTM	SGD	LRelu	0.0001	24.61162936
BiGRU- BiGRU	Adam	Prelu	0.0001	24.10067599
BiGRU- BiGRU	SGD	Tanh	0.001	22.64522369
LSTM-BiLSTM	Adam	Tanh	0.0001	16.4516655
LSTM-BiLSTM	SGD	Tanh	0.001	16.76979497
GRU-BiGRU	Adam	Softmax	0.0001	19.1203574
GRU-BiGRU	SGD	LRelu	0.001	17.06359603
LSTM-GRU	Adam	Tanh	0.0001	12.24564472
LSTM-GRU	SGD	Prelu	0.001	12.62704598
GRU-LSTM	Adam	LRelu	0.0001	13.80322407
GRU-LSTM	SGD	Tanh	0.001	14.14591413

Meskipun memiliki durasi *training* dan validasi yang paling lama, namun performansi model dengan kombinasi BiLSTM-BiLSTM masih lebih rendah dibandingkan model dengan kombinasi LSTM-GRU dan LSTM-LSTM, sehingga penggunaan model kombinasi LSTM-GRU dan LSTM-LSTM lebih direkomendasikan. Rincian durasi training dan validasi untuk masing-masing model disajikan pada Tabel 4.6. Berdasarkan tabel tersebut, model kombinasi

LSTM-LSTM dengan optimasi Adam memiliki durasi *training* dan validasi terpendek dibandingkan model lainnya. Di sisi lain, kinerja model hanya sedikit lebih rendah dibandingkan model dengan kinerja tertinggi dalam penelitian ini, yaitu model kombinasi LSTM-GRU. Model dengan performa tertinggi ini memiliki durasi 0,48043075 menit lebih lama dibandingkan model kombinasi LSTM-LSTM.

4.5. Pembahasan Hasil

Dari hasil penelitian ini diketahui bahwa model yang menghasilkan akurasi tertinggi adalah metode gabungan *(hybrid)*, yaitu metode kombinasi LSTM-GRU dimana LSTM digunakan sebagai *input layer* dan GRU sebagai *hidden layer*. Konfigurasi parameter untuk model ini yaitu menggunakan optimasi Adam, fungsi aktivasi Tanh dan *learning rate* 0.0001. Hasil ini didapatkan setelah melakukan beberapa kali eksperimen pada penggunaan layer LSTM dan GRU.

Pada awal penelitian hanya kombinasi layer yang digunakan, yaitu LSTM-LSTM dan GRU-GRU. Peningkatan akurasi yang dihasilkan tidak terlalu signifikan jika dibandingkan dengan hasil penelitian sebelumnya dimana penulis (Khan, 2021) yang menggunakan metode HCRNN menghasilkan akurasi model 97.75%, sedangkan akurasi tertinggi yang dihasilkan oleh model LSTM-LSTM hanya mencapai 97.8893%.

Pada dua kelompok eksperimen berikutnya dilihat pengaruh layer Bidirectional terhadap performa model dengan menguji kombinasi metode BiLSTM-BiLSTM, BiGRU-BiGRU, LSTM-BiLSTM dan GRU-BiGRU. Dari hasil pengujian layer Bidirectional terhadap performa model, diketahui bahwa

penggunaan layer Bidirectional baik pada *hidden layer* saja maupun *input layer* dan *hidden layer* tidak meningkatkan akurasi model. Dengan demikian, meskipun model mempelajari data dalam dua arah saat menggunakan layer Bidirectional (Schuster & Paliwal, 1997), namun tidak terjadi peningkatan pada performa model. Perlu diingat bahwa data yang digunakan pada penelitian ini berupa data angka, bukan data berupa teks yang mungkin mempunyai hubungan antar kata dan memiliki konteks. Oleh karena itu, dapat diasumsikan tidak ada pengaruh penggunaan Bidirectional layer pada model yang menggunakan dataset CIC IDS 2017.

