Sistemasi: Jurnal Sistem Informasi ISSN:2302-8149
Volume 15, Nomor 5, 2026: 1831-1844 e-ISSN:2540-9719

Model *Deep Learning* Hibrida CNN dan *Autoencoder* untuk Deteksi *Malware* Jaringan

Hybrid CNN and Autoencoder Deep Learning Model for Network Malware Detection

[1]Mayra Anggraini*, [2]Rama Aria Megantara
[1,2]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
[1,2]Jl. Imam Bonjol No.207, Pendrikan Kidul, Kec. Semarang Tengah, Kota Semarang, Jawa Tengah 50131
*e-mail: aria@dsn.dinus.ac.id

(***received*:** 1 May 2026, ***revised*:** 13 May 2026, ***accepted*:** 15 May 2026)

Abstrak

Malware merupakan salah satu ancaman utama dalam keamanan jaringan yang terus berkembang dengan pola serangan yang semakin kompleks dan sulit dideteksi menggunakan metode konvensional. Ketidakseimbangan data serta tingginya dimensi fitur menjadi tantangan utama dalam meningkatkan performa model deteksi malware. Penelitian ini bertujuan untuk mengembangkan model deteksi malware berbasis *deep learning* dengan pendekatan hibrida yang menggabungkan *Convolutional Neural Network* (CNN) dan Autoencoder. Dataset yang digunakan adalah CICIDS2017 versi perbaikan dengan lebih dari 2 juta data dan 91 fitur. Tahapan penelitian meliputi *data collection*, *exploratory data analysis* (EDA), *data preprocessing*, *feature selection*, serta penyeimbangan data menggunakan SMOTE diikuti dengan perancangan model dan evaluasi. Autoencoder digunakan untuk melakukan reduksi dimensi dan menghasilkan representasi fitur menjadi 32 fitur, yang selanjutnya digunakan sebagai input pada model CNN untuk klasifikasi multi-kelas. Hasil penelitian menunjukkan bahwa model yang di usulkan mampu mencapai akurasi yang tinggi dengan nilai *precision*, *recall*, dan *f1-score* yang baik pada sebagian besar kelas. Namun, performa pada kelas minoritas masih menunjukkan keterbatasan akibat ketidakseimbangan data. Dengan demikian, pendekatan hibrida CNN-Autoencoder terbukti efektif dalam meningkatkan kinerja deteksi malware jaringan.

Kata kunci: autoencoder, CNN, deep learning, deteksi malware, SMOTE

Abstract

Malware remains one of the primary threats to network security, continuously evolving with increasingly complex attack patterns that are difficult to detect using conventional methods. Data imbalance and high feature dimensionality are major challenges in improving the performance of malware detection models. This study aims to develop a deep learning-based malware detection model using a hybrid approach that combines Convolutional Neural Networks (CNN) and Autoencoders. The dataset used in this study was the improved version of the CICIDS2017 dataset, consisting of more than 2 million records and 91 features. The research stages included data collection, exploratory data analysis (EDA), data preprocessing, feature selection, and data balancing using SMOTE, followed by model design and evaluation. The Autoencoder was employed for dimensionality reduction, generating a compressed representation of 32 features, which was subsequently used as input for the CNN model in multi-class classification. The results demonstrate that the proposed model achieved high accuracy, along with strong precision, recall, and F1-score values across most classes. However, performance on minority classes still exhibited limitations due to data imbalance. Therefore, the hybrid CNN–Autoencoder approach proved effective in improving network malware detection performance.

Keywords: *autoencoder, CNN, deep learning, malware detection, SMOTE*

 ISSN:2302-8149
e-ISSN:2540-9719

1 Pendahuluan

Peningkatan aktivitas komunikasi data pada jaringan modern menyebabkan jumlah lalu lintas jaringan terus bertambah dan semakin kompleks. Kondisi ini berdampak pada meningkatnya resiko serangan siber, terutama malware jaringan yang mampu menyusup dan menyebar melalui aktivitas komunikasi data tersembunyi [1].

Metode deteksi tradisional berbasis signature masih memiliki keterbatasan dalam mengenali pola serangan baru, terutama pada malware yang menggunakan teknik *obfuscation* dan komunikasi tersembunyi. Kondisi ini menyebabkan beberapa jenis serangan sulit terdeteksi apabila pola serangan belum pernah tercatat sebelumnya [2].

Seiring jumlah fitur dan kompleksitas data jaringan menyebabkan metode machine learning konvensional mulai mengalami penurunan performa. Oleh karena itu, penelitian terbaru mulai memanfaatkan pendekatan deep learning karena mampu mempelajari pola data secara otomatis tanpa proses feature engineering yang kompleks [3].

Dalam penelitian ini, CNN digunakan karena mampu mengenali pola kompleks pada data hasil ekstraksi fitur jaringan. Sementara itu, Autoencoder dimanfaatkan untuk mereduksi dimensi data serta menghasilkan representasi fitur yang lebih ringkas sehingga proses klasifikasi dapat dilakukan secara lebih efisien [4].

Namun demikian, meskipun berbagai pendekatan deep learning telah diterapkan, beberapa penelitian masih menunjukkan keterbatasan dalam menangani data jaringan yang kompleks, terutama pada proses klasifikasi multi-kelas dan deteksi kelas minoritas. Hal tersebut menunjukkan bahwa penggunaan satu model saja belum sepenuhnya optimal untuk menghasilkan performa deteksi yang stabil.

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan pengembangan model hibrida yang mengintegrasikan CNN dan Autoencoder. Pendekatan ini bertujuan untuk menggabungkan keunggulan ekstraksi fitur lokal dari CNN dengan kemampuan representasi global dari Autoencoder, sehingga menghasilkan model deteksi malware yang lebih komprehensif, efisien, dan akurat.

Tujuan utama penelitian ini adalah merancang dan mengevaluasi model *Hybrid CNN-Autoencoder* untuk meningkatkan performa deteksi anomali dan klasifikasi malware pada jaringan. Model yang diusulkan diharapkan mampu memberikan tingkat akurasi yang lebih tinggi serta menurunkan *false positive rate* dibandingkan metode berbasis model tunggal.

