Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2025128267
Vol. 12, No. 1, Februari 2025, hlm. 21-30 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

DEEP LEARNING DENGAN TEKNIK *EARLY STOPPING* UNTUK MENDETEKSI *MALWARE* PADA PERANGKAT *IOT*

Iwang Moeslem Andika Surya[*1], Triawan Adi Cahyanto[2] Lutfi Ali Muharom[3]

[1,2,3]Universitas Muhammadiyah Jember, Kabupaten Jember
Email: [1]moeslemiwang@gmail.com, [2]triawanac@unmuhjember.ac.id, [3]lutfi.muharom@unmuhjember.ac.id
*Penulis Korespondensi

(Naskah masuk: 13 Desember 2023, diterima untuk diterbitkan: 07 Februari 2025)

Abstrak

Perkembangan pesat teknologi, khususnya *Internet of Things* (IoT), telah memberikan dampak signifikan dalam berbagai sektor kehidupan manusia. IoT memungkinkan pertukaran data antar perangkat secara otomatis melalui jaringan internet, mengubah cara manusia berinteraksi dengan lingkungan sekitarnya. Meskipun IoT memberikan berbagai manfaat seperti kemudahan mengakses perangkat dari jarak jauh, kehadirannya juga membawa potensi bahaya terkait dengan keamanan siber, privasi, dan ketergantungan terhadap teknologi. Artikel ini membahas upaya untuk mengatasi ancaman keamanan siber pada ekosistem IoT dengan mengimplementasikan sistem deteksi berbasis klasifikasi *malware*. Pendekatan ini memanfaatkan teknik pembelajaran mesin, terutama *deep learning*, untuk mengidentifikasi dan memitigasi ancaman siber. Penggunaan *Convolutional Neural Network* (CNN) dalam klasifikasi dataset IoT-23 dengan teknik penyeimbangan data SMOTE (*Synthetic Minority Oversampling Technique*) serta fungsi *Early Stopping* menunjukkan hasil yang tinggi. Meskipun CNN awalnya dirancang untuk pengolahan gambar, algoritma ini juga efektif dalam mendeteksi pola kompleks pada data non-gambar seperti lalu lintas jaringan IoT karena kemampuannya dalam ekstraksi fitur hierarkis. Akurasi yang diperoleh pada dataset tidak seimbang adalah sebesar 99%, sedangkan pada dataset seimbang sebesar 75%. Presisi dan Recall yang diperoleh pada dataset tidak seimbang di kelas 0 adalah 100% dan 35%, sedangkan pada kelas 1 adalah 99% dan 100%. Pada dataset seimbang, presisi dan recall di kelas 0 adalah 67% dan 100%, sedangkan pada kelas 1 adalah 100% dan 51%.

Kata kunci: *Malware, Pembelajaran Mendalam, SMOTE, Early Stopping, Convolution Neural Network.*

DEEP LEARNING WITH EARLY STOPPING TECHNIQUE FOR MALWARE DETECTION ON IOT DEVICES

Abstract

The rapid development of technology, especially the Internet of Things (IoT), has significantly impacted various sectors of human life. IoT enables automatic data exchange between devices via the internet, changing how humans interact with their surroundings. Although IoT provides multiple benefits, such as ease of accessing devices remotely, its presence also brings potential dangers related to cybersecurity, privacy, and dependence on technology. This article discusses efforts to address cybersecurity threats in the IoT ecosystem by implementing a malware classification-based detection system. This approach utilizes machine learning techniques and intense learning to identify and mitigate cyber threats. The use of Convolutional Neural Network (CNN) in classifying the IoT-23 dataset with the SMOTE (Synthetic Minority Oversampling Technique) data balancing technique and the Early Stopping function shows high results. Although CNN was initially designed for image processing, this algorithm also effectively detects complex patterns in non-image data, such as IoT network traffic, due to its ability to extract hierarchical features. The accuracy obtained on the unbalanced dataset is 99%, while on the balanced dataset, it is 75%. The precision and recall obtained on the unbalanced dataset in class 0 are 100% and 35%, while in class 1 are 99% and 100%. In the balanced dataset, the precision and recall in class 0 are 67% and 100%, while in class 1 are 100% and 51%.

Keywords: *Malware, Deep Learning, SMOTE, Early Stopping, Convolution Neural Network.*

1. PENDAHULUAN

IoT adalah sebuah sistem yang mengumpulkan data menggunakan sensor, memproses informasi melalui interaksi, dan menghasilkan keluaran untuk melayani kebutuhan bersama (Sorri, Mustafee and Seppänen, 2022). Perkembangan pesat teknologi,

khususnya IoT, telah memberikan dampak signifikan dalam berbagai aspek kehidupan manusia. IoT memungkinkan perangkat bertukar data secara otomatis melalui jaringan internet, mengubah cara manusia berinteraksi dengan lingkungan sekitarnya (Cahyanto *et al.*, 2022). Meskipun IoT menawarkan berbagai manfaat seperti kemudahan mengakses perangkat dari jarak jauh, keberadaannya juga menimbulkan potensi risiko terkait dengan keamanan siber, privasi, dan ketergantungan terhadap teknologi.