Di sisi lain, durasi *training* model menjadi lebih lama jika dibandingkan dengan model yang tidak menggunakan layer Bidirectional. Hal ini memang sesuai dengan cara kerja layer Bidirectional yang meningkatkan kompleksitas perhitungan pada neuron. Hal serupa juga disampaikan oleh peneliti (Imrana et al., 2021) dimana hasil penelitiannya mengungkapkan bahwa metode Bidirectional LSTM mempunyai durasi *training* yang lebih lama dari LSTM. Namun hal berlawanan dengan hasil penelitian (Imrana et al., 2021) muncul dari sisi performa model dimana peneliti (Imrana et al., 2021) menyebutkan bahwa metode Bidirectional LSTM mempunyai performa yang lebih tinggi dari LSTM. Hasil penelitian ini mengungkapkan hasil yang sebaliknya, bahkan pada model dengan *learning rate* yang sama. Perbedaan dataset yang digunakan pada proses *training* memungkinkan hal ini terjadi dimana peneliti (Imrana et al., 2021) menggunakan dataset NSL-KDD dan penelitian ini menggunakan dataset CIC IDS 2017. Di sisi lain, hasil yang berbeda dihasilkan oleh kombinasi GRU. Kombinasi GRU-BiGRU menghasilkan

akurasi yang lebih rendah dibandingkan kombinasi BiGRU-BiGRU, namun masih lebih tinggi dari hasil kombinasi GRU-GRU.

LSTM layer memiliki akurasi yang lebih baik pada *dataset* ini ketika diimplementasikan tanpa Bidirectional layer. Sebaliknya, GRU layer memiliki akurasi yang lebih tinggi ketika diimplementasikan dengan Bidirectional layer. Fakta tentang GRU yang diperoleh dalam percobaan ini sesuai dengan hasil penelitian (Xu et al., 2018) yang mengusulkan Bidirectional GRU untuk membangun arsitektur *neural network*.

Eksperimen dilanjutkan ke metode *hybrid* mengingat tidak ada peningkatan yang dianggap cukup dari penggunaan layer Bidirectional. Dari metode hybrid ini ditemukan peningkatan akurasi yang cukup tinggi untuk kombinasi metode LSTM-GRU, yaitu mencapai akurasi 98,1087% dalam 100 iterasi *training*. Namun hal yang sama tidak terjadi pada kombinasi metode GRU-LSTM. Metode LSTM-GRU merupakan satu-satunya metode yang mencapai akurasi di atas 98% pada *training* fase 1.

Secara umum, model yang menggunakan fungsi optimasi Adam mempunyai akurasi yang lebih tinggi jika dibandingkan dengan model yang menggunakan optimasi SGD. Hasil ini sesuai dengan hasil penelitian penulis (Andalib & Vakili, 2020) dimana fungsi optimasi Adam mempunyai performa yang lebih tinggi dari fungsi optimasi SGD, RMSProp, Adagrad dan Adadelta. Studi (Andalib & Vakili, 2020) menggunakan dataset yang sama dengan penelitan (Imrana et al., 2021), yaitu dataset NSL-KDD. Hasil penelitian (Ni & Cao, 2020) menemukan hal yang sama bahwa fungsi optimasi Adam membuat kecepatan dan

tingkat akurasi yang lebih baik jika dibandingkan dengan fungsi optimasi SGD, RMSProp, Adadelta dan Momentum. Dari hasil penelitian ini diketahui bahwa model yang menghasilkan hasil tertinggi pada kelompok Adam dilatih dengan *learning rate* 0,0001 di semua kombinasi *neural network*.

