KINERJA MODEL DEEP LEARNING UNTUK DETEKSI SERANGAN SSH BRUTE FORCE

Victor Gilbert Danielo Waliana, Rissal Efendi
Teknik Informatika, Universitas Kristen Satya Wacana
Jalan Diponegoro 52-60, Kelurahan Noborejo, Kecamatan Argomulyo, Kota Salatiga, Jawa Tengah 50711
victorwaliana123@gmail.com

ABSTRAK

Serangan *Secure Shell (SSH) brute force* merupakan salah satu ancaman umum pada layanan jaringan karena dilakukan melalui percobaan autentikasi berulang hingga memperoleh akses tidak sah. Pendekatan berbasis *deep learning* banyak digunakan dalam sistem deteksi *intrusi (Intrusion Detection System/IDS)* untuk meningkatkan kemampuan identifikasi pola serangan. Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja model *Convolutional Neural Network (CNN)*, *Long Short-Term Memory (LSTM)*, dan *Autoencoder* dalam mendeteksi serangan *SSH brute force*. Dataset diperoleh dari hasil simulasi lalu lintas jaringan pada lingkungan virtual berbasis *Linux* yang menghasilkan 100.000 paket mentah dan difilter menjadi 20.163 *record trafik SSH*, terdiri dari 12.163 data serangan dan 8.000 data normal. Data dibagi menggunakan skema *train-test split* 80:20. Evaluasi model dilakukan menggunakan *confusion matrix, accuracy, precision, recall, F1-score*, dan *ROC-AUC*. Hasil eksperimen menunjukkan bahwa *CNN* memberikan performa terbaik dengan *ROC-AUC* sebesar 0,9934 dan *recall* 1,00 tanpa *false negative*. *LSTM* menunjukkan performa yang sangat kompetitif dengan *ROC-AUC* 0,9825 dan *recall* 1,00, sedangkan *Autoencoder* menghasilkan performa lebih rendah dengan *ROC-AUC* 0,7325 dan *recall* 0,03. Berdasarkan hasil tersebut, *CNN* dinilai sebagai arsitektur yang paling efektif untuk mendeteksi serangan *SSH brute force* pada dataset penelitian ini.

Kata kunci : *SSH brute force, Intrusion Detection System, Convolutional Neural Network, Long Short-Term Memory, Autoencoder, Deep Learning*

1. PENDAHULUAN

Perkembangan infrastruktur jaringan berbasis internet dalam satu dekade terakhir mendorong peningkatan kompleksitas arsitektur sistem serta volume lalu lintas data. Transformasi digital pada sektor pendidikan, industri, dan layanan publik memperluas permukaan serangan *(attack surface)* sehingga kebutuhan terhadap sistem keamanan jaringan yang adaptif menjadi semakin krusial. Salah satu layanan yang rentan terhadap eksploitasi adalah *Secure Shell* (*SSH*), protokol komunikasi terenkripsi yang umum digunakan pada server Linux untuk akses jarak jauh. Serangan *brute force* pada *SSH* dilakukan dengan mencoba kombinasi kredensial secara otomatis dan berulang hingga diperoleh akses yang valid. Pola serangan ini biasanya ditandai oleh frekuensi percobaan login gagal yang tinggi dalam interval waktu tertentu.

Dalam sistem deteksi intrusi, distribusi lalu lintas jaringan sering kali tidak seimbang *(class imbalance)*, sehingga model klasifikasi berpotensi bias terhadap kelas dengan proporsi dominan. Pada penelitian ini, dataset diperoleh dari lingkungan virtual berbasis Linux dengan total 20.163 record yang terdiri dari 60,32% trafik serangan dan 39,67% trafik normal. Distribusi tersebut merepresentasikan kondisi jaringan yang sedang terpapar aktivitas serangan intensif dan tetap mencerminkan ketidakseimbangan kelas yang relevan untuk evaluasi stabilitas model.

Pendekatan deteksi tradisional berbasis aturan *(rule-based atau signature-based IDS)* sebagaimana dijelaskan dalam studi Wang [1] efektif dalam mendeteksi ancaman yang telah memiliki pola terdokumentasi. Namun, karena bergantung pada *signature* yang telah ditentukan, metode ini memiliki keterbatasan dalam menghadapi serangan baru atau varian yang belum dikenali. Ahmed et al. [2] juga menjelaskan bahwa peningkatan kompleksitas lalu lintas jaringan dapat menyebabkan ketidakstabilan tingkat *false positive* dan *false negative*, serta meningkatkan beban pemrosesan ketika jumlah aturan semakin banyak.

Keterbatasan tersebut mendorong pemanfaatan pendekatan berbasis *deep learning* yang mampu mempelajari representasi fitur secara otomatis. Model *Long Short-Term Memory (LSTM)*, menurut penelitian Dash et al. [3], mampu "menangkap ketergantungan temporal dalam data lalu lintas jaringan sekuensial secara efektif." Pendekatan ini relevan untuk pola serangan yang terjadi secara berurutan dalam waktu tertentu. Studi Abdulganiyu et al. [4]menunjukkan peningkatan akurasi dibanding metode tradisional, meskipun efektivitasnya sangat bergantung pada struktur data dan keberadaan dependensi temporal yang jelas.