Klasifikasi *malware* pada perangkat IoT menghadapi beberapa tantangan utama. Pertama, beragamnya jenis *malware* yang mencakup berbagai bentuk seperti *virus*, *worm*, *trojan*, hingga *ransomware*, yang masing-masing memiliki karakteristik unik yang memerlukan pendekatan khusus dalam deteksinya (Zhang *et al.*, 2019). Kedua, *volume* data yang dihasilkan oleh perangkat IoT sangat besar dan bervariasi, sehingga diperlukan teknik klasifikasi yang mampu menangani big data dan ekstraksi fitur yang kompleks (Sabah *et al.*, 2019). Ketiga, dalam *dataset malware*, sering kali jumlah data *benign* jauh lebih banyak dibandingkan dengan data yang mengandung *malware*. Ketidakseimbangan kelas ini dapat menyebabkan masalah dalam akurasi model klasifikasi (Jain, Ratnoo and Kumar, 2020).

Untuk mengatasi tantangan-tantangan tersebut, penelitian ini mengusulkan penggunaan *Convolutional Neural Network* (CNN) sebagai algoritma *deep learning* dalam deteksi *malware* pada perangkat IoT. CNN dipilih karena beberapa alasan kuat. Pertama, CNN memiliki kemampuan untuk mengekstraksi fitur hierarkis dari data, yang memungkinkan model untuk mengenali pola kompleks dalam data jaringan IoT (Lecun, Bengio and Hinton, 2015). Kedua, meskipun CNN awalnya dirancang untuk pengolahan gambar, penelitian telah menunjukkan bahwa CNN juga efektif dalam analisis data non-gambar, seperti data lalu lintas jaringan, karena kemampuannya dalam menangani berbagai jenis data (Bakar and Adiwijaya, 2021). Ketiga, CNN telah terbukti memberikan kinerja yang tinggi dalam berbagai aplikasi klasifikasi, termasuk deteksi *malware*, dengan akurasi yang lebih baik dibandingkan dengan teknik pembelajaran mesin tradisional (Yuan *et al.*, 2020).

Deep learning menurut (Libovický, 2017) adalah subbidang dalam *Machine learning* yang mencakup penggunaan jaringan saraf tiruan (*neural networks*) dengan banyak lapisan (yang dikenal sebagai *deep neural networks)* untuk mengekstraksi pola dan fitur dari data yang kompleks dan besar. Proses pembuatan sistem deteksi ini melibatkan *deep learning*, sebuah model jaringan dengan lapisan-lapisan yang dapat mengenali ciri-ciri dan tanda-tanda khas dari *malware*. *Deep learning*, khususnya *Convolutional Neural Network (CNN*), telah terbukti berhasil dalam klasifikasi dataset *malware*, meskipun dirancang untuk tugas pengolahan gambar. Namun, proses pelatihan pada *deep learning*, terutama dengan dataset besar, memakan waktu yang lama. Teknik *Early Stopping* dapat diterapkan untuk meningkatkan efisiensi waktu dan mengurangi masalah *overfitting* pada klasifikasi *deep learning* (Zoumpekas, Salamó and Puig, 2022; Al-rimy et al., 2023).

Early Stopping adalah suatu metode yang diterapkan selama pelatihan *neural network* dengan tujuan mencegah terjadinya *overfitting* dan mengurangi waktu yang diperlukan dalam proses pelatihan (Fujo, Subramanian and Khder, 2022). Teknik *Early Stopping* dianggap bermanfaat karena dapat menghentikan pelatihan model saat sudah mencapai kinerja optimal pada data validasi. Selama proses pelatihan, evaluasi dilakukan pada performa model terhadap data validasi pada setiap *epoch*. *validation Loss* menjadi parameter untuk mengukur sejauh mana model dapat berperforma baik pada data baru.

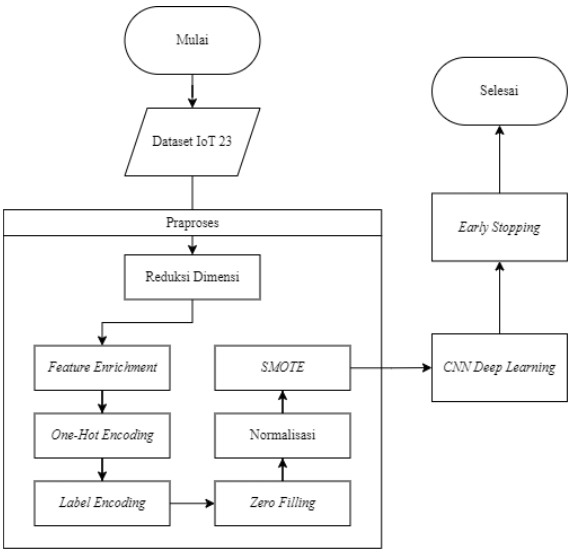
Penelitian ini bertujuan untuk mengembangkan sistem deteksi berbasis klasifikasi menggunakan dataset IoT23 oleh (Liang and Vankayalapati, 2021). Dengan menerapkan *CNN* dan teknik *Early Stopping*, penelitian ini berupaya meningkatkan akurasi deteksi *malware* dibandingkan dengan penelitian sebelumnya (Liang and Vankayalapati, 2021). Melalui eksperimen dan perbandingan, diharapkan penelitian ini dapat memberikan kontribusi pada pengembangan sistem deteksi keamanan siber di lingkungan *IoT*.