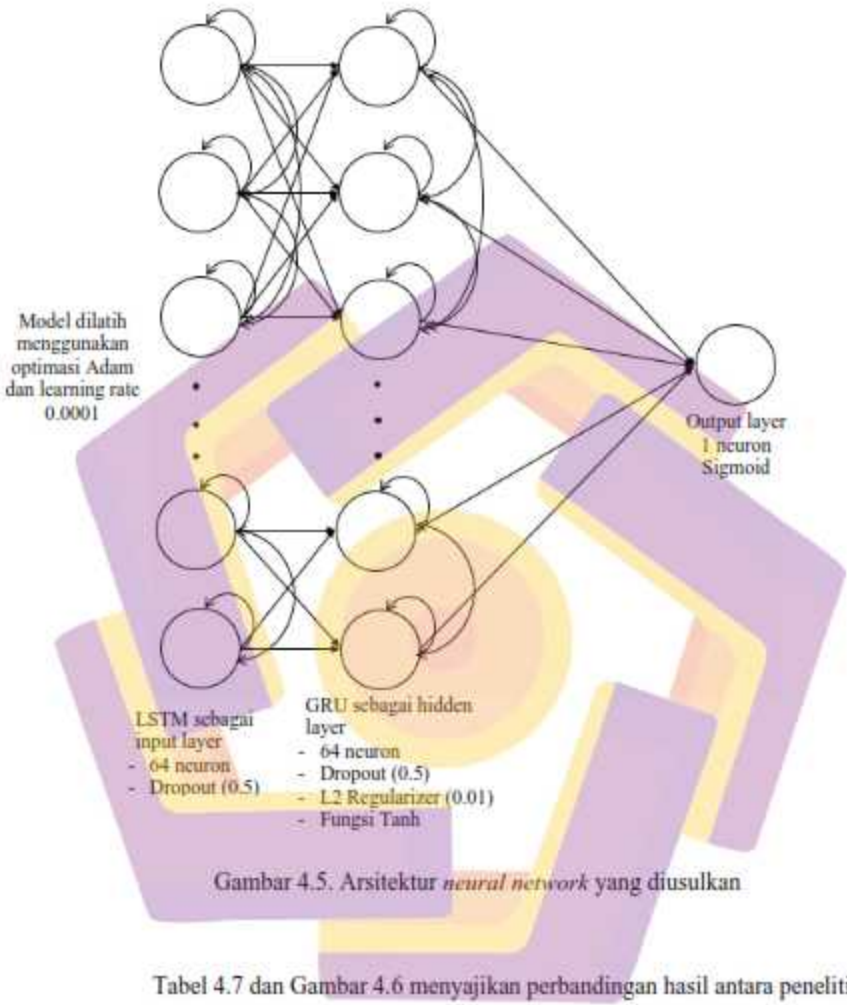
Fungsi optimasi Adam diusulkan oleh peneliti (Kingma & Ba, 2015) dimana fungsi optimasi ini mempunyai kemampuan untuk menyesuaikan momentum dalam *training*. Dengan menggunakan fungsi optimasi Adam, model dapat melakukan *convergence* dengan lebih cepat sehingga sebenarnya dapat mengurangi jumlah iterasi yang diperlukan dalam *training*. Hal lain yang sangat menarik dari fungsi optimasi Adam adalah kemampuan model untuk terus meningkat dalam jumlah iterasi yang sangat besar. Hal ini terlihat dari hasil *training* dan *test* pada fase 2, dimana model yang menggunakan fungsi optimasi Adam dapat mencapai akurasi terbaiknya pada jumlah iterasi training yang besar sedangkan model yang menggunakan fungsi optimasi SGD mencapai akurasi terbaiknya pada jumlah iterasi yang lebih rendah. Di dalam penelitian (Ni & Cao, 2020), model dilatih hingga 50.000 iterasi menggunakan fungsi optimasi Adam dan hasilnya performa model dapat terus berkembang.

Dari penelitian ini juga dapat diasumsikan bahwa layer LSTM yang digunakan sebagai input layer menghasilkan model dengan akurasi tinggi saat dilatih menggunakan fungsi optimasi Adam. Hal ini terlihat dari performa model LSTM-LSTM, LSTM-GRU, LSTM-BiLSTM dan BiLSTM-BiLSTM yang mencapai akurasi di atas 98% saat dilatih pada fase kedua (1000 iterasi).

Metode *hybrid* LSTM-GRU telah digunakan oleh beberapa peneliti, diantaranya peneliti (Ni & Cao, 2020), (Wang et al., 2019), (Muhammad et al., 2020) dan (Zhu et al., 2021). Peneliti (Ni & Cao, 2020) menggunakan metode ini untuk melakukan analisa sentimen pada dataset IMDB. Peneliti (Wang et al., 2019) menggunakan metode ini untuk melakukan prediksi pada Ball Screw. Peneliti (Zhu et al., 2021) menggunakan metode ini untuk melakukan prediksi harga air. Dan yang terakhir, peneliti (Zhu et al., 2021) menggunakan metode ini untuk mengenali kondisi-kondisi dalam proses pemanasan pada perusahaan baja. Dengan demikian, sampai saat laporan tesis ini ditulis, pada penelitian ini adalah pertama kalinya metode ini digunakan dalam membuat model Intrusion Detection System.