2 Tinjauan Literatur

Berbagai penelitian terbaru mulai memanfaatkan pendekatan deep learning dalam mendeteksi malware dan intrusi jaringan karena mampu mempelajari pola serangan secara otomatis pada data jaringan berskala besar. Pendekatan ini dinilai lebih efektif dibandingkan metode machine learning konvensional, khususnya dalam menangani pola serangan yang kompleks dan dinamis [3].

CNN menjadi salah satu arsitektur deep learning yang banyak digunakan dalam penelitian keamanan jaringan karena mampu mengenali pola kompleks pada data secara otomatis. Beberapa penelitian menunjukkan bahwa CNN mampu menghasilkan performa klasifikasi yang baik pada proses deteksi malware dan intrusi jaringan [2].

Selain CNN, Autoencoder juga banyak diterapkan pada system deteksi anomali karena mampu mempelajari representasi data normal secara otomatis. Model ini dapat digunakan untuk mengenali pola yang menyimpang dari karakteristik data normal melalui proses rekonstruksi data [4].

Meskipun memberikan performa yang baik, penggunaan satu arsitektur saja masih memiliki beberapa keterbatasan. CNN lebih efektif dalam mengenali pola tertentu pada data, sedangkan Autoencoder lebih unggul dalam menghasilkan representasi fitur yang ringkas. Kondisi ini menyebabkan masing-masing belum sepenuhnya optimal apabila digunakan secara terpisah [3].

Untuk meningkatkan performa deteksi, beberapa penelitian mulai menggabungkan CNN dan Autoencoder dalam satu arsitektur hybrid. Kombinasi tersebut memungkinkan proses ekstraksi fitur dan representasi data dilakukan secara lebih optimal sehingga performa klasifikasi dapat ditingkatkan [3].

Namun demikian, pendekatan hybrid yang ada umumnya masih memiliki kelemahan dalam hal kompleksitas model serta efisiensi komputasi, sehingga kurang optimal untuk implementasi pada lingkungan nyata [1]. Selain itu, permasalahan seperti tingginya tingkat *false positive* dan

 ISSN:2302-8149
e-ISSN:2540-9719

keterbatasan generalisasi model terhadap dataset yang berbeda masih menjadi tantangan yang belum sepenuhnya teratasi.

Berdasarkan analisis tersebut, penelitian ini mengusulkan model hibrida CNN dan Autoencoder yang dirancang untuk mengoptimalkan integrasi fitur lokal dan global tanpa meningkatkan kompleksitas model secara signifikan. Pendekatan ini diharapkan mampu meningkatkan akurasi deteksi serta mengurangi tingkat *false positive* dibandingkan metode sebelumnya.

Dengan demikian, penelitian ini berkontribusi dalam pengembangan system deteksi malware jaringan yang lebih efisien, dan adaptif terhadap berbagai jenis serangan modern.

3 Metode Penelitian

Penelitian ini bertujuan untuk mengembangkan model deteksi malware jaringan berbasis *deep learning* dengan pendekatan hibrida yang menggabungkan Convolutional Neural Network (CNN) dan Autoencoder. Pendekatan hybrid ini telah terbukti mampu meningkatkan performa deteksi dengan mengombinasikan keunggulan ekstraksi fitur dan representasi data pada berbagai penelitian sebelumnya [3], [12].

Tahapan penelitian terdiri dari beberapa proses utama, yaitu pengumpulan data *(data collection)*, prapemrosesan data *(data preprocessing)* dan Exploratory Data Analysis, Feature Selection dan Balancing, perancangan model *(CNN dan Autoencoder)*, serta evaluasi performa model. Pendekatan ini mengacu pada penelitian deteksi intrusi berbasis *deep learning* yang menunjukkan bahwa integrasi beberapa tahap pengolahan data dapat meningkatkan akurasi stabilitas model [8]. Alur penelitian secara umum ditunjukkan pada Gambar 1.

Gambar 1 Alur metode penelitian

3.1 Data Collection

Pada penelitian ini, dataset diperoleh dari platform Kaggle melalui tautan *https://www.kaggle.com/datasets/ernie55ernie/improved-cicids2017-and-csecicids2018*, yang merupakan versi perbaikan dari dataset CICIDS2017 asli. Dataset ini telah mengalami proses pembersihan dan perbaikan terhadap kesalahan data serta inkonsistensi label, sehingga lebih sesuai untuk digunakan dalam penelitian *machine learning*. Data yang digunakan berupa *network traffic* berbasis aliran (*flow-based data)*, yaitu data yang diperoleh dari agensi paket jaringan menjadi satu aliran komunikasi. Setiap aliran direpresentasikan dalam bentuk fitur statistic yang menggambarkan karakteristik komunikasi jaringan.

Dataset ini terdiri dari sekitar 80 hingga 87 fitur yang mencakup beberapa parameter, seperti durasi aliran (*flow-duration*), jumlah paket, panjang paket, laju jaringan data (*flow byte per second)*, waktu antar paket (*inter arrival time)*, serta parameter waktu aktif dan idle. Fitur-fitur tersebut

 ISSN:2302-8149
e-ISSN:2540-9719

diperoleh melalui proses ekstraksi menggunakan tool seperti CICFLOWMeter, sehingga data telah dalam bentuk terstruktur dan siap digunakan dalam proses analisis.

Dataset ini mencakup beberapa kategori kelas, yaitu trafik normal (*BENIGN)* serta berbagai jenis serangan seperti DDoS, Portscan, Botnet, dan Botnet Attempted. Distribusi data dalam dataset ini bersifat tidak seimbang (*imbalanced dataset),* dimana jumlah data pada kelas normal jauh lebih besar dibandingkan dengan beberapa kelas serangan. Oleh karena itu, diperlukan teknik penanganan ketidak seimbangan data pada tahap selanjutnya. Dataset ini dipilih karena memiliki variasi jenis serangan yang kompleks serta mampu merepresentasikan kondisi jaringan secara realistis, sehingga sesuai untuk digunakan dalam penelitian deteksi malware berbasis deep learning.