2. METODE PENELITIAN

Demi mencapai tujuan penelitian, terdapat tahapan-tahapan penelitian yang dilakukan secara berurutan dan sistematis.

2.1 Tahapan Penelitian

Penelitian ini dilakukan melalui beberapa tahapan utama yang ditunjukkan pada diagram berikut:



Gambar 1. Diagram Tahapan Penelitian

2.2 Dataset *IoT-23*

Dataset pada penelitian ini mengacu pada hasil penelitian (Garcia, Parmisano and Erquiaga, 2020) yang disebut "IoT 23". IoT-23 adalah himpunan data terbaru yang berisi lalu lintas jaringan dari perangkat IoT. Dataset ini mencakup 20 rangkuman dari *malware* yang dijalankan di simulasi perangkat IoT dan 3 rangkuman dari lalu lintas perangkat IoT asli yang bersifat *benign*. Data yang digunakan berjumlah satu juta baris dan telah dilabeli. Tabel berikut adalah daftar atribut dari dataset IoT-23: (Liang and Vankayalapati, 2021).

Tabel 1. Daftar fitur

No	Variabel	Keterangan
1	*ts*	*Timestamp* atau keterangan waktu
2	*uid*	*Unique identifier* dari sebuah koneksi
3	*proto*	Protokol yang digunakan dalam koneksi
4	*service*	Tipe layanan atau aplikasi yang terlibat
5	*id.orig_h*	*Identifier.origin_host* mewakili sumber alamat IP
6	*id.orig_p*	*Identifier.origin_port* mewakili port sumber dari koneksi
7	*id.resp_h*	*Identifier.response_host* mewakili tujuan alamat IP
8	*id.resp_p*	*Identifier.response_port* mewakili tujuan port dari koneksi
9	*duration*	Durasi dari koneksi yang berjalan
10	*orig_bytes*	Jumlah data yang dikirim dari sumber (origin)
11	*resp_bytes*	Jumlah data yang dikirim sebagai respon
12	*conn_state*	Keadaan atau status dari sebuah koneksi
13	*local_orig*	Sumber koneksi lokal (jika benar maka True)
14	*local_resp*	Sumber respon lokal (jika benar maka True)
15	*missed_bytes*	Jumlah data yang hilang selama koneksi berjalan
16	*history*	Rekam jejak dari koneksi
17	*orig_pkts*	Jumlah paket yang dikirim dari sumber (origin)
18	*orig_ip_bytes*	Jumlah *IP* yang dikirim dari sumber (origin)
19	*resp_pkts*	Jumlah paket yang dikirim sebagai respon
20	*resp_ip_bytes*	Jumlah *IP* yang dikirim sebagai respon

Terdapat 6 label tipe *malware* pada dataset ini dengan keterangan seperti berikut:

Tabel 1. Jumlah data pada setiap label

No.	Label	Jumlah data
1	*Part Of A horizontal Por tScan*	607214
2	*Okiru*	248458
3	*Benign*	19257
4	*DDoS*	124943
5	C&C-HeartBeat	98
6	C&C-Torii	30

2.3 Praproses

Sebelum dapat digunakan untuk pembuatan model, data harus melalui beberapa tahapan praproses. Semua proses praproses dalam penelitian ini dilakukan di dalam *Jupyter Notebook* dengan memanfaatkan berbagai pustaka yang memadai seperti *pandas* dan *numpy*.

2.3.1 Reduksi Dimensi

Menurut beberapa peneliti seperti (Liang and Vankayalapati, 2021; Van Dartel, 2021), fitur seperti "*ts, uid, id.orig_h, local_orig, local_resp, missed_bytes,* dan *tunnel_parents*" dihapus karena tidak menunjukkan korelasi signifikan terhadap label dan banyak nilai kosong.

Tabel 2. Daftar kolom pada dataset

kolom yang dipakai	kolom yang dihapus
proto, duration, orig_bytes, resp_bytes, conn_state, orig_pkts, orig_ip_bytes, resp_pkts, resp_ip_bytes,	*ts, uid, id.orig_h, local_orig, local_resp, missed_bytesand tunnel_parents, history*

2.3.2 *Feature Enrichment*

Feature Enrichment, atau peningkatan fitur, diterapkan dengan cara menambahkan fitur ke dataset dari sumber internal atau eksternal. Pada konteks ini, peningkatan fitur dilakukan secara internal, di mana seluruh data diurutkan berdasarkan kolom *timestamp* (*ts*) sebelum kolom tersebut dihapus.

Tabel 3. Waktu diurutkan secara *ascending*

ts (sebelum diurutkan)	*ts* (sesudah diurutkan)
2023-09-01 08:45:00	2023-09-01 08:00:00
2023-09-01 08:30:00	2023-09-01 08:15:00
2023-09-01 08:00:00	2023-09-01 08:30:00
2023-09-01 08:15:00	2023-09-01 08:45:00

Feature enrichment perlu dilakukan untuk menambah fitur baru ke dalam dataset yang dapat memberikan informasi tambahan yang relevan dan meningkatkan performa model analisis atau prediksi. Langkah-langkah teknisnya melibatkan pengurutan data berdasarkan kolom *timestamp* (ts) secara *ascending*, agar memungkinkan untuk menghitung selisih waktu antara setiap pasangan *timestamp* berturut-turut. Setelah pengurutan, *timestamp* pertama akan memiliki nilai selisih yang tidak terdefinisi (NaN), sedangkan selisih waktu antara *timestamp* berikutnya dihitung sebagai perbedaan waktu dalam satuan menit atau detik. Nilai selisih ini kemudian disimpan dalam kolom baru yang disebut "*time_diff*".