4.6. Perbandingan dengan Penelitian Sebelumnya

Dari hasil perbandingan akurasi dan durasi *training*, maka pada penelitian ini akan diusulkan arsitektur *neural network* dengan kombinasi layer LSTM-GRU, fungsi optimasi Adam, fungsi aktivasi Tanh dan learning rate 0.0001 sebagai model dengan performa terbaik untuk dibandingkan dengan penelitian sebelumnya. Meskipun model dengan kombinasi LSTM-LSTM dengan konfigurasi yang sama mempunyai durasi *training* yang lebih pendek, namun dalam penelitian ini faktor yang menjadi fokus utama adalah akurasi model sehingga model kombinasi LSTM-GRU yang pilih sebagai metode yang diusulkan. Arsitektur neural network yang diusulkan dapat dilihat pada Gambar 4.5. Arsitektur tersebut sesuai dengan *template neural network* yang sudah dipersiapkan sebelumnya pada Gambar 3.3 dengan konfigurasi *fix* dari model yang dijelaskan di atas.

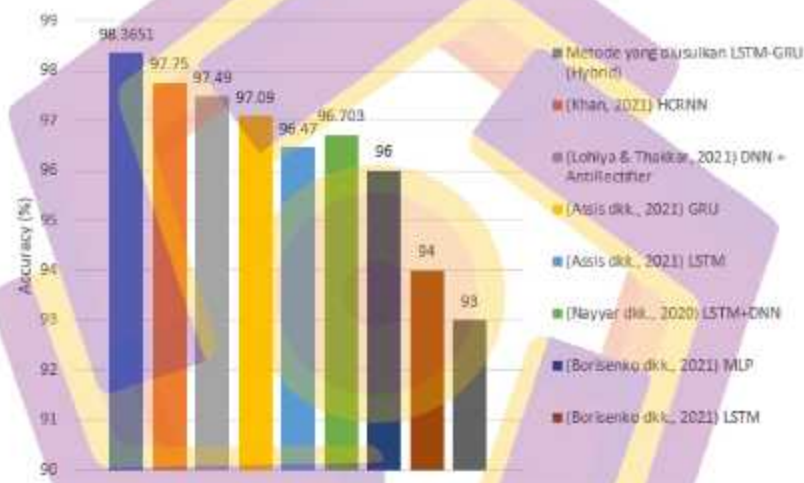


Gambar 4.5. Arsitektur *neural network* yang diusulkan

Tabel 4.7 dan Gambar 4.6 menyajikan perbandingan hasil antara penelitian ini dengan penelitian sebelumnya yang menggunakan *dataset* yang sama, dalam hal ini dataset CIC IDS 2017 secara keseluruhan dan tidak dilakukan pengurangan *feature*. Model kombinasi LSTM-GRU menghasilkan akurasi yang lebih tinggi dibandingkan dengan metode lain yang diusulkan sebelumnya oleh penulis lain. Metode ini meningkatkan akurasi model IDS yang dilatih pada *dataset* yang sama.

Tabel 4.7. Perbandingan kinerja metode dengan penelitian sebelumnya

Penelitian	Metode	Accuracy (%)
Metode yang diusulkan	LSTM-GRU (Hybrid)	98.3651
(Khan, 2021)	HCRNN	97.75
(Lohiya & Thakkar, 2021)	DNN + AntiRectifier	97.49
(Assis et al., 2021)	GRU LSTM	97.09 96.47
(Nayyar et al., 2020)	LSTM+DNN	96.7030
(Borisenko et al., 2021)	MLP LSTM	96.00 94.00
(Kim et al., 2020)	CNN-LSTM	93.00



Gambar 4.6. Perbandingan akurasi metode yang diusulkan dengan penelitian sebelumnya