3.2 Data Preprocessing dan EDA

Tahap kedua dalam penelitian ini adalah *Exploratory Data Analysis* dan *Data preprocessing*, yang bertujuan untuk memahami karakteristik dataset serta mempersiapkan data sebelum digunakan dalam proses pelatihan model. Pada tahap EDA, dilakukan analisis terhadap struktur dataset, termasuk jumlah data, tipe data pada setiap fitur, serta distribusi kelas untuk mengidentifikasi adanya ketidakseimbangan data (*imbalanced dataset).* Hasil analisis menunjukkan bahwa dataset memiliki distribusi kelas yang tidak seimbang, dimana jumlah data pada kelas *BENIGN* jauh lebih besar dibandingkan dengan kelas serangan seperti *Botnet* dan *Botnet Attempted.* Permasalahan ketidakseimbangan data merupakan salah satu tantangan utama dalam mendeteksi malware, karena dapat menyebabkan model cenderung bias terhadap kelas mayoritas [11].

Selanjutnya dilakukan tahap preprocessing untuk meningkatkan kualitas data. Proses ini diawali dengan penghapusan fitur yang tidak relevan, seperti *Flow ID, Source IP, Destination IP*, dan *Timestamp,* karena fitur tersebut tidak memberikan kontribusi langsung terhadap proses klasifikasi. Selanjutnya dilakukan penanganan nilai tak hingga (*infinite values)* dengan menggantinya menggunakan nilai yang valid agar tidak mengganggu proses komputasi. Proses normalisasi dilakukan untuk menyamakan skala data sehingga meningkatkan stabilitas pelatihan model *deep learning*, sebagaimana ditunjukkan dalam penelitian berbasis Autoencoder yang memerlukan input data terstandarisasi [6].

Selain itu, dilakukan proses encoding pada label untuk mengubah data kategorikal menjadi bentuk numerik sehingga dapat diproses oleh model machine learning. Data kemudian dinormalisasi menggunakan metode *Min-Max Scaling* untuk mengubah rentang nilai fitur menjadi 0 hingga 1. Normalisasi ini bertujuan untuk meningkatkan stabilitas dan performa model, terutama pada algoritma deep learning seperti CNN dan Autoencoder.

Melalui tahapan EDA dan Preprocessing ini, data yang semula mentah berhasil diubah menjadi dataset yang bersih, terstruktur, dan siap digunakan, untuk proses seleksi fitur serta pelatihan model.

3.3 Feature Selection dan Balancing

Tahap ketiga ini dilakukan proses penanganan ketidakseimbangan data (*imbalanced dataset)* untuk meningkatkan performa model dalam mengklasifikasi berbagai jenis serangan jaringan.

Berdasarkan hasil analisis pada tahap sebelumnya, diketahui bahwa distribusi data pada dataset tidak seimbang, jumlah data pada kelas mayoritas seperti BENIGN, jauh lebih besar dibandingkan dengan kelas minoritas seperti Botnet, Infiltration, dan beberapa jenis serangan lainnya. Ketidakseimbangan ini dapat menyebabkan model cenderung bias terhadap kelas mayoritas sehingga menurunkan kemampuan deteksi pada kelas minoritas.

Untuk mengatasi permasalahan tersebut, digunakan metode SMOTE (Synthetic Minority Over-sampling technique). Metode ini bekerja dengan menghasilkan data sintetis pada kelas minoritas berdasarkan kedekatan antar sampel dalam ruang fitur, sehingga distribusi data menjadi lebih seimbang.

Pada penelitian ini, parameter *K-neighbors* ditetapkan sebesar 1. Pemilihan nilai ini disesuaikan dengan kondisi dataset, karena terdapat beberapa kelas dengan jumlah data yang sangat sedikit, sehingga diperlukan pendekatan oversampling yang lebih konservatif agar tidak menghasilkan data sintetis yang terlalu menyimpang dari distribusi asli.

 ISSN:2302-8149
e-ISSN:2540-9719

Penerapan SMOTE dilakukan hanya pada data pelatihan (*training data)* untuk menjaga keaslian distribusi data pada data pengujian (*testing data)*. Dengan demikian, proses evaluasi model tetap mencerminkan performa model terhadap data yang belum pernah dilihat sebelumnya.

Setelah dilakukan proses balancing, jumlah data pada kelas minoritas meningkat secara signifikan, sehingga dataset menjadi lebih representatif. Hal ini diharapkan dapat meningkatkan kemampuan model dalam mengenali pola serangan, terutama pada kelas dengan jumlah data terbatas.

3.4 Model (CNN dan Autoencoder)

Pada penelitian ini digunakan pendekatan *deep learning* berbasis model hibrida, yaitu kombinasi antara Autoencoder dan Convolutional Neural Network (CNN) untuk meningkatkan kemampuan dalam mendeteksi dan mengklasifikasikan serangan jaringan.

Autoencoder digunakan sebagai metode *feature extraction* untuk menghasilkan representasi data yang lebih efisien dan mampu menangkap pola non-linear dari data lalu lintas jaringan. Model ini bekerja dengan dua bagian utama, yaitu *encoder* dan *decoder* [6], [10]. Bagian encoder berfungsi untuk mengompresi data ke dalam bentuk representasi berdimensi lebih rendah (*latent representation)*, sedangkan decoder berfungsi utuk merekonstruksi kembali data ke bentuk semula. Proses ini memungkinkan model untuk mempelajari fitur-fitur penting dari data secara otomatis tanpa supervise.

Selanjutnya, hasil representasi fitur dari Autoencoder digunakan sebagai input untuk model Convolutional Neural Network (CNN). Sebelum dimasukan ke dalam CNN, data terlebih dahulu diubah ke dalam bentuk matriks dua dimensi (*pseudo-image)* melalui proses *reshape*, sehingga dapat diproses oleh arsitektur CNN. CNN digunakan untuk melakukan klasifikasi data ke dalam beberapa kelas serangan jaringan karena kemampuannya yang tinggi dalam mengenali pola kompleks pada data jaringan serta meningkatkan akurasi deteksi malware dibandingkan metode tradisional [2], [15].

Arsitektur CNN yang digunakan terdiri dari beberapa lapisan utama, yaitu *Convolutional layer* untuk mengekstraksi fitur, *pooling layer* untuk mengurangi dimensi data, serta *fully connected layer* yang digunakan untuk proses klasifikasi akhir. Fungsi aktivasi yang digunakan pada lapisan output adalah softmax, yang memungkinkan model untuk menghasilkan probabilitas pada setiap kelas.