Selain *LSTM*, *Convolutional Neural Network (CNN)* juga banyak diterapkan dalam deteksi intrusi. Berdasarkan penelitian Bella et al. [5], *CNN* mampu melakukan ekstraksi fitur spasial secara otomatis dari representasi numerik lalu lintas jaringan. Studi Alqarni et al. [6] melaporkan performa kompetitif *CNN* dalam klasifikasi multi-kelas, terutama karena kemampuannya menangkap korelasi lokal antar atribut

numerik seperti panjang paket, nomor port, ukuran *window*, serta *sequence* dan *acknowledgment number*.

Teknik lain yang digunakan adalah *Autoencoder* sebagai model deteksi anomali. Menurut Hasan et al. [7], *Autoencoder* bekerja dengan mempelajari pola mayoritas dan mendeteksi anomali berdasarkan *reconstruction error*. Pendekatan ini efektif ketika trafik normal mendominasi dataset. Namun, sensitivitas terhadap distribusi data menjadi tantangan ketika proporsi serangan tidak jauh berbeda dari trafik normal atau ketika pola serangan memiliki kemiripan karakteristik numerik dengan trafik sah.

Studi Wu et al. [8][9] menegaskan bahwa serangan *brute force* terhadap layanan *SSH* merupakan ancaman signifikan pada server Linux dan memiliki pola statistik yang dapat dimanfaatkan untuk deteksi. Meskipun demikian, sebagian penelitian terdahulu lebih berfokus pada peningkatan performa model secara individual tanpa analisis komparatif yang konsisten pada distribusi data terukur, serta sering kali hanya melaporkan akurasi tanpa menyertakan metrik evaluasi yang lebih komprehensif.

Berdasarkan permasalahan tersebut, penelitian ini melakukan analisis komparatif kinerja *LSTM, CNN,* dan *Autoencoder* dalam mendeteksi serangan *SSH brute* force menggunakan dataset 20.163 record dari lingkungan virtual berbasis Linux. Evaluasi dilakukan menggunakan metrik *accuracy, precision, recall, F1-score, dan ROC-AUC* untuk memperoleh gambaran performa yang objektif dan menyeluruh. Hasil penelitian ini diharapkan dapat menjadi dasar pemilihan arsitektur *deep learning* yang lebih tepat untuk implementasi sistem *IDS* pada skenario serangan *SSH brute force*.

2. TINJAUAN PUSTAKA

Keamanan jaringan modern tidak lagi dapat dipahami sebagai mekanisme proteksi statis berbasis kontrol akses semata. Kompleksitas arsitektur terdistribusi, integrasi layanan *cloud*, dan mobilitas pengguna menuntut sistem keamanan yang adaptif serta mampu menganalisis pola trafik secara dinamis. Prinsip *Confidentiality, Integrity, dan Availability (CIA)* tetap menjadi fondasi konseptual, namun implementasinya memerlukan mekanisme pemantauan dan deteksi aktif.

Berdasarkan penelitian Randhika [10], keamanan jaringan merupakan sistem terintegrasi yang mencakup pemantauan, deteksi, dan respons. Deteksi intrusi berfungsi sebagai komponen analitik dalam arsitektur pertahanan berlapis. Ketergantungan eksklusif pada kontrol preventif berisiko karena serangan dapat melewati lapisan awal proteksi.

IDS secara umum diklasifikasikan menjadi Network-*based IDS (NIDS)* dan *Host-based IDS (HIDS).* Namun fokus penelitian ini berada pada paradigma deteksi. Wang [1][2] menjelaskan bahwa pendekatan berbasis aturan efektif untuk serangan yang telah memiliki *signature*, tetapi bersifat reaktif dan kurang adaptif terhadap pola baru. Kompleksitas aturan yang meningkat juga berdampak pada stabilitas *false positive* dan *false negative*.

Sebaliknya, pendekatan berbasis anomali membangun model perilaku normal dan mendeteksi deviasi sebagai indikasi serangan. Efektivitas pendekatan ini sangat dipengaruhi oleh distribusi data pelatihan. Oleh karena itu, *IDS* dalam penelitian ini diposisikan sebagai sistem klasifikasi berbasis pembelajaran mendalam yang dievaluasi secara kuantitatif melalui metrik komprehensif.

2.1. Karakteristik Serangan SSH Brute Force

Secure Shell (SSH) menyediakan akses jarak jauh terenkripsi pada sistem berbasis Linux. Meskipun mekanisme kriptografinya kuat, autentikasi berbasis kata sandi tanpa pembatasan percobaan login menciptakan celah terhadap serangan *brute force.*

Serangan *brute force* ditandai oleh percobaan autentikasi berulang dalam interval waktu singkat, menghasilkan frekuensi koneksi gagal yang tinggi. Wu [8]menunjukkan bahwa *trafik brute force* memiliki pola statistik berbeda dibanding trafik normal, terutama pada intensitas dan struktur sesi. Hal ini mengindikasikan bahwa serangan memiliki pola numerik terstruktur yang dapat dimodelkan.

Selanjutnya, Satpute [9] menjelaskan bahwa metode pembelajaran mesin mampu mengklasifikasikan *trafik normal* dan *brute force* secara efektif apabila fitur representatif tersedia. Dalam penelitian ini, trafik direpresentasikan melalui delapan atribut numerik paket jaringan, termasuk parameter port dan atribut *TCP*, sehingga memungkinkan analisis relasi non-linear tanpa ketergantungan pada aturan eksplisit.