BAB V

PENUTUP

5.1. Kesimpulan

1. Pada training tahap pertama, dilatih dalam 100 iterasi. Kinerja model tertinggi mencapai akurasi 98,1087% dan f1 score 98,1586% yang dihasilkan oleh kombinasi LSTM-GRU dengan optimasi Adam, fungsi aktivasi Tanh dan *learning rate* 0,0001. Selanjutnya, dipilih 16 model yang mempunyai akurasi tertinggi di setiap kombinasi *neural network* baik dengan optimasi Adam maupun optimasi SGD. Model terpilih dilatih lebih lanjut hingga 1000 epoch pada tahap kedua. Akurasi model tertinggi masing-masing mencapai akurasi 98,3651% dan f1 score 98,3993% dan ini masih dihasilkan oleh model yang sama.
2. Akurasi model kombinasi LSTM-GRU dengan optimasi Adam, fungsi aktivasi Tanh dan *learning rate* 0,0001 menghasilkan akurasi yang lebih tinggi dari pada penelitian sebelumnya yang menggunakan *dataset* yang sama.

5.2. Saran

Ada beberapa hal yang penulis rekomendasikan untuk dilakukan peneliti lain pada topik penelitian yang sama, antara lain:

1. Menambah atau mengurangi jumlah layer *neural network* dan kemudian melihat dampaknya pada durasi training dan akurasi model.

2. Mengubah nilai parameter Dropout layer dan L2 regularizer, dan juga nilai *batch size* dan kemudian melihat dampaknya pada durasi training dan akurasi model.
3. Beberapa fungsi aktivasi terbaru yang memiliki hasil yang menjanjikan seperti dalam penelitian (K. & K., 2020) belum diuji dalam penelitian ini.
4. Peneliti selanjutnya juga dapat melakukan *feature ranking* pada *dataset* dengan berbagai metode dengan harapan dapat mempersingkat durasi *training* dan validasi model.
5. Sebagian besar penelitian dilakukan dengan pendekatan *binary classification*, sehingga perlu ditingkatkan intensitas penelitian dengan pendekatan *multiclass classificaiton* pada dataset ini.
6. Jika peneliti selanjutnya memiliki *resource* komputasi yang sangat besar maka disarankan untuk menggunakan *dataset* yang lebih besar dan komprehensif yaitu CIC IDS 2018.
7. Terakhir, pembuatan *dataset* dengan jenis serangan terbaru tentunya akan sangat membantu pengembangan model IDS berbasis *deep learning*.

DAFTAR PUSTAKA

PUSTAKA BUKU

Calin, O., 2020, *Deep learning* Architectures – A Mathematical Approach, Springer, Switzerland.

El-Amir, H., Hamdy, M., 2020, *Deep learning* Pipeline: Building a *Deep learning* Model with TensorFlow, Apress Media, New York.

Garcia, S., Luengo, J., Herrera, F., 2015, *Data preprocessing* in Data Mining, Springer, Switzerland.

Goodfellow, I., Bengio, Y., Courville, A., 2016, *Deep learning*, MIT Press.

Gulli, A., Kapoor, A., Pal, S., 2019, *Deep learning* with TensorFlow 2 and Keras, Second Edition, Packt Publishing, Birmingham.

Michie, D., Spiegelhalter, D., Taylor, C. 2009. Machine Learning: Neural and Statistical Classification. Overseas Press, 2009th edition.

Moolayil, J., 2019, Learn Keras for Deep Neural *Networks* – A Fast-Track Approach to Modern *Deep learning* with Python, Apress, Canada.

Perrotta, P., 2020, Programming Machine learning – From Coding to *Deep learning*, Andy Hunt, Raleigh.

Smith, B., 2020, *Deep learning* With Python: Simple and Effective Tips and Tricks to Learn *Deep learning* with Python.

Wani, M.A., Bhat, F.A., Afzal, S., Khan, A.I., 2020, Advances in *Deep learning* – Studies in Big Data 57, Springer Nature, Singapore.

Yan, W.Q., 2020, Computational Methods for *Deep learning* - Theoretic, Practice and Applications, Springer, Switzerland.

Yi, Z., Tan, K.K., 2004, Convergence Analysis of Recurrent Neural *Networks*, Springer Science+Business Media, Dordrecht.