Kombinasi antara Autoencoder dan CNN dalam penelitian ini bertujuan untuk meningkatkan performa model dalam mendeteksi pola kompleks pada data jaringan. Autoencoder membantu dalam mereduksi dimensi dan mengekstraksi fitur penting, sedangkan CNN berperan dalam melakukan klasifikasi secara efektif berdasarkan representasi fitur yang telah dihasilkan.

3.5 Model Evaluation

Tahap evaluasi dilakukan untuk mengetahui kemampuan model dalam mengklasifikasi lalu lintas jaringan ke dalam beberapa kategori serangan. Proses pengujian menggunakan data testing yang tidak dilibatkan pada tahap pelatihan, sehingga hasil evaluasi dapat menunjukkan kemampuan model dalam mengenali pola pada data baru.

Evaluasi performa model dilakukan menggunakan beberapa metrik, yaitu accuracy, precision, recall, dan F1-score. Accuracy digunakan untuk melihat tingkat keberhasilan prediksi model secara keseluruhan, sedangkan precision dan recall digunakan untuk mengevaluasi kemampuan model dalam mengenali setiap kelas serangan. Selain itu, F1-score digunakan sebagai indikator keseimbangan antara precision dan recall dalam proses klasifikasi [8].

Selain metrik evaluasi utama, penelitian ini juga menggunakan confusion matrik untuk menganalisis hasil klasifikasi secara lebih detail. Melalui confusion matrik, dapat diketahui distribusi prediksi pada setiap kelas, sehingga kesalahan klasifikasi dan performa model pada masing-masing kategori serangan dapat dianalisis dengan lebih mendalam.

Hasil evaluasi digunakan untuk mengetahui efektivitas model hybrid CNN-Autoencoder dalam mendeteksi serangan jaringan serta mengidentifikasi bagian yang masih perlu dikembangkan guna meningkatkan performa model pada penelitian selanjutnya.

 ISSN:2302-8149
e-ISSN:2540-9719

4 Hasil dan Pembahasan

Pada penelitian ini dilakukan serangkaian proses mulai dari penggabungan dataset, preprocessing data, ekstraksi fitur menggunakan Autoencoder, hingga klasifikasi menggunakan Convolutional Neural Network (CNN). Dataset yang digunakan merupakan CICIDS2017 yang terdiri dari lima file harian (Monday-Friday) yang kemudian digabungkan satu dataset utuh, hasil penggabungan dataset dengan jumlah 2.099.972 data dan 91 fitur.

Berdasarkan hasil eksplorasi awal menggunakan fungsi **df.info()**, diketahui bahwa dataset memiliki variasi tipe data yang terdiri dari 61 fitur bertipe integer, 25 fitur bertipe float, dan 5 fitur bertipe object. Selain itu penggunakan memori dataset mencapai sekitar 1,4 GB

Seperti ditunjukan pada Gambar 2, struktur dataset menunjukkan bahwa seluruh kolom telah terbaca dengan baik tanpa adanya anomali pada tipe data.

```
# ========================
# 1. INFO DATA
# ========================
print("cek informasi struktur data")
df.info()
print("Shape:", df.shape)

print("\nTipe data:")
print(df.dtypes.value_counts())
```

```
cek informasi struktur data
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2099976 entries, 0 to 2099975
Data columns (total 91 columns):
```

Gambar 2 Struktur dataset

Selain itu dilakukan pengecekan terhadap duplikasi data untuk memastikan kualitas dataset. Hasil menunjukkan bahwa tidak ada duplikasi pada dataset. Sebagaimana ditunjukan pada Gambar 3, jumlah duplikasi Adalah nol.

```
# ========================
# 4. DUPLIKASI
# ========================
print("cek duplikasi rows")
print(df.duplicated().sum())
```

```
cek duplikasi rows
0
```

Gambar 3 Hasil pengecekan duplikasi data

Lalu dilakukan analisis terhadap missing value untuk memastikan tidak terdapat data yang hilang dan dapat mempengaruhi performa model. Hasil analisis menunjukkan bahwa tidak terdapat missing value yang signifikan pada dataset. Hal ini dapat dilihat pada Gambar 4

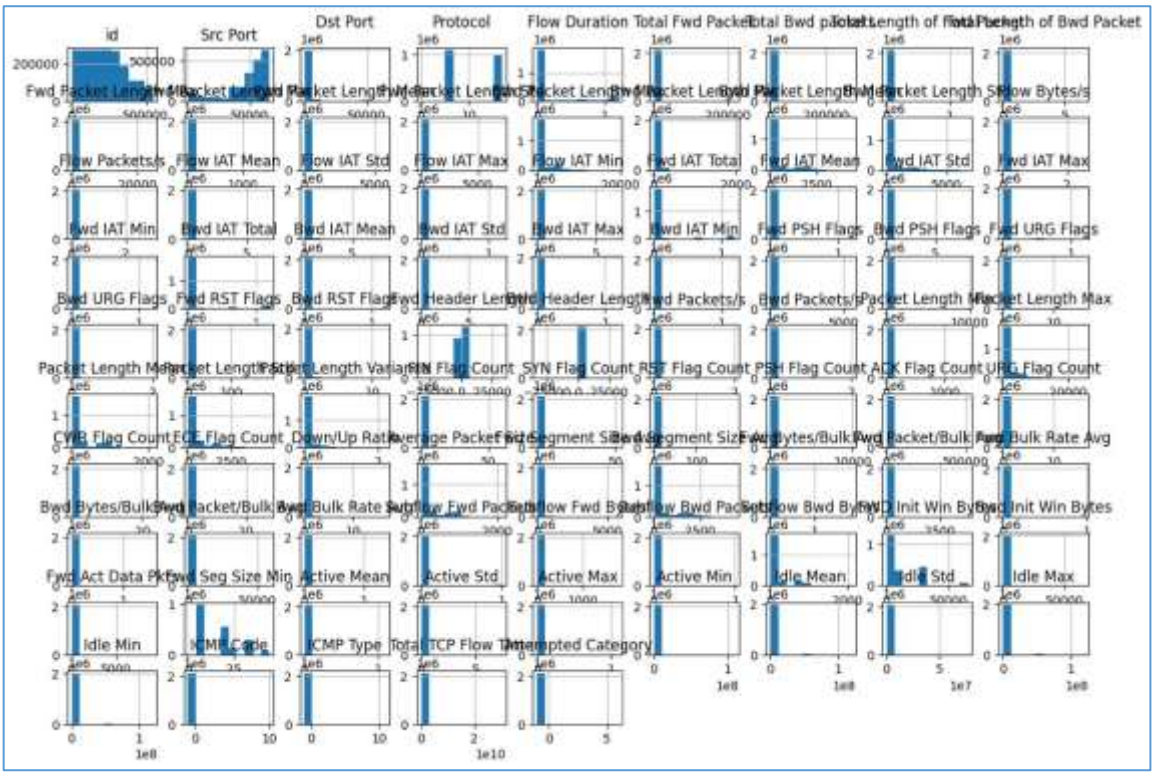
 ISSN:2302-8149
e-ISSN:2540-9719

```
# ========================
# 2. MISSING VALUE
# ========================
print("cek missing value")
print(df.isnull().sum().sort_values(ascending=False).head())
```

```
cek missing value
id                      0
Bwd Packet/Bulk Avg     0
Fwd Bulk Rate Avg       0
Fwd Packet/Bulk Avg     0
Fwd Bytes/Bulk Avg      0
dtype: int64
```