Tabel 4. Selisih per *timestamp*

ts (*timestamp*)	*time_diff*
2023-09-01 08:00:00	*NaN*
2023-09-01 08:15:00	15 menit
2023-09-01 08:30:00	15 menit
2023-09-01 08:45:00	15 menit

Proses ini tidak hanya memperkaya dataset dengan fitur baru yang mungkin relevan untuk analisis temporal, tetapi juga dapat digunakan untuk mendeteksi pola periodik, lonjakan aktivitas, atau

anomali yang bergantung pada interval waktu, sehingga memberikan model analisis atau prediksi lebih banyak informasi untuk menghasilkan hasil yang lebih akurat dan relevan.

2.3.3 *One-hot Encoding*

Proses pelatihan model dalam penelitian ini terbatas pada data numerik, sehingga diperlukan tahap *one-hot encoding* pada kolom-kolom fitur yang memiliki nilai bukan numerik seperti kolom *proto (protocol)* dan *conn_state (connection state)*. Penerapan teknik ini melibatkan penciptaan kolom-kolom baru yang menyimpan nilai-nilai *encoding*, sehingga jumlah total kolom akan meningkat setelah proses *encoding* dilakukan. Dengan *one-hot encoding*, setiap nilai unik dalam kolom kategori diubah menjadi kolom baru yang bersesuaian. Setiap kolom baru ini akan memiliki nilai biner (0 atau 1) yang menunjukkan ada tidaknya kategori tertentu dalam sampel data tersebut.

Contohnya, kolom "proto" yang awalnya berisi nilai-nilai kategori seperti "tcp", "udp", dan "icmp" diubah menjadi kolom biner seperti "proto_tcp", "proto_udp", dan "proto_icmp". Jika nilai asli adalah "tcp", maka kolom "proto_tcp" akan bernilai 1, dan kolom "proto_udp" serta "proto_icmp" akan bernilai 0. Hal ini membantu model memahami bahwa ketiga protokol ini adalah fitur yang berbeda tanpa memberikan bobot numerik yang tidak sesuai jika dikonversi secara langsung.

Tabel 5. *One-hot encoding* pada kolom *proto*

index	***proto***	***proto_tcp***	***proto_udp***	***proto_icmp***
0	*tcp*	***true***	*false*	*false*
1	*udp*	*false*	***true***	*false*
2	*icmp*	*false*	*false*	***true***

2.3.4 *Label Encoding*

Tahap praproses berikutnya melibatkan *label encoding*. Teknik praproses ini mengubah enam label diubah menjadi dua, yakni *Malware* dan *Benign*. Pendekatan ini bertujuan untuk mengurangi durasi pelatihan dan kompleksitas model. Proses label *encoding* ini secara spesifik mengubah seluruh nilai dalam kolom "label" menjadi 1, kecuali untuk nilai *benign* yang diubah menjadi 0.

Tabel 6. *Label encoding*

sebelum	***sesudah***
part of a horizontal port scan	1
miraii	1
benign	0
...	…
benign	0

2.3.5 *Zero Filling*

Tahap *zero filling* dalam *preprocessing* dilakukan untuk memastikan semua sel dalam dataset memiliki nilai yang dapat diproses oleh model pembelajaran mesin. Sel kosong atau berisi karakter spesial seperti '-' tidak dapat dihitung oleh model, sehingga diisi dengan angka 0 untuk menghindari kesalahan perhitungan dan menjaga konsistensi data. *Zero filling* juga memenuhi kebutuhan model yang hanya bisa menangani data numerik, menyederhanakan implementasi *preprocessing*, dan memastikan kelengkapan data. Dengan mengisi sel kosong dengan 0, dataset menjadi siap untuk dianalisis dan diproses lebih lanjut oleh model pembelajaran mesin.

2.3.6 Normalisasi *Min-Max*

Normalisasi *MinMax* menurut (Azzahra Nasution, Khotimah and Chamidah, 2019) merupakan sebuah transformasi linier terhadap data asli. Data normalisasi didapat dari perhitungan nilai minimal dan maksimal dari setiap baris data di setiap kolom seperti berikut:

$$x_{norm} = \frac{x - \min f}{\max - \min f} \qquad (1)$$

Dimana:

x_norm	= data setelah normalisasi
x	= data yang akan dinormalisasi
Min *f*	= data terkecil dari kolom
Max *f*	= data terbesar dari kolom

Tahap normalisasi merubah data menjadi angka desimal dengan rentang 0 sampai 1 diterapkan kepada setiap data di setiap kolom kecuali kolom label. Pada penelitian ini, tahap normalisasi *MinMax* dilakukan agar data *input* seragam dan konsisten.