2.2. Traffic Imbalance dalam IDS

Ketidakseimbangan kelas *(class imbalance)* merupakan isu krusial dalam sistem *IDS*. Secara umum, trafik normal mendominasi dibanding trafik serangan. Mbow [11] menegaskan bahwa distribusi tidak proporsional dapat menyebabkan bias model terhadap kelas mayoritas, menghasilkan akurasi tinggi namun sensitivitas rendah terhadap kelas minoritas. Fenomena misleading *accuracy* muncul ketika model tampak berkinerja baik secara statistik tetapi gagal mendeteksi serangan secara konsisten, Sayegh et al. [12] menunjukkan bahwa distribusi tidak seimbang dapat mengurangi kemampuan generalisasi terhadap variasi tertentu dalam data.

Berbeda dengan penelitian yang memodifikasi distribusi melalui *oversampling* atau *undersampling*, studi ini mempertahankan distribusi asli dataset untuk menjaga validitas eksperimental. Evaluasi dilakukan menggunakan *precision, recall, F1-score, dan ROC-AUC* guna memastikan analisis performa tidak bias terhadap satu metrik tunggal.

2.3. Deep Learning untuk Intrusion Detection

Deep learning memiliki kapasitas representasi tinggi untuk memodelkan relasi *non-linear* antar fitur

jaringan. Dalam *IDS*, hubungan antara port, panjang paket, dan atribut *TCP* sering kali tidak dapat direpresentasikan secara deterministik. Berdasarkan penelitian Bella [5][6], *Convolutional Neural Network (CNN)* efektif dalam mengekstraksi korelasi spasial antar fitur numerik melalui operasi konvolusi. Model ini mampu mengidentifikasi kombinasi nilai fitur yang secara konsisten muncul pada pola serangan.

Sementara itu, Dash et al. [3] menjelaskan bahwa *Long Short-Term Memory (LSTM)* dirancang untuk menangkap dependensi temporal dalam data sekuensial melalui mekanisme memori internal. Keunggulan ini optimal ketika data direpresentasikan sebagai rangkaian waktu eksplisit.

Autoencoder, sebagaimana dijelaskan dalam studi Hasan [7], menggunakan pendekatan rekonstruksi untuk mengukur deviasi melalui reconstruction error. Model ini efektif ketika distribusi normal stabil dan dominan, sehingga anomali dapat terdeteksi sebagai penyimpangan signifikan. Perbedaan mekanisme representasi ketiga arsitektur tersebut menjadikan perbandingan eksperimental relevan untuk menentukan kecocokan terhadap karakteristik dataset.

2.4. Evaluasi Kinerja

Evaluasi *IDS* menuntut pendekatan *multi-metrik* karena implikasi kesalahan klasifikasi berbeda secara operasional. *Confusion matrix terdiri dari True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN),* yang memberikan gambaran distribusi prediksi secara menyeluruh.

Precision mengukur validitas alarm, sedangkan *recall* mengukur sensitivitas deteksi terhadap seluruh serangan. *F1-score* menyeimbangkan kedua metrik tersebut dan dirumuskan sebagai:

ROC-AUC digunakan untuk mengevaluasi kemampuan diskriminatif model secara keseluruhan. Pendekatan evaluasi komprehensif ini memastikan bahwa performa model dianalisis secara objektif dan proporsional terhadap distribusi data.

2.5. Posisi dan Kontribusi Penelitian

Sebagian besar penelitian *IDS* berbasis deep learning berfokus pada satu arsitektur tanpa perbandingan langsung dalam kondisi dataset identik. Selain itu, penggunaan teknik penyeimbangan data sering mengubah distribusi asli sehingga meningkatkan performa secara artifisial.

Penelitian ini berkontribusi melalui desain eksperimen komparatif yang menguji *CNN, LSTM, dan Autoencoder* pada dataset yang sama tanpa modifikasi distribusi. Konsistensi preprocessing dan konfigurasi pelatihan memastikan bahwa perbedaan performa mencerminkan karakteristik arsitektur, bukan variasi eksperimental. Dengan pendekatan tersebut, penelitian ini tidak hanya memberikan hasil numerik, tetapi juga memperkuat validitas metodologis dalam evaluasi arsitektur *deep learning* untuk deteksi *SSH brute force*.

3. METODE PENELITIAN

3.1. Desain Penelitian

Penelitian ini menggunakan pendekatan kuantitatif dengan desain eksperimen terkontrol dan komparatif untuk mengevaluasi performa tiga arsitektur *deep learning* dalam mendeteksi serangan *SSH brute force* pada infrastruktur jaringan virtual berbasis Linux. Pendekatan kuantitatif dipilih karena fokus penelitian terletak pada pengukuran performa model menggunakan metrik klasifikasi yang terukur dan dapat direplikasi secara objektif.

Eksperimen dirancang untuk membandingkan *Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), dan Autoencoder* menggunakan dataset yang identik tanpa manipulasi distribusi kelas. Tidak dilakukan penyeimbangan ulang data karena penelitian ini bertujuan mempertahankan karakteristik asli hasil simulasi lalu lintas jaringan. Dengan demikian, setiap perbedaan performa yang muncul dapat diatribusikan pada karakteristik arsitektur model, bukan akibat modifikasi data. Pendekatan ini menjaga validitas internal serta memastikan komparasi berlangsung secara adil.