Gambar 4 Hasil pengecekan missing value

Untuk memahami distribusi data pada setiap fitur, dilakukan visualisasi menggunakan histogram sebagaimana ditunjukkan pada Gambar 5. Hasil visualisasi menunjukkan bahwa sebagaian besar fitur memiliki distribusi yang tidak merata dan cenderung tidak simetris. Selain itu, beberapa fitur memiliki nilai yang sangat tinggi dibandingkan fitur lainnya, sehingga mengindikasikan adanya perbedaan skala data dan potensi outlier. Kondisi ini dapat mempengaruhi performa model dalam proses pelatihan, sehingga, diperlukan tahapan preprocessing seperti normalisasi dan feature scaling untuk meningkatkan stabilitas model.



Gambar 5 Distribusi nilai pada setiap fitur

Tahap selanjutnya yaitu preprocessing dilakukan untuk memastikan data dalam kondisi bersih dan siap digunakan dalam proses pelatihan model.

Langkah pertama Adalah mengubah seluruh fitur ke dalam tipe numerik. Selanjutnya dilakukan penanganan terhadap nilai infinite (∞) dengan mengubahnya menjadi nilai NaN. Setelah itu, dilakukan proses imputasi menggunakan metode mean untuk menggantikan nilai yang hilang.

Hasil akhir preprocessing menunjukkan bahwa seluruh data telah bersih nilai NaN dan infinite. Hal ini ditunjukkan pada Gambar 6, dimana jumlah nilai NaN dan infinite telah menjadi nol.

 ISSN:2302-8149
e-ISSN:2540-9719

```
#hapus NAN
# PAKSA ke float
X_train = np.array(X_train, dtype=np.float64)
X_test = np.array(X_test, dtype=np.float64)

# Ganti inf → NaN
X_train[np.isinf(X_train)] = np.nan
X_test[np.isinf(X_test)] = np.nan

# Imputasi
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# CEK FINAL (WAJIB 0 semua)
print("NaN:", np.isnan(X_train).sum())
print("Inf:", np.isinf(X_train).sum())
```

```
NaN: 0
Inf: 0
```

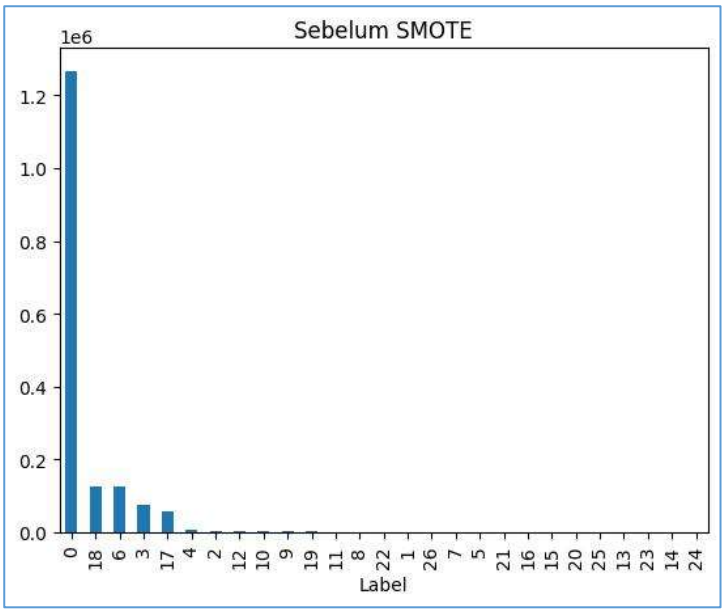
Gambar 6 Hasil pembersihan data NaN dan infinite

Untuk mengurangi ketidakseimbangan distribusi kelas, dilakukan proses oversampling pada data pelatihan sehingga jumlah data antar kelas menjadi lebih proporsional. Metode ini bekerja dengan menghasilkan data sintetis pada kelas minoritas agar distribusi data menjadi lebih seimbang.

Pada penelitian ini digunakan parameter **K_neighbors = 1** untuk menyesuaikan dengan jumlah data pada kelas minoritas yang sangat sedikit.