2.3.7 *SMOTE (Syntethic Minority Oversampling Technique)*

SMOTE melakukan *oversampling* untuk menyeimbangkan set pelatihan asli dengan menciptakan data sintetis, bukan sekadar menggandakan *instance* kelas minoritas. Hal ini dilakukan dengan menginterpolasi antara beberapa *instance* kelas minoritas dalam tetangga tertentu, berfokus pada "ruang fitur" dan bukan "ruang data" yaitu nilai-nilai fitur dan hubungannya. Metode SMOTE digunakan untuk menyeimbangkan jumlah label dengan menciptakan data sintetis dari kelas minoritas. Berikut proses penciptaan data sintetis dari *SMOTE* menurut (Fernández et al., 2018):

a) Pilih secara acak sebuah *instance* x_i dari kelas mintoritas dalam dataset, misalkan *instance* minoritas x_i memiliki fitur $(x_{i1}, x_{i2}, \dots, x_{in})$.
b) Temukan k tetangga terdekat dari *instance* x_i, nn dari x_i dalam ruang fitur, misal tertangga terdekat $x_{i,nn}$ dengan fitur $(x_{i,nn1}, x_{i,nn2}, \dots, x_{i,nnn})$.
c) Untuk setiap fitur *j* (dari 1 sampai *n*), interpolasi dilakukan pada persamaan 1.

$$x_{new,j} = x_{i,j} + \delta * (x_{i,nn,j-} x_{i,j}) \qquad (1)$$

Di mana δ adalah bilangan acak antara 0 sampai 1. Misal *instance* minoritas x_i = (1,2) dan terdekatnya x_i, nn = (2,3). Jika δ = 0.5, maka *instance* sintetis baru dihitung pada persamaan 2 dan 3

$$x_{new,1} = 1 + 0.5 * (2 - 1) = 1 + 0.5 * 1 = 1.5 \quad (2)$$

$$x_{new,2} = 2 + 0.5 * (3 - 2) = 2 + 0.5 * 1 = 2.5 \quad (3)$$

Sehingga *instance* sintetis baru x_{new} adalah (1.5, 2.5). Tabel 8 adalah perbandingan jumlah data pada tiap kelas sebelum dan sesudah dilakukan resampling dengan metode *SMOTE.*

Tabel 7. Hasil *resampling*

sebelum resampling		**sesudah resampling**	
kelas 1	kelas 0	kelas 1	kelas 0
980743	19257	784628	784628

2.4 CNN Deep Learning

Convolutional neural networks (CNN) adalah model *deep learning* yang meniru pola neuron dalam otak manusia dan memerlukan sedikit *preprocessing*. CNN terdiri dari beberapa lapisan, termasuk lapisan *convolution, pooling, fully connected, dan normalization.* Lapisan *convolution* memiliki hyperparameter seperti jumlah saluran input/output, ukuran padding, dan kernel. Lapisan *pooling* mengurangi dimensi data dengan menggabungkan output dari beberapa neuron menjadi satu neuron. Pada Lapisan *fully connected* setiap neuron menerima input dari semua neuron di lapisan sebelumnya (Akhtar and Feng, 2022).

Pada penelitian ini, proses klasifikasi data dilakukan dengan metode CNN namun tidak menggunakan *convolutional* layer karena masukkan datanya tidak berbentuk gambar, melainkan hanya menggunakan *input*, *hidden*, dan *output* layer saja, model ini biasa disebut *Feedforward Neural Network*. (Putra, 2020). Persamaan 3 dan 4 merupakan proses perhitungan *hidden* dan *output layer.*

$$o_j = \sigma\left(\sum_{k=1}^{K} x_k W_{kj} + \beta_j\right) \quad (3)$$

Di mana:
o_j: *Output* neuron ke-*j* pada *hidden layer.*
σ: Fungsi aktivasi
$\sum_{k=1}^{K} x_k W_{kj}$: Kombinasi linear dari input x_k dengan bobot W_{kj} yang menghubungkan input ke neuron-*j*
β_j*:* Bias atau *noise* yang ditambahkan ke neuron-*j*

Persamaan ini menghitung *output* dari neuron ke-*j* pada *hidden layer* dengan menerapkan fungsi aktivasi σ.

$$v_i = \sigma\left(\sum_{j=1}^{J} o_j u_{j,i} + \gamma_j\right) = \sigma\left(\sum_{j=1}^{J} \sigma\left(\sum_{k=1}^{J} x_k W_{kj} + \beta_j\right) u_{j,i} + \gamma_j\right) \quad (4)$$

Di mana:
v_i: *Output* neuron ke-*i* pada *layer* berikutnya (*output layer*).
o_j: *Output* dari neuron ke-*j* pada *hidden layer* sebelumnya, seperti yang dihitung pada persamaan 3.
$u_{j,i}$: Bobot yang menghubungkan neuron ke-*j* pada *hidden layer* ke neuron ke-*i* pada *output layer*.
γ_j: Bias atau *noise* yang ditambahkan ke neuron *i*.

Persamaan ini menghitung *output* dari neuron ke-*i* pada *layer* berikutnya dengan menerapkan fungsi aktivasi σ pada kombinasi linear dari bias dan *output hidden layer* sebelumnya. Persamaan ini juga menunjukkan bahwa *output layer* sebelumnya menjadi *input* untuk *layer* ini.