Seluruh tahapan dilakukan secara sistematis mulai dari *preprocessing*, pembagian data latih dan uji, pelatihan model dengan konfigurasi *hiperparameter* yang konsisten, hingga evaluasi *multi-metrik*. Struktur ini memastikan eksperimen dapat direplikasi dalam kondisi yang sama.

3.2. Dataset dan Lingkungan Eksperimen

Eksperimen dijalankan menggunakan *Python* dengan dukungan *TensorFlow dan Scikit-learn*. Proses pelatihan dilakukan pada *Google Colab* dengan akselerasi *GPU* untuk mempercepat komputasi matriks selama proses training. Dataset yang digunakan merupakan hasil simulasi lalu lintas jaringan yang telah melalui ekstraksi fitur dan proses pelabelan. Total dataset berjumlah 20.163 record, terdiri dari dua kelas, yaitu normal (0) dan serangan SSH brute force (1).

Tabel 1. Distribusi dataset penelitian

Komponen	Jumlah	Persentase
Total Data	20.163	100%
Data Normal	8.000	39,67%
Data Serangan	12.163	60,32%

Dari Tabel 1 dapat dilihat bahwa distribusi kelas tidak sepenuhnya seimbang, dengan dominasi kelas serangan sebesar 60,32%. Meskipun tidak ekstrem, kondisi ini berpotensi memengaruhi sensitivitas model terhadap kelas mayoritas. Penelitian ini secara sadar tidak melakukan balancing agar evaluasi mencerminkan kondisi data hasil eksperimen yang sebenarnya. Dengan demikian, interpretasi metrik seperti *precision* dan *recall* dilakukan dengan mempertimbangkan distribusi ini.

Dataset dibagi menggunakan metode *train-test split* dengan rasio 80:20 dan *parameter random_state*

42 untuk menjaga konsistensi replikasi. Proporsi ini dipilih untuk memberikan data latih yang cukup besar guna pembelajaran pola, sekaligus menyediakan data uji yang representatif untuk evaluasi objektif.

3.3. Preprocessing Data

Preprocessing dilakukan untuk memastikan stabilitas pelatihan dan mencegah bias evaluasi.

a. Pertama, dataset dipisahkan menjadi variabel fitur (X) dan label (y). *Label biner* digunakan dengan representasi 0 untuk normal dan 1 untuk serangan. Pemisahan ini menjadi dasar pembentukan model klasifikasi.
b. Kedua, normalisasi dilakukan menggunakan *MinMaxScaler* dengan rentang nilai 0 hingga 1. Proses fitting scaler dilakukan hanya pada data latih, kemudian diterapkan pada data uji. Strategi ini mencegah terjadinya data leakage yang dapat menyebabkan inflasi performa evaluasi secara tidak sah.
c. Ketiga, untuk kebutuhan arsitektur *CNN* dan *LSTM*, data diubah menjadi bentuk tiga dimensi *(samples, features, 1).* Transformasi ini diperlukan karena kedua arsitektur bekerja dalam bentuk *tensor* dan memerlukan dimensi tambahan untuk memproses hubungan antar fitur.
d. Keempat, *Autoencoder* dilatih hanya menggunakan data normal pada data latih. Pendekatan ini sesuai dengan konsep *anomaly detection*, di mana model mempelajari representasi normalitas dan mendeteksi penyimpangan berdasarkan *reconstruction error*. Strategi ini memungkinkan identifikasi serangan sebagai anomali tanpa pembelajaran eksplisit terhadap pola serangan.

3.4. Arsitektur dan Konfigurasi Model

Ketiga model dikonfigurasi dengan *parameter* pelatihan yang konsisten untuk menjaga keadilan komparatif. *Optimizer* yang digunakan adalah *Adam* dengan *learning rate* 0.001 dan *batch size* 32. *CNN* dan *LSTM* menggunakan *Binary Crossentropy* sebagai *loss function* dengan 30 *epoch* pelatihan. *Autoencoder* menggunakan *Mean Squared Error* karena berorientasi pada rekonstruksi input.

Tabel 2. Konfigurasi arsitektur model

Komponen	CNN	LSTM	AE
Layer utama	Conv1D 64 filter, kernel 2	LSTM 64 unit	Dense 32 → 16 (encoder)
Hidden layer	Dense 64 ReLU	Dense 64 ReLU	Dense 16 → 32 (decoder)
Output	Dense 1 Sigmoid	Dense 1 Sigmoid	Sigmoid
Loss function	Binary Crossentropy	Binary Crossentropy	Mean Squared Error
Optimizer	Adam (0.001)	Adam (0.001)	Adam (0.001)
Epoch	30	30	30
Batch size	32	32	32

Dari Tabel 2 dapat dianalisis bahwa *CNN* dirancang untuk menangkap pola lokal antar fitur melalui operasi konvolusi, sedangkan *LSTM* memodelkan dependensi fitur melalui mekanisme memori internal. *Autoencoder* menggunakan pendekatan kompresi representasi melalui *bottleneck layer* untuk mengidentifikasi penyimpangan pola. Keseragaman konfigurasi pelatihan memastikan bahwa perbedaan performa lebih merefleksikan karakteristik arsitektur dibandingkan variasi *hiperparameter*.