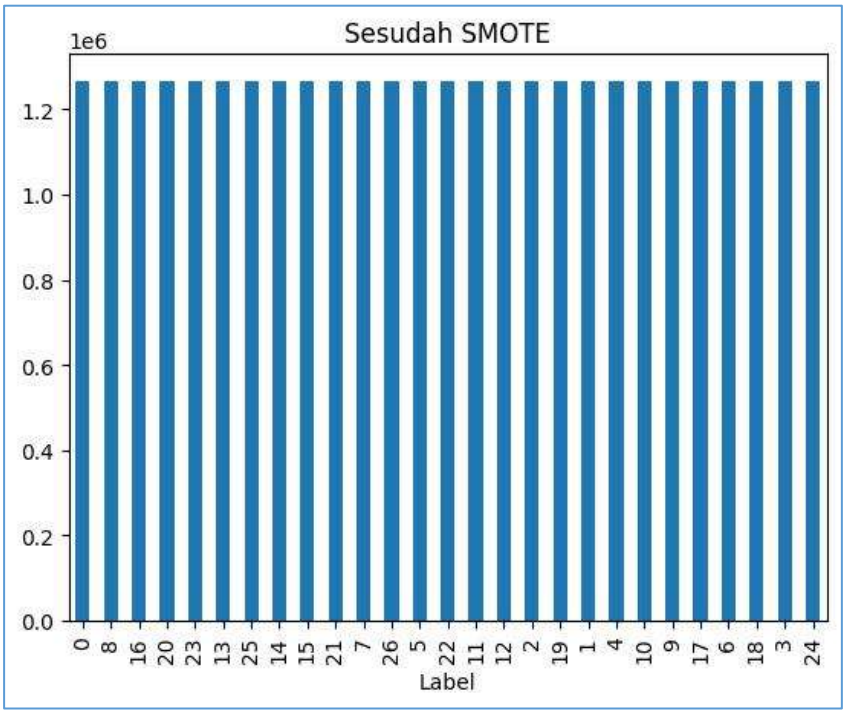
Setelah penerapan SMOTE, jumlah data training meningkat secara signifikan,yang menunjukkan bahwa metode ini berhasil menyeimbangkan distribusi data.

Hal ini ditunjukkan pada Gambar 7, 8 sebelum dan sesudah SMOTE.



Gambar 7 Data sebelum SMOTE

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 8 Data sesudah SMOTE

Pada penelitian ini, Autoencoder dimanfaatkan untuk menghasilkan representasi fitur yang lebih ringkas dari data jaringan. Proses ekstraksi fitur dilakukan melalui tahapan encoding sehingga data dengan dimensi tinggi dapat direpresentasikan ke bentuk fitur yang lebih efisien tanpa menghilangkan informasi penting. Hasil ekstraksi fitur tersebut kemudian digunakan sebagai input pada proses klasifikasi menggunakan CNN, hal ini bisa dilihat pada Gambar 9, yang menunjukkan hasil output dari proses encoding.

```
X_train_ae = encoder.predict(X_train_res)
X_test_ae  = encoder.predict(X_test)

print(X_train_ae.shape)
```

```
1068232/1068232 [==============================] - 266s 248us/step
13125/13125 [==============================] - 3s 238us/step
(34183404, 32)
```

Gambar 9 Hasil ekstraksi fitur menggunakan AE

Data hasil ekstraksi fitur kemudian diubah ke dalam bentuk matriks dua dimensi dengan ukuran (4x 8 x 1) agar dapat digunakan sebagai input pada model CNN. Proses pembentukan model CNN ditunjukkan pada Gambar 10, yang terdiri dari beberapa lapisan konvolusi, pooling, dan fully connected layer untuk melakukan klasifikasi data. Model yang digunakan merupakan model klasifikasi multi-kelas dengan total 27 kelas output.

Selanjutnya, proses pelatihan model CNN dilakukan menggunakan data hasil preprocessing dan balancing, sebagaimana ditunjukkan pada Gambar 11. Model dilatih menggunakan optimizer adam dengan fungsi loss sparse categorical crossentropy dan metrik evaluasi accuracy, serta menggunakan data validasi untuk memantau performa model selama proses pelatihan.

 ISSN:2302-8149
e-ISSN:2540-9719

```
#membangun model cnn

model = Sequential([
    Conv2D(32, (2,2), activation='relu', padding='same', input_shape=(4,8,1)),
    MaxPooling2D((2,2)),

    Conv2D(64, (2,2), activation='relu', padding='same'),
    MaxPooling2D((2,2)),

    Flatten(),
    Dense(64, activation='relu'),
    Dense(27, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Gambar 10 Arsitektur model CNN

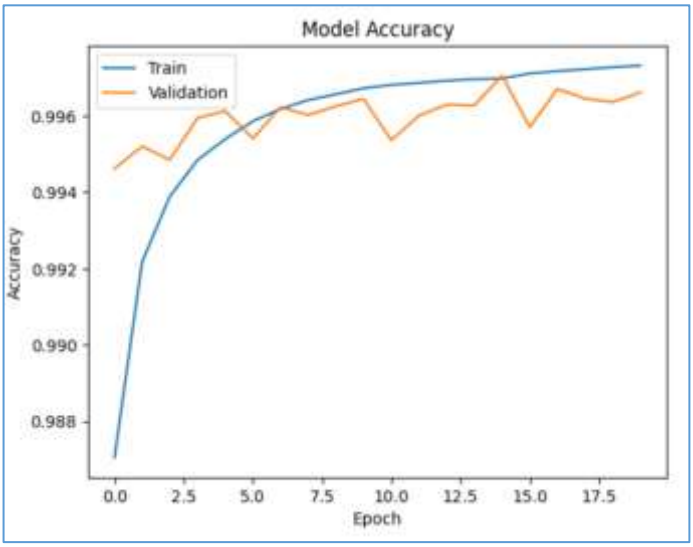
```
#training model cnn