Tahap klasifikasi diawali dengan pembagian data menjadi data latih dan data uji. Proses pembagian data dilakukan dengan rasio 80:20 yaitu 80% untuk data latih dan sisanya sebagai data uji. *Deep learning* menggunakan beberapa *layer* atau lapisan. Setiap *layer* memiliki fungsi khusus dalam pelatihan data. Dalam penelitian ini, jenis *neural network* yang diterapkan adalah *sequential*, di mana pelatihan dilakukan secara berurutan, lapisan per lapisan, tanpa iterasi atau pengulangan. Jumlah total *layer* yang digunakan dalam penelitian ini adalah sebanyak 5, dengan rincian sebagai berikut:

Tabel 8. Arsitektur lapisan *neural network*

No	**Lapisan (tipe)**	**Output**	**Param**	**Fungsi Aktivasi**
1	*dense (input layer)*	(none, 20)	0	-
2	*dense_1 (hidden layer)*	(none, 500)	10500	*relu*
3	*dense_2 (hidden layer)*	(none, 200)	100200	*relu*
4	*dropout_1*	(none, 200)	0	-
5	*dense_3 (output layer)*	(none, 1)	201	*sigmoid*

2.5 Early Stopping

Early Stopping merupakan sebuah teknik yang digunakan saat melatih data *neural network* untuk mencegah *overfitting* dan menghemat waktu pelatihan (Fujo, Subramanian and Khder, 2022). *Early Stopping* dianggap sebagai teknik yang menguntungkan karena bisa mencegah model meneruskan pelatihan meskipun telah mencapai performa optimal pada data validasi. Saat proses pelatihan, performa model terhadap data validasi dievaluasi di setiap *epoch.* Cara kerja *Early Stopping* adalah membuat sebuah pemicu yang akan aktif jika sebuah kondisi tertentu terpenuhi. Kondisi atau

parameter tersebut antara lain; *patience*, *monitor*, *mode*, dan *min_delta*.

Tabel 9. Konfigurasi *early stopping*

patience	***monitor***	***mode***	***min_delta***
5	akurasi	max	0.001

a) *Patience* menunjukkan jumlah *epoch* yang diizinkan untuk berjalan tanpa ada peningkatan performa sebelum pelatihan dihentikan. Dalam kasus ini, nilai *patience* adalah 5, yang berarti jika tidak ada peningkatan yang signifikan dalam metrik yang dimonitor (dalam hal ini akurasi) selama 5 *epoch* berturut-turut, pelatihan akan dihentikan.
b) *Monitor* adalah metrik yang dipantau selama pelatihan untuk menentukan apakah model mengalami peningkatan performa atau tidak. Pada tabel ini, *monitor* diatur ke akurasi, yang berarti bahwa akurasi model pada data validasi akan dievaluasi di setiap *epoch*.
c) *Mode*: Menentukan apakah metrik yang dipantau harus dimaksimalkan atau diminimalkan. Dalam kasus ini, *mode* diatur ke max, yang berarti bahwa kita ingin memaksimalkan akurasi. Jika mode diatur ke min, berarti kita ingin meminimalkan metrik yang dipantau (misalnya, loss).
d) *min_delta* adalah nilai perubahan minimum yang dianggap sebagai peningkatan yang signifikan dalam metrik yang dimonitor. Dalam tabel ini, *min_delta* diatur ke 0.001, yang berarti bahwa peningkatan akurasi harus setidaknya 0.001 untuk dianggap sebagai peningkatan yang signifikan. Jika perubahan akurasi kurang dari 0.001, maka peningkatan tersebut tidak akan dianggap signifikan dan tidak akan mereset *counter patience*.

3. HASIL DAN ANALISIS

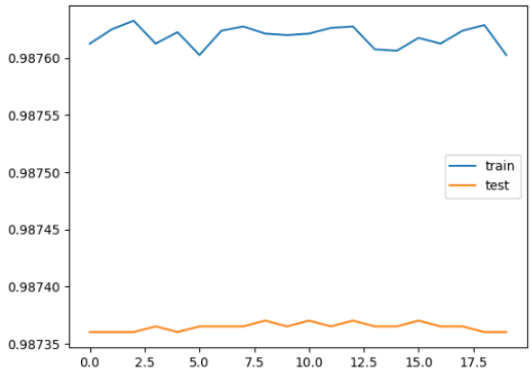
3.1 Pelatihan Pada Dataset Tidak Seimbang

Proses pelatihan awal dilakukan pada dataset yang tidak seimbang selama 20 *epoch*. Fungsi *callback* atau *early stopping* diaktifkan pada *epoch* ke-16 ketika akurasi pada data uji dan data validasi mencapai 98%, dan proses ini membutuhkan waktu 236 detik. Kejadian ini disebabkan oleh pengaturan *patience* sebanyak 10, yang berarti parameter yang dimonitor (*val_accuracy*) tidak mengalami peningkatan sejak *epoch* ke-5. Teknik *early stopping* menghitung 10 *epoch* berturut-turut mulai dari *epoch* ke-6 hingga mencapai epoch ke-16.

Perbedaan antara akurasi pelatihan dan validasi sangat kecil. Ini adalah tanda bahwa model tidak mengalami *overfitting*, karena performa pada data validasi hampir sama dengan performa pada data pelatihan (Fernández et al., 2018).