Penentuan *threshold* klasifikasi pada *Autoencoder* dilakukan menggunakan persentil ke-95 dari *reconstruction error* pada data uji. Data dengan error di atas *threshold* dikategorikan sebagai serangan.

3.5. Metode Evaluasi

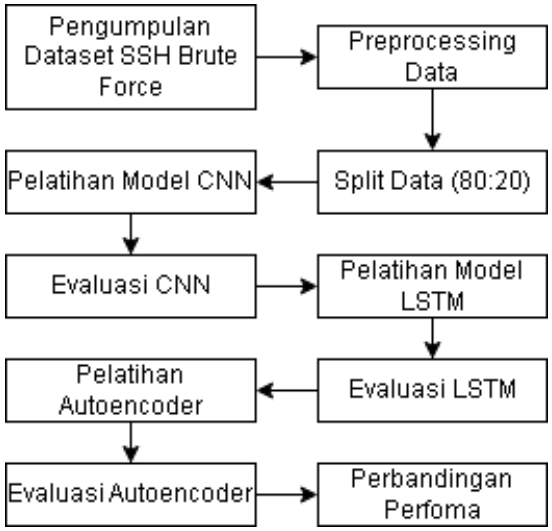
Evaluasi dilakukan menggunakan metrik *Accuracy, Precision, Recall, F1-score, False Positive Rate, dan ROC-AUC*. Pendekatan multi-metrik dipilih untuk menghindari bias evaluasi tunggal, terutama mengingat distribusi kelas yang tidak sepenuhnya seimbang.

Tabel 3. Rumus metrik evaluasi

Metrik	Rumus
Accuracy	(TP + TN) / (TP +TN + FP + FN)
Precision	TP / (TP + FP)
Recall	TP / (TP + FN)
F1-score	2 x (Precision x Recall) / (Precision + Recall)
False Positive Rate	FP / (FP + TN)

Dari Tabel 3 dapat dipahami bahwa *accuracy* mengukur ketepatan keseluruhan klasifikasi, *precision* mengevaluasi validitas alarm serangan, *recall* mengukur sensitivitas deteksi terhadap serangan aktual, sedangkan *F1-score* menjaga keseimbangan antara *precision* dan *recall*. *ROC-AUC* digunakan untuk menilai kemampuan diskriminatif model secara keseluruhan terhadap kedua kelas. *Confusion matrix* digunakan untuk mengidentifikasi distribusi *True Positive, True Negative, False Positive, dan False Negative* sebagai dasar analisis performa.

3.6. Alur Eksperimen Penelitian



Gambar 1. Diagram alur penelitian

Diagram pada Gambar 1 menggambarkan tahapan sistematis penelitian dimulai dari pengumpulan dataset hasil simulasi, *preprocessing*, pembagian data latih dan uji, pelatihan masing-masing model, evaluasi performa, hingga komparasi akhir. Struktur ini memastikan tidak terjadi tumpang tindih tahapan dan setiap model diuji dalam kondisi eksperimental yang setara.

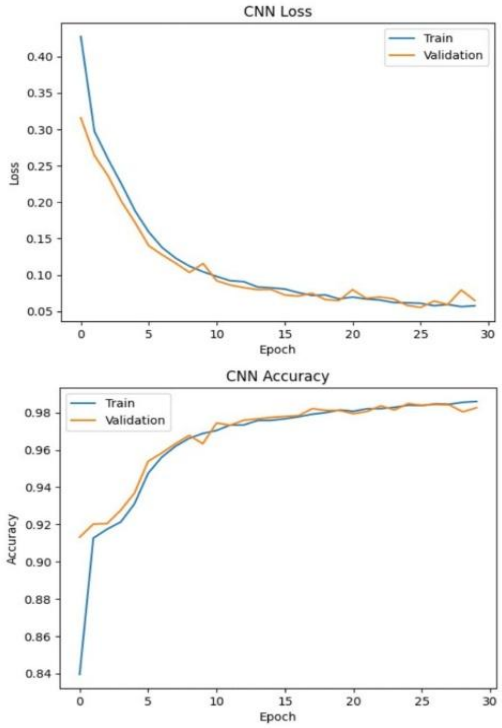
Pendekatan terstruktur tersebut memperkuat validitas komparatif penelitian serta memastikan bahwa kesimpulan yang dihasilkan benar-benar merepresentasikan performa aktual masing-masing arsitektur dalam mendeteksi serangan *SSH brute force*.

4. HASIL DAN PEMBAHASAN

4.1. Hasil dan Evaluasi Model CNN

Model *Convolutional Neural Network* dilatih selama 30 *epoch* dengan *batch size* 32 menggunakan *optimizer Adam* dan fungsi *loss Binary Crossentropy*. Proses pelatihan menunjukkan konvergensi yang stabil sejak *epoch* awal. Penurunan *loss* signifikan terjadi pada lima *epoch* pertama, yang menunjukkan bahwa model dengan cepat menangkap pola dominan dalam dataset. Setelah fase awal tersebut, penurunan *loss* berlangsung lebih gradual hingga *epoch* ke-30, yang mengindikasikan proses penyempurnaan bobot secara stabil tanpa fluktuasi ekstrem.