PUSTAKA MAJALAH, JURNAL ILMIAH ATAU PROSIDING

Abdella, J. A., Zaki, N. M., Shuaib, K., & Khan, F. (2021). Airline ticket price and demand prediction: A survey. *Journal of King Saud University - Computer and Information Sciences*, *33*(4), 375–391. https://doi.org/https://doi.org/10.1016/j.jksuci.2019.02.001

Aghdam, M. H., & Kabiri, P. (2016). Feature selection for intrusion detection system using ant colony optimization. *International Journal of Network Security*.

Al-Emadi, S., Al-Mohannadi, A., & Al-Senaid, F. (2020). Using Deep Learning Techniques for Network Intrusion Detection. *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020*. https://doi.org/10.1109/ICIoT48696.2020.9089524

Almalaq, A., & Zhang, J. J. (2020). Deep learning application: Load forecasting in big data of smart grids. In *Studies in Computational Intelligence*. https://doi.org/10.1007/978-3-030-31760-7_4

Andalib, A., & Vakili, V. T. (2020). An autonomous intrusion detection system using an ensemble of advanced learners. *2020 28th Iranian Conference on Electrical Engineering, ICEE 2020*. https://doi.org/10.1109/ICEE50131.2020.9260808

Anderson, J. P. (1980). Computer security threat monitoring and surveillance. *Technical Report James P Anderson Co Fort Washington Pa*. https://doi.org/citeulike-article-id:592588

Assis, M. V. O., Carvalho, L. F., Lloret, J., & Proença Jr, M. L. (2021). A GRU deep learning system against attacks in software defined networks. *Journal of Network and Computer Applications*, *177*, 102942.

Bace, R., & Mell, P. (2001). NIST special publication on intrusion detection systems. In *Nist Special Publication*.

Bansal, T., Belanger, D., & McCallum, A. (2016). Ask the GRU: Multi-task learning for deep text recommendations. *RecSys 2016 - Proceedings of the 10th ACM Conference on Recommender Systems*. https://doi.org/10.1145/2959100.2959180

Bhuvaneswari Amma, N. G., Selvakumar, S., & Leela Velusamy, R. (2021). Sagru: A stacked autoencoder-based gated recurrent unit approach to intrusion detection. *Advances in Intelligent Systems and Computing*, *1177*. https://doi.org/10.1007/978-981-15-5679-1_5

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys and Tutorials*, *16*(1). https://doi.org/10.1109/SURV.2013.052213.00046

Borisenko, B. B., Erokhin, S. D., Fadeev, A. S., & Martishin, I. D. (2021). Intrusion Detection Using Multilayer Perceptron and Neural Networks with Long Short-Term Memory. *2021 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO*, 1–6. https://doi.org/10.1109/SYNCHROINFO51390.2021.9488416

Chen, Z., Jiang, C., Masood, M. K., Soh, Y. C., Wu, M., & Li, X. (2020). Deep learning for building occupancy estimation using environmental sensors. In *Studies in Computational Intelligence*. https://doi.org/10.1007/978-3-030-31760-7_11

Chung, J. (2015). Gated Recurrent Neural Networks on Sequence Modeling arXiv : 1412 . 3555v1 [cs . NE] 11 Dec 2014. *International Conference on Machine Learning*.

Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR, abs/1412.3*. http://arxiv.org/abs/1412.3555

CyberEdge Group. (2021). *2021 Cyberthreat Defense Report*. https://cyber-edge.com/cdr/

Fulkerson, B., Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1995). *Machine Learning, Neural and Statistical Classification*. Technometrics. https://doi.org/10.2307/1269742

Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, *169*. https://doi.org/10.1016/j.jnca.2020.102767

Gharib, A., Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2017). An Evaluation Framework for Intrusion Detection Dataset. *ICISS 2016 - 2016 International Conference on Information Science and Security*. https://doi.org/10.1109/ICISSEC.2016.7885840

Graves, A, & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM networks. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, *4*, 2047–2052 vol. 4. https://doi.org/10.1109/IJCNN.2005.1556215

Graves, Alex, & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5), 602–610. https://doi.org/https://doi.org/10.1016/j.neunet.2005.06.042

Heberlein, L. T., Dias, G. V, Levitt, K. N., Mukherjee, B., Wood, J., & Wolber, D. (1989). *A network security monitor*. https://doi.org/https://doi.org/10.2172/6223037

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*. https://doi.org/10.1162/neco.1997.9.8.1735

Hochreiter, S., & Urgen Schmidhuber, J. J. (1997). Long Short-Term Memroy. In *Neural Computation*.