Tabel 11. Akurasi di setiap *epoch*

epoch ke-	*accuracy*	*val_accuracy*
1	0.9841	0.98734
2	0.9841	0.98738
3	0.9876	0.98738
4	0.9876	0.98738
5	0.9876	0.98739
6	0.9876	0.98739
7	0.9876	0.98739
8	0.9876	0.98739
9	0.9876	0.98739
10	0.9876	0.98739
11	0.9876	0.98739
12	0.9876	0.98739
13	0.9876	0.98739
14	0.9876	0.98739
15	0.9876	0.98739
16	0.9876	0.98739



Gambar 2. Trend akurasi per *epoch* data tidak seimbang

Kedua garis, baik untuk set data pelatihan maupun pengujian, berada pada kisaran yang sangat tinggi (sekitar 0.9874 sampai 0.9876). Ini menunjukkan bahwa model memiliki performa yang sangat baik pada kedua set data. Garis akurasi untuk set data pelatihan dan pengujian terlihat stabil dan tidak menunjukkan fluktuasi yang signifikan. Ini adalah tanda yang baik bahwa model tidak mengalami *overfitting* atau *underfitting* secara drastis. Perbedaan antara akurasi pelatihan dan pengujian sangat kecil. Ini mengindikasikan bahwa model tidak *overfitting*, karena *overfitting* biasanya ditandai dengan akurasi pelatihan yang sangat tinggi tetapi akurasi pengujian yang jauh lebih rendah. Namun, karena ada ketidakseimbangan jumlah kelas, *recall* untuk kelas 0 adalah 0.36 (Tabel 3), yang berarti model hanya berhasil mengidentifikasi 36% dari semua sampel kelas 0. Hal ini menunjukkan bahwa model cenderung bias terhadap kelas mayoritas, yang menyebabkan performa yang buruk pada kelas minoritas.

3.2 Pelatihan Pada Dataset Seimbang

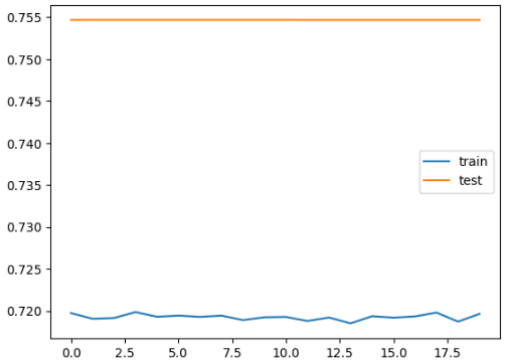
Proses pelatihan berikutnya diterapkan pada dataset yang telah diresampling menggunakan teknik *SMOTE*. Langkah-langkah pelatihan ini mirip dengan pelatihan pada dataset yang tidak seimbang, yaitu dilakukan selama 20 *epoch* dan dihentikan pada epoch ke-20. Meskipun demikian, pelatihan pada dataset ini memerlukan waktu lebih lama, mencapai

345 detik, karena jumlah *instance* dari kedua kelas menjadi seimbang. Hasil pelatihan pada dataset ini menunjukkan akurasi sebesar 72% pada data uji dan 75% pada data validasi.

Tabel 12. Akurasi di setiap *epoch*

epoch ke-	*accuracy*	*val_accuracy*
1	0.7197	0.75465
2	0.7143	0.75465
3	0.7191	0.75465
4	0.7199	0.75465
5	0.7193	0.75465
6	0.7194	0.75465
7	0.7193	0.75465
8	0.7195	0.75465
9	0.7189	0.75465
10	0.7192	0.75465
11	0.7193	0.75465
12	0.7188	0.75465
13	0.7192	0.75465
14	0.7185	0.75465
15	0.7194	0.75465
16	0.7192	0.75465
17	0.7193	0.75465
18	0.7198	0.75465
19	0.7187	0.75465

Pada dataset yang seimbang, akurasi model menurun menjadi sekitar 0.75 yang menunjukkan bahwa model tidak lagi hanya mengandalkan prediksi kelas mayoritas dan mencoba untuk memprediksi kedua kelas dengan lebih seimbang. Pada dataset yang seimbang, *recall* untuk kelas 0 menjadi 1.00 (Tabel 4) dan untuk kelas 1 menurun (0.51). Hal ini menunjukkan bahwa model sekarang mampu mengidentifikasi hampir semua sampel kelas 0, meskipun dengan *trade-off recall* yang lebih rendah untuk kelas 1.

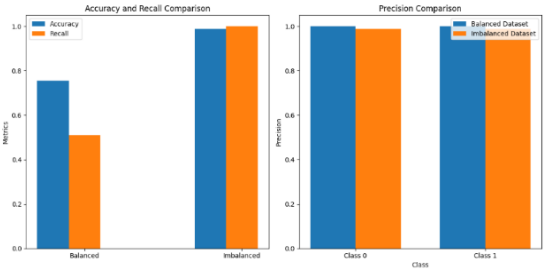


Gambar 3. Trend akurasi per *epoch* data seimbang

Akurasi yang stabil di kedua set pelatihan dan validasi menunjukkan bahwa model tidak mengalami *overfitting* atau *underfitting*. Dengan menggunakan SMOTE, hasil pelatihan menunjukkan bahwa model sekarang memperlakukan kedua kelas dengan lebih seimbang. Namun, akurasi yang lebih rendah dari sebelumnya menunjukkan bahwa model sedang berusaha untuk menyeimbangkan kinerja antar kelas.