Dari grafik pada Gambar 2 dapat dilihat bahwa *kurva accuracy* meningkat secara konsisten dan mendekati 0,98 pada akhir pelatihan. *Kurva validation* mengikuti pola yang hampir identik dengan *kurva training* tanpa *divergensi* mencolok. Kondisi ini menunjukkan bahwa model tidak mengalami *overfitting* signifikan dan mampu melakukan generalisasi terhadap data uji.



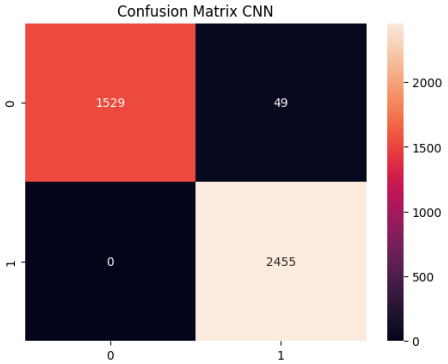
Gambar 2. Grafik training CNN

Evaluasi terhadap data uji menghasilkan performa sebagai berikut:

Tabel 4. Hasil evaluasi model CNN

Metrik	Nilai
Accuracy	98,26%
Precision	97,22%
Recall	100%
F1-score	0,9859
ROC-AUC	0,9934

Dari Tabel 4 dapat dilihat bahwa *CNN* mencapai *recall* 100%, yang berarti tidak terdapat *False Negative* pada data uji. Dalam konteks sistem deteksi intrusi, kondisi ini sangat krusial karena *False Negative* merepresentasikan serangan yang lolos dari deteksi. Nilai *ROC-AUC* sebesar 0,9934 menunjukkan kemampuan diskriminatif yang sangat tinggi dalam memisahkan trafik normal dan serangan.



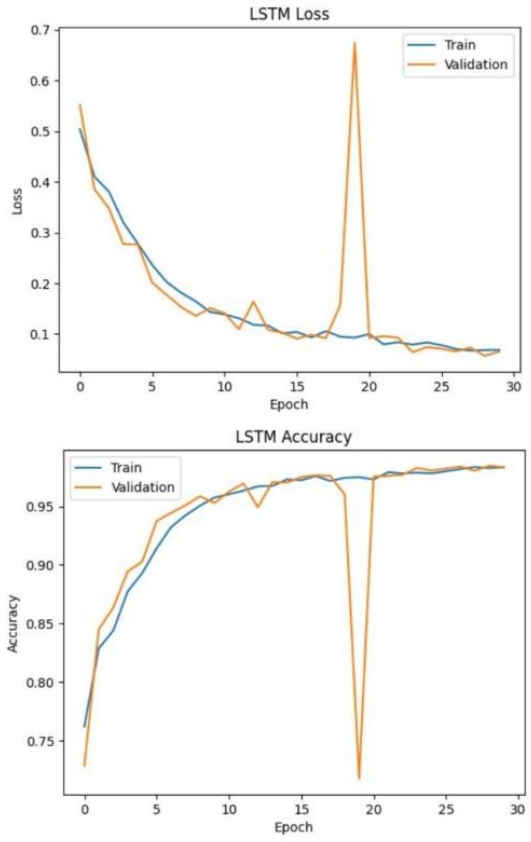
Gambar 3. Confusion matrix CNN

Confusion matrix pada Gambar 3 menunjukkan *True Negative* 1529, *False Positive* 49, *False Negative* 0, dan *True Positive* 2455. Tidak adanya *False Negative* memperkuat temuan bahwa model sangat sensitif terhadap pola serangan *brute force*. *False Positive* sebanyak 49 masih berada dalam batas yang dapat diterima untuk implementasi *IDS*, terutama jika dibandingkan dengan risiko kehilangan deteksi serangan.

Secara analitis, *CNN* efektif karena mampu menangkap korelasi lokal antar fitur numerik seperti *dst_port*, ukuran paket, dan parameter koneksi *TCP* melalui operasi konvolusi satu dimensi.

4.2. Hasil dan Evaluasi Model LSTM

Model *LSTM* dilatih dengan konfigurasi identik untuk menjaga keadilan komparasi. Proses pelatihan menunjukkan pola konvergensi yang stabil, meskipun terdapat satu lonjakan *validation loss* pada pertengahan *epoch*. Lonjakan tersebut bersifat sementara dan kembali stabil, sehingga tidak menunjukkan *overfitting* permanen.



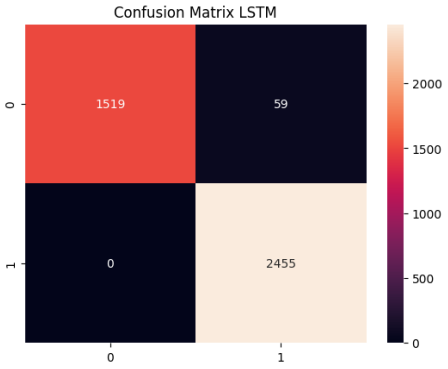
Gambar 4. Grafik training LSTM

Dari grafik pada Gambar 4 dapat diamati bahwa *kurva training* dan *validation* relatif paralel hingga akhir epoch. Nilai *accuracy* mendekati 0,98, sebanding dengan *CNN*, yang menunjukkan kemampuan generalisasi yang baik.