history = model.fit(
    X_train_cnn, y_train_res,
    epochs=20,
    batch_size=128,
    validation_data=(X_test_cnn, y_test)
)
```

Gambar 11 Training model CNN

Hasil training model menunjukkan bahwa model memilki performa yang baik. Akurasi training meningkat hingga mencapai sekitar 99.7%, sedangkan akurasi validasi mencapai sekitar 99.6%.

Perkembangan akurasi model selama proses training ditunjukkan pada Gambar 12.

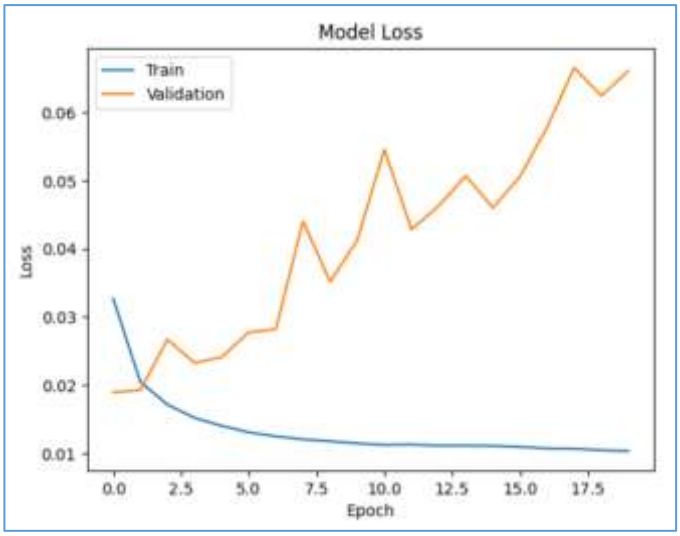


Gambar 12 Grafik akurasi model

Namun, jika dilihat dari nilai loss, terdapat perbedaan antara training loss dan validation loss. Training loss terus menurun, sedangkan validation loss cenderung meningkat seiring bertambahnya epoch.

 ISSN:2302-8149
e-ISSN:2540-9719

Hal ini dapat dilihat pada Gambar 13 yang menunjukkan ada indikasi overfitting pada model.



Gambar 13 Grafik loss model selama training

Evaluasi model dilakukan menggunakan classification report untuk mengukur performa model pada data testing. Hasil evaluasi menunjukkan bahwa model memilki Tingkat akurasi yang sangat tinggi, selain itu, nilai weighted average juga menunjukkan performa yang baik.

Namun nilai macro average lebih rendah, yang menunjukkan bahwa model kurang optimal dalam menangani kelas minoritas. Hasil evaluasi model ditunjukkan pada Gambar 14.

```
#evaluasi model

y_pred = model.predict(X_test_cnn)
y_pred = y_pred.argmax(axis=1)

print(classification_report(y_test, y_pred))
```

```
13125/13125 [==============================] - 6s 427us/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    316514
           1       0.97      1.00      0.98       147
           2       1.00      1.00      1.00       813
           3       1.00      1.00      1.00     19029
           4       0.99      1.00      0.99      1514
           5       1.00      1.00      1.00        16
           6       1.00      1.00      1.00     31694
           7       0.99      0.97      0.98       116
           8       0.97      0.99      0.98       348
           9       1.00      1.00      1.00       674
          10       0.99      1.00      0.99       772
          11       0.99      0.99      0.99       369
          12       1.00      1.00      1.00       794
          13       1.00      1.00      1.00         2
          14       1.00      1.00      1.00         2
          15       0.38      0.43      0.40         7
          16       0.82      1.00      0.90         9
          17       0.93      0.98      0.96     14354
          18       0.99      0.97      0.98     31813
          19       0.99      1.00      0.99       592
          20       1.00      1.00      1.00         5
          21       0.94      1.00      0.97        15
          22       0.82      0.80      0.81       258
          23       0.18      0.67      0.29         3
          24       0.00      0.00      0.00         1
          25       1.00      0.75      0.86         4
          26       0.64      0.68      0.66       131

    accuracy                           1.00    419996
   macro avg       0.87      0.90      0.88    419996
weighted avg       1.00      1.00      1.00    419996
```

Gambar 14 Hasil evaluasi model

 ISSN:2302-8149
e-ISSN:2540-9719

Distribusi hasil prediksi menunjukkan bahwa model cenderung lebih akurat dalam memprediksi kelas dengan jumlah data besar, seperti BENIGN dan DoS Hulk. Sebaliknya, pada kelas dengan

```
print("\nDistribusi hasil prediksi:")
for i, label in enumerate(le.classes_):
    print(f"{label} ({i}) = {np.bincount(y_pred)[i]}")
```