Hosseini, M. P., Lu, S., Kamaraj, K., Slowikowski, A., & Venkatesh, H. C. (2020). Deep Learning Architectures. In *Studies in Computational Intelligence*. https://doi.org/10.1007/978-3-030-31756-0_1

Imrana, Y., Xiang, Y., Ali, L., & Abdul-Rauf, Z. (2021). A bidirectional LSTM deep learning approach for intrusion detection. *Expert Systems with Applications*, *185*, 115524. https://doi.org/https://doi.org/10.1016/j.eswa.2021.115524

K., V., & K., S. (2020). Towards activation function search for long short-term model network: A differential evolution based approach. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/https://doi.org/10.1016/j.jksuci.2020.04.015

Khan, M. A. (2021). HCRNNIDS: Hybrid convolutional recurrent neural network-based network intrusion detection system. *Processes*, *9*(5). https://doi.org/10.3390/pr9050834

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*. https://doi.org/10.1186/s42400-019-0038-7

Kim, A., Park, M., & Lee, D. H. (2020). AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection. *IEEE Access*, *8*, 70245–70261. https://doi.org/10.1109/ACCESS.2020.2986882

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. *Cluster Computing*, *22*. https://doi.org/10.1007/s10586-017-1117-8

Le, T.-T.-H., Kim, Y., & Kim, H. (2019). Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks. *Applied Sciences*, *9*(7). https://doi.org/10.3390/app9071392

Leevy, J. L., & Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *Journal of Big Data*, *7*(1). https://doi.org/10.1186/s40537-020-00382-x

Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. In *Applied Sciences (Switzerland)*. https://doi.org/10.3390/app9204396

Lohiya, R., & Thakkar, A. (2021). Intrusion Detection Using Deep Neural Network with AntiRectifier Layer. In S. M. Thampi, J. Lloret Mauri, X. Fernando, R. Boppana, S. Geetha, & A. Sikora (Eds.), *Applied Soft Computing and Communication Networks* (pp. 89–105). Springer Singapore.

Madan, R., & Sarathimangipudi, P. (2018). Predicting Computer Network Traffic: A Time Series Forecasting Approach Using DWT, ARIMA and RNN. *2018 11th International Conference on Contemporary Computing, IC3 2018*. https://doi.org/10.1109/IC3.2018.8530608

Mazini, M., Shirazi, B., & Mahdavi, I. (2019). Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. *Journal of King Saud University - Computer and Information Sciences*, *31*(4), 541–553. https://doi.org/https://doi.org/10.1016/j.jksuci.2018.03.011

Muhammad, A. U., Yahaya, A. S., Kamal, S. M., Adam, J. M., Muhammad, W. I., & Elsafi, A. (2020). A Hybrid Deep Stacked LSTM and GRU for Water Price Prediction. *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, 1–6. https://doi.org/10.1109/ICCIS49240.2020.9257651

Narayanan, S. J., Perumal, B., Saman, S., & Singh, A. P. (2020). Deep learning for person re-identification in surveillance videos. In *Studies in Computational Intelligence*. https://doi.org/10.1007/978-3-030-31760-7_9

Nayyar, S., Arora, S., & Singh, M. (2020). Recurrent Neural Network Based Intrusion Detection System. *2020 International Conference on Communication and Signal Processing (ICCSP)*, 136–140. https://doi.org/10.1109/ICCSP48568.2020.9182099

Ni, R., & Cao, H. (2020). Sentiment Analysis based on GloVe and LSTM-GRU. *2020 39th Chinese Control Conference (CCC)*, 7492–7497. https://doi.org/10.23919/CCC50068.2020.9188578

Otoum, Y., & Nayak, A. (2021). AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. *Journal of Network and Systems Management*, *29*(3). https://doi.org/10.1007/s10922-021-09589-6