Tabel 5. Hasil evaluasi model LSTM

Metrik	Nilai
Accuracy	98,31%
Precision	97,30%
Recall	100%
F1-score	0,9863
ROC-AUC	0,9825

Dari Tabel 5 terlihat bahwa *LSTM* memiliki accuracy sedikit lebih tinggi dibanding *CNN*, namun *ROC-AUC* sedikit lebih rendah. Hal ini menunjukkan bahwa meskipun klasifikasi keseluruhan sangat baik, stabilitas pemisahan probabilistik *CNN* sedikit lebih unggul.



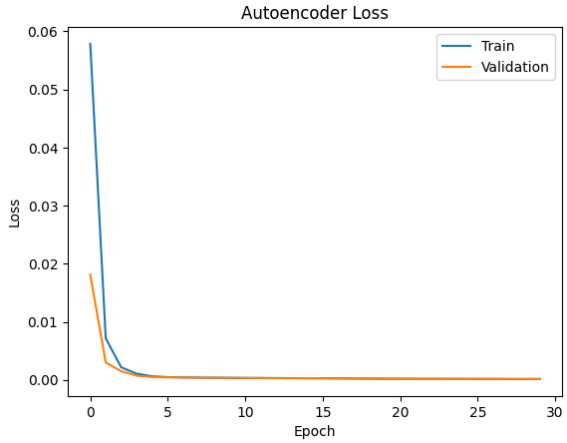
Gambar 5. Confusion matrix LSTM

Confusion matrix menunjukkan *True Negative* 1519, *False Positive* 59, *False Negative* 0, dan *True Positive* 2455. Seperti *CNN*, model ini tidak menghasilkan *False Negative*. Dengan demikian, seluruh serangan pada data uji berhasil terdeteksi.

Secara teoritis, *LSTM* dirancang untuk memproses dependensi temporal. Namun dalam eksperimen ini setiap *record* diperlakukan sebagai *instance independen* tanpa struktur *time-series* panjang. Hal tersebut menjelaskan mengapa keunggulan memori jangka panjang *LSTM* tidak memberikan peningkatan signifikan dibanding *CNN*.

4.3. Hasil dan Evaluasi Autoencoder

Autoencoder dilatih hanya menggunakan data normal pada data latih sesuai pendekatan *anomaly detection*. *Threshold* klasifikasi ditentukan menggunakan persentil ke-95 dari *reconstruction error*.



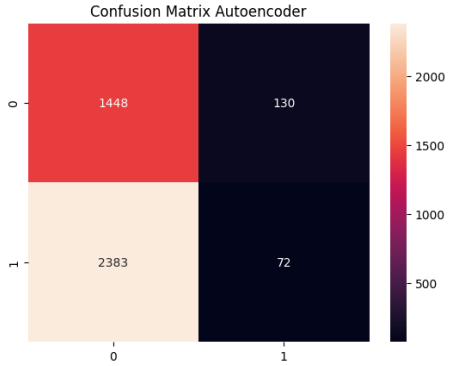
Gambar 6. Grafik training Autoencoder

Grafik pada Gambar 6 menunjukkan penurunan *Mean Squared Error* yang stabil selama pelatihan, yang menandakan model mampu merekonstruksi pola normal dengan baik. Namun hasil evaluasi pada data uji menunjukkan performa yang jauh lebih rendah dibanding model supervised.

Tabel 6. Hasil evaluasi Autoencoder

Metrik	Nilai
Accuracy	38,78%
Precision	46,53%
Recall	3,82%
F1-score	0,07
ROC-AUC	0,7325

Nilai *recall* sebesar 3,82% menunjukkan bahwa sebagian besar serangan tidak terdeteksi.



Gambar 7. Confusion matrix Autoencoder

Confusion matrix menunjukkan *True Negative* 1448, *False Positive* 130, *False Negative* 2383, dan *True Positive* 72. Tingginya *False Negative* mengindikasikan pendekatan *unsupervised* tidak sesuai dengan karakteristik dataset.

Distribusi dataset yang didominasi kelas serangan sebesar 60,32% tidak memenuhi asumsi dasar *anomaly detection* yang mengharapkan dominasi data normal. Selain itu, kedekatan numerik antar fitur menyebabkan *reconstruction error* tidak mampu menghasilkan pemisahan tajam.

4.4. Perbandingan dan Analisis Komparatif

Tabel 7. Perbandingan perfoma seluruh model

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
CNN	98,26%	97,22%	100%	0,9859	0,9934
LSTM	98,31%	97,30%	100%	0,9863	0,9825
AE	38,78%	46,53%	3,82%	0,07	0,7325

Dari Tabel 7 terlihat perbedaan jelas antara pendekatan supervised dan unsupervised. *CNN* dan *LSTM* menunjukkan performa sangat tinggi dengan *recall* 100%, sedangkan *Autoencoder* gagal mendeteksi mayoritas serangan.

Jika dibandingkan berdasarkan *ROC-AUC*, *CNN* memperoleh nilai tertinggi sebesar 0,9934, menunjukkan kemampuan diskriminatif paling stabil. *LSTM* sangat kompetitif namun tidak memberikan peningkatan signifikan. *Autoencoder* berada pada tingkat moderat dengan 0,7325, yang tidak memadai untuk sistem *IDS*.