```
Distribusi hasil prediksi:
BENIGN (0) = 316406
Botnet (1) = 151
Botnet - Attempted (2) = 812
DDoS (3) = 19039
DoS GoldenEye (4) = 1522
DoS GoldenEye - Attempted (5) = 19
DoS Hulk (6) = 31704
DoS Hulk - Attempted (7) = 113
DoS Slowhttptest (8) = 346
DoS Slowhttptest - Attempted (9) = 672
DoS Slowloris (10) = 772
DoS Slowloris - Attempted (11) = 367
FTP-Patator (12) = 794
FTP-Patator - Attempted (13) = 6
Heartbleed (14) = 2
Infiltration (15) = 9
Infiltration - Attempted (16) = 10
Infiltration - Portscan (17) = 14765
Portscan (18) = 31463
SSH-Patator (19) = 597
SSH-Patator - Attempted (20) = 7
Web Attack - Brute Force (21) = 15
Web Attack - Brute Force - Attempted (22) = 258
Web Attack - SQL Injection (23) = 4
Web Attack - SQL Injection - Attempted (24) = 4
Web Attack - XSS (25) = 3
Web Attack - XSS - Attempted (26) = 136
```

Gambar 15 Distribusi hasil prediksi

jumlah data kecil, jumlah prediksi juga relatif kecil. Hal ini dapat dilihat pada Gambar 15 yang menunjukkan hasil prediksi.

Pada tahap evaluasi, model menunjukkan performa yang sangat baik dengan akurasi mendekati 1.00 serta nilai precision, recall, dan f1-score yang tinggi pada sebagian besar kelas, terutama kelas mayoritas seperti BENIGN, DoS Hulk, dan Portscan.

Namun, pada beberapa kelas minoritas, performa model masih lebih rendah, yang ditunjukkan oleh nilai f1-score yang tidak stabil. Hal ini disebabkan oleh ketidakseimbangan data, sehingga model cenderung lebih optimal dalam mengenali kelas dengan jumlah data besar.

Berdasarkan distribusi hasil prediksi, terlihat bahwa lebih banyak memprediksi kelas mayoritas dibandingkan kelas minoritas. Meskipun teknik SMOTE telah diterapkan, pengaruh ketidakseimbangan data masih terlihat.

Secara keseluruhan, model berbasis Autoencoder dan CNN berhasil memberikan performa klasifikasi yang sangat tinggi, namun masih perlu peningkatan dalam mendeteksi kelas dengan jumlah data yang sangat sedikit.

5 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dapat disimpulkan bahwa model klasifikasi malware jaringan berbasis kombinasi Convolutional Neural Network (CNN) dan Autoencoder mampu memberikan performa yang baik. Proses ekstraksi fitur menggunakan Autoencoder berhasil mereduksi dimensi data dari 91 fitur menjadi 32 fitur tanpa menghilangkan informasi penting, sehingga meningkatkan efisiensi dan efektifitas proses klasifikasi. Hasil evaluasi menunjukkan bahwa model mencapai tingkat akurasi yang tinggi, yaitu sebesar 99,7% pada data penelitian dan 99,6% pada

 ISSN:2302-8149
e-ISSN:2540-9719

data validasi, dengan nilai *precision, recall, F1-score* yang baik pada sebagian besar kelas, terutama pada kelas mayoritas seperti BENIGN, Dos Hulk, dan portscan. Namun demikian performa model, pada beberapa kelas minoritas masih belum optimal akibat ketidakseimbangan data, meskipun teknik SMOTE telah diterapkan. Oleh karena itu penelitian selanjutnya disarankan untuk menggunakan teknik penanganan data tidak seimbang yang lebih advanced, seperti *adaptive synthetic sampling* atau *cost-sensitive learning*, serta mengeksplorasi arsitektur model yang lebih kompleks seperti kombinasi dengan transformer atau model ensemble guna meningkatkan performa pada kelas minoritas dan memperkuat kemampuan generalisasi model.

Referensi

[1] S. A. Hashmi, "*Malware Detection and Classification on Different Dataset by Hybridization of CNN and Machine Learning*," International Journal of Intelligent Systems and Applications in Engineering, Vol. 12, No. 6s, pp. 650–667, 2024, DOI: 10.18201/ijisae.2024.006s.4004.

[2] Sharipuddin, R. S. Putra, M. F. Aulia, S. A. Maulana, and P. A. Jusia, "*Android Malware Detection using Convolutional Neural Network*," Media Journal of General Computer Science, Vol. 1, No. 1, pp. 7–13, 2024, DOI: 10.62205/mjgcs.v1i1.7.

[3] R. Almuhanna and S. Dardouri, "*A Deep Learning/Machine Learning Approach for Anomaly based Network Intrusion Detection*," Frontiers in Artificial Intelligence, Vol. 8, 2025, DOI: 10.3389/frai.2025.1625891.

[4] L. Mohammadpour, T. C. Ling, C. S. Liew, and A. Aryanfar, "*A Survey of CNN-based Network Intrusion Detection*," Applied Sciences, Vol. 12, No. 16, p. 8162, 2022, DOI: 10.3390/app12168162.

[5] B. H. Egga, A. S. Audu, G. O. I. Aimufua, M. Olalere, B. A. Ajayi, and I. T. Solomon, "*Autoencoder-based Model for Detecting IoT Network Traffic Anomalies*," Science World Journal, Vol. 20, No. 4, 2025, DOI: 10.4314/swj.v20i4.8.

[6] K. Janani and R. Gunasundari, "*Detection of Malware in Large Networks using Deep Auto Encoders*," International Journal on Recent and Innovation Trends in Computing and Communication, Vol. 11, No. 6s, 2023, DOI: 10.17762/ijritcc.v11i6s.6894.

[7] R. Jablaoui, O. Cheikhrouhou, M. Hamdi, and N. Liouane, "*Deep Learning Enabled Intrusion Detection System for IoT Security*," EURASIP Journal on Wireless Communications and Networking, Vol. 2025, No. 66, 2025, DOI: 10.1186/s13638-025-02477-6.

[8] E. C. P. Neto, S. Iqbal, S. Buffett, M. Sultana, and A. Taylor, "*Deep Learning for Intrusion Detection in Emerging Technologies: A Comprehensive Survey and New Perspectives*," Artificial Intelligence Review, Vol. 58, No. 340, 2025, DOI: 10.1007/s10462-025-11346-z.

[9] A. H. A. Alsaroah, "*Anomaly based Network Intrusion Detection using Autoencoders*," Journal of Al-Qadisiyah for Computer Science and Mathematics, Vol. 18, No. 1, pp. 24–36, 2026, DOI: 10.29304/jqcsm.2026.18.12540.

[10] Z. Zhao, H. Guo, and Y. Wang, "*A Multi-Information Fusion Anomaly Detection Model based on Convolutional Neural Networks and AutoEncoder*," Scientific Reports, Vol. 14, 2024, DOI: 10.1038/s41598-024-66760-0.

[11] M. A. Hossain and M. S. Islam, "*Enhanced Detection of Obfuscated Malware in Memory Dumps: A Machine Learning Approach for Advanced Cybersecurity*," Cybersecurity, Vol. 7, No. 16, 2024, DOI: 10.1186/s42400-024-00205-z.

[12] S. B. Selvakumar, M. Sivaanandh, K. Muneeswaran, and B. Lakshmanan, "*Ensemble of Feature Augmented Convolutional Neural Network and Deep Autoencoder for Efficient Detection of Network Attacks*," Scientific Reports, Vol. 15, No. 1, 2025, DOI: 10.1038/s41598-025-88243-6.

[13] N. Cassavia, L. Caviglione, M. Guarascio, A. Liguori, and M. Zuppelli, "*Learning Autoencoder Ensembles for Detecting Malware Hidden Communications in IoT Ecosystems*," Journal of Intelligent Information Systems, Vol. 62, pp. 925–949, 2024, DOI: 10.1007/s10844-023-00819-8.

[14] M. M. Aslam, A. Tufail, L. C. De Silva, R. A. A. H. M. Apong, and A. Namoun, "*An Improved Autoencoder-based Approach for Anomaly Detection in Industrial Control Systems*," Systems Science & Control Engineering, Vol. 12, No. 1, 2024, DOI: 10.1080/21642583.2024.2334303.

[15] A. Souri, R. Hosseini, and M. Rahmani, "*A State-of-the-Art Survey of Malware Detection*

ISSN:2302-8149
e-ISSN:2540-9719

Approaches using Data Mining Techniques," Human-Centric Computing and Information Sciences, Vol. 10, No. 30, 2020, DOI: 10.1186/s13673-020-00228-0.

[16] A. Anand, S. Rani, D. Anand, H. M. Aljahdali, and D. Kerr, "*An Efficient CNN-based Deep Learning Model to Detect Malware Attacks (CNN-DMA) in 5G-IoT Healthcare Applications*," Sensors, Vol. 21, No. 19, p. 6346, 2021, DOI: 10.3390/s21196346.