Potluri, S., Ahmed, S., & Diedrich, C. (2020). Securing Industrial Control Systems

from False Data Injection Attacks with Convolutional Neural Networks. In *Studies in Computational Intelligence*. https://doi.org/10.1007/978-3-030-31764-5_8

Priyadarshini, R., & Barik, R. K. (2019). A deep learning based intelligent framework to mitigate DDoS attack in fog environment. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/https://doi.org/10.1016/j.jksuci.2019.04.010

Ripley, B. D. (2008). *Pattern recognition and neural networks* (1st ed.). Cambridge University Press.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*. https://doi.org/10.1109/78.650093

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*. https://doi.org/10.5220/0006639801080116

Sohn, I. (2021). Deep belief network based intrusion detection techniques: A survey. In *Expert Systems with Applications* (Vol. 167). https://doi.org/10.1016/j.eswa.2020.114170

Thapa, N., Liu, Z., Kc, D. B., Gokaraju, B., & Roy, K. (2020). Comparison of machine learning and deep learning models for network intrusion detection systems. *Future Internet*. https://doi.org/10.3390/fi12100167

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*. https://doi.org/10.1109/ACCESS.2019.2895334

Wang, K., Huang, Y., Gong, L., Cai, C., & Zhang, Y. (2019). State-Wise LSTM-GRU Method for Ball Screw Prediction. *2019 IEEE Aerospace Conference*, 1–8. https://doi.org/10.1109/AERO.2019.8741555

Xie, L., & Yuille, A. (2017). Genetic CNN. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*. https://doi.org/10.1109/ICCV.2017.154

Xu, C., Shen, J., Du, X., & Zhang, F. (2018). An Intrusion Detection System Using a Deep Neural Network with Gated Recurrent Units. *IEEE Access*, *6*. https://doi.org/10.1109/ACCESS.2018.2867564

Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*. https://doi.org/10.1109/ACCESS.2017.2762418

Zhang, D., & Kabuka, M. R. (2018). Combining Weather Condition Data to Predict Traffic Flow: A GRU Based Deep Learning Approach. *Proceedings - 2017 IEEE 15th International Conference on Dependable, Autonomic and Secure Computing, 2017 IEEE 15th International Conference on Pervasive Intelligence and Computing, 2017 IEEE 3rd International Conference on Big Data Intelligence and Compu*. https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2017.194

Zhu, Q., Cao, W., & Song, W. (2021). Multi-condition recognition method based on LSTM_GRU for heating processes. *2021 40th Chinese Control Conference (CCC)*, 6544–6549. https://doi.org/10.23919/CCC52363.2021.9550672

LAMPIRAN 1. DATASET FEATURES

Dataset CIC IDS 2017 mempunyai 79 kolom dengan detail nama kolom berikut: Destination Port, Flow Duration, Total Fwd Packets, Total Backward Packets, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean, Fwd Packet Length Std, Bwd Packet Length Max, Bwd Packet Length Min, Bwd Packet Length Mean, Bwd Packet Length Std, Flow Bytes/s, Flow Packets/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min,Fwd IAT Total, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max, Fwd IAT Min, Bwd IAT Total, Bwd IAT Mean, Bwd IAT Std, Bwd IAT Max, Bwd IAT Min, Fwd PSH Flags, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Fwd Header Length, Bwd Header Length, Fwd Packets/s, Bwd Packets/s, Min Packet Length, Max Packet Length, Packet Length Mean, Packet Length Std, Packet Length Variance, FIN Flag Count, SYN Flag Count, RST Flag Count, PSH Flag Count, ACK Flag Count, URG Flag Count, CWE Flag Count, ECE Flag Count, Down/Up Ratio, Average Packet Size, Avg Fwd Segment Size, Avg Bwd Segment Size, Fwd Header Length.1, Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, Bwd Avg Bulk Rate, Subflow Fwd Packets, Subflow Fwd Bytes, Subflow Bwd Packets, Subflow Bwd Bytes, Init_Win_bytes_forward, Init_Win_bytes_backward, act_data_pkt_fwd, min_seg_size_forward, Active Mean, Active Std, Active Max, Active Min, Idle Mean, Idle Std, Idle Max, Idle Min, dan Label.