3.3 Evaluasi Kinerja

Perbedaan utama antara dua set data terletak pada distribusi kelas dan dampak metrik evaluasi. Pada set data yang tidak seimbang, presisi tinggi dan recall yang rendah untuk kelas 0 mencerminkan kehati-hatian model dalam memprediksi kelas 0, menghasilkan presisi yang tinggi namun menemukan sebagian kecil dari instance kelas 0, sehingga recall rendah. Sebaliknya, pada set data yang seimbang, presisi kelas 0 lebih rendah tetapi recall tinggi, menandakan kemampuan model untuk menemukan sebagian besar instance kelas 0. Dalam situasi ketidakseimbangan data, distribusi kelas sangat tidak seimbang, dengan mayoritas instance (kelas 1) mendominasi dan minoritas instance (kelas 0) jumlahnya sedikit.



Gambar 4. Perbandingan performa

Hal ini mencerminkan kondisi di mana identifikasi kelas minoritas merupakan hasil yang sangat diinginkan dan sulit ditemukan. Sebaliknya, set data seimbang memiliki jumlah *instance* yang setara untuk kedua kelas, lebih sesuai untuk kasus di mana kedua kelas memiliki tingkat penting yang setara.

Metrik presisi dan *recall* berfungsi untuk mengevaluasi kompromi antara mengurangi *false positive* (presisi tinggi) dan menemukan sebanyak mungkin *instance* positif yang benar (*recall* tinggi). Pada set data yang tidak seimbang, presisi tinggi untuk kelas mayoritas (1) mencerminkan kehati-hatian model dalam memprediksi kelas tersebut dan menghindari *false positive*. Namun, dampaknya adalah *recall* yang rendah, menandakan bahwa sebagian besar *instance* kelas minoritas (0) terlewatkan. Pada data seimbang, *trade-off* ini dapat bervariasi, dengan *recall* yang lebih tinggi untuk kelas 0, menandakan kemampuan model untuk menemukan lebih banyak *instance* kelas 0 dengan sedikit pengorbanan pada presisi. Di sisi lain, pada data yang tidak seimbang, akurasi mungkin sangat tinggi karena mayoritas prediksi adalah kelas mayoritas yang benar. Namun, hal ini dapat menimbulkan kesan kinerja yang baik padahal sebenarnya tidak akurat. Pada data seimbang, akurasi mungkin lebih rendah, tetapi memberikan evaluasi yang lebih adil terkait seberapa baik model mengidentifikasi kedua kelas. Evaluasi kinerja model dilakukan dengan cara melihat laporan klasifikasi dari *library sklearn classification_report*. Metrik-metrik yang akan dievaluasi dari laporan klasifikasi ini adalah akurasi, presisi, *recall* dan *f1-score*.

Tabel 13. Metrik pada data tidak seimbang

	presisi	***recall***	**f1-*score***	support
0	1.00	0.35	0.52	3885
1	0.99	1.00	0.99	196115
akurasi			0.99	200.000
macro avg	0.99	0.67	0.76	200.000
weighted avg	0.99	0.99	0.98	200.000

Tabel 14. Metrik pada data seimbang

	presisi	***recall***	**f1-*score***	support
0	0.67	1.00	0.80	196115
1	1.00	0.51	0.68	196115
akurasi			0.75	392230
macro avg	0.84	0.75	0.74	392230
weighted avg	0.84	0.75	0.74	392230

Tabel 15. Perbandingan akurasi

metode	akurasi
CNN	0.75 (penulis)
CNN	0.6935 ((Liang and Vankayalapati, 2021)

4. Kesimpulan

Kinerja yang didapatkan terbukti lebih tinggi dibandingkan dengan penelitian (Liang and Vankayalapati, 2021). Dengan Presisi dan *Recall* yang didapat pada dataset tidak seimbang di kelas 0 adalah 100% dan 35%, sedangkan pada kelas 1 adalah 99% dan 100%. Pada dataset seimbang, presisi dan *recall* di kelas 0 adalah 67% dan 100%, sedangkan pada kelas 1 adalah 100% dan 51%. Lalu akurasi yang didapat pada dataset tidak seimbang adalah sebesar 99% sedangkan pada dataset seimbang sebesar 75%.

Dalam konteks dataset tidak seimbang, distribusi kelas sangat tidak seimbang, dengan sebagian besar *instance* milik kelas mayoritas (1) dan hanya sejumlah kecil milik kelas minoritas (0). Hal ini mencerminkan situasi di mana kelas minoritas memiliki nilai yang sangat diinginkan dan sulit untuk diidentifikasi. Sebaliknya, dataset seimbang memiliki jumlah instance yang sama untuk kedua kelas, yang mungkin lebih sesuai untuk kasus di mana kedua kelas memiliki tingkat penting yang setara. Ketidakseimbangan kelas merupakan permasalahan umum dalam tugas klasifikasi.

DAFTAR PUSTAKA

AKHTAR, M.S. AND FENG, T., 2022. Malware Analysis and