4.5. Implikasi dan Rekomendasi Implementasi

Hasil penelitian menunjukkan bahwa *CNN* memberikan performa terbaik secara keseluruhan jika mempertimbangkan *ROC-AUC* tertinggi, *recall* 100%, keseimbangan *precision-recall*, serta stabilitas pelatihan. *LSTM* menunjukkan performa hampir identik, namun kompleksitas arsitekturnya tidak menghasilkan peningkatan signifikan dalam konteks dataset ini.

Autoencoder tidak sesuai untuk skenario di mana distribusi data tidak didominasi oleh trafik normal.

Berdasarkan evaluasi kuantitatif dan analisis komparatif, *CNN* direkomendasikan untuk implementasi sistem deteksi intrusi berbasis *SSH brute force* pada skenario penelitian ini, karena memberikan keseimbangan optimal antara performa klasifikasi dan efisiensi komputasi.

5. KESIMPULAN DAN SARAN

Penelitian ini membandingkan kinerja *CNN, LSTM, dan Autoencoder* dalam mendeteksi serangan *SSH brute force* menggunakan 20.163 *record* hasil filtrasi dari 100.000 paket mentah. Evaluasi menggunakan *confusion matrix, accuracy, precision, recall, F1-score, dan ROC-AUC* menunjukkan bahwa *CNN* memberikan performa terbaik dengan *ROC-AUC* 0,9934 dan *recall* 1,00 tanpa *false negative*, diikuti *LSTM* dengan *ROC-AUC* 0,9825 dan *recall* 1,00. *Autoencoder* menunjukkan performa rendah dengan *ROC-AUC* 0,7325 dan recall 0,03 sehingga kurang efektif pada distribusi data penelitian ini. Hasil ini menegaskan bahwa pendekatan *supervised* lebih sesuai dibandingkan pendekatan *unsupervised* untuk deteksi *SSH brute force* pada skenario yang diuji, dan *CNN* direkomendasikan sebagai model yang paling efektif. Untuk pengembangan selanjutnya, penelitian dapat diperluas pada dataset yang lebih besar, variasi jenis serangan lain, serta pengujian pada skenario *real-*

*tim*e atau pendekatan *hybrid* guna meningkatkan kemampuan generalisasi dan *robustnes* sistem deteksi intrusi.

DAFTAR PUSTAKA

[1] Y. Wang, "Deep Learning-Based Network Intrusion Detection Systems," *Applied and Computational Engineering*, vol. 109, pp. 179–188, 2024, doi: 10.54254/2755-2721/109/2024.18104.

[2] U. Ahmed *et al.*, "Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering," *Sci. Rep.*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-85866-7.

[3] N. Dash, S. Chakravarty, A. K. Rath, N. C. Giri, K. M. AboRas, and N. Gowtham, "An optimized LSTM-based deep learning model for anomaly network intrusion detection," *Sci. Rep.*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-85248-z.

[4] O. H. Abdulganiyu *et al.*, "Explainable attention based few shot LSTM for intrusion detection in imbalanced cyber physical system networks," *Sci. Rep.*, vol. 16, no. 1, p. 7217, Feb. 2026, doi: 10.1038/s41598-026-38668-4.

[5] K. Bella *et al.*, "An efficient intrusion detection system for IoT security using CNN decision forest," *PeerJ Comput. Sci.*, vol. 10, 2024, doi: 10.7717/PEERJ-CS.2290.

[6] A. A. Alqarni, A. Batin Hafr Al Batin, and S. M. Arabia El-Sayed El-Alfy, "Improving Intrusion Detection for Imbalanced Network Traffic using Generative Deep Learning," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, 2022. [Online]. Available: www.ijacsa.thesai.org

[7] T. Hasan, A. Hossain, M. Q. Ansari, and T. H. Syed, "Enhanced Intrusion Detection in IIoT Networks: A Lightweight Approach with Autoencoder-Based Feature Learning," *arXiv preprint* arXiv:2501.15266, 2025. [Online]. Available: http://arxiv.org/abs/2501.15266

[8] Y. Wu, P. M. Cao, A. Withers, Z. T. Kalbarczyk, and R. K. Iyer, "Mining Threat Intelligence from Billion-scale SSH Brute-Force Attacks," in Proceedings of the Workshop on Decentralized IoT Systems and Security (DISS), Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 2020. doi: 10.14722/diss.2020.23007.

[9] A. Satpute, S. Nikam, V. Gaikwad, Y. Kakade, and C. Mhaske, "AI-Driven Intrusion Detection System Using SSH Honeypots," *ICCK Transactions on Cybersecurity*, vol. 1, no. 1, p. 19, 2025, doi: 10.62762/TC.2025.521799.

[10] M. Radhika and M. Rashmi, "A Survey on Deep Learning-Based Intrusion Detection Systems with SHAP Explainability: Techniques, Challenges, and Future Directions," *IJFMR*, vol. 7, no. 5, 2025. [Online]. Available: www.ijfmr.com

[11] M. Mbow and H. Koide, "Handling class Imbalance problem in Intrusion Detection System based on deep learning," *IJNC*, vol. 12, no. 2, pp. 467–492, July 2022. [Online]. Available: www.ijnc.org

[12] H. R. Sayegh, W. Dong, B. H. Taher, M. M. Kadum, and A. M. Al-madani, "Optimal intrusion detection for imbalanced data using Bagging method with deep neural network optimized by flower pollination algorithm," *PeerJ Comput. Sci.*, vol. 11, 2025, doi: 10.7717/peerj-cs.2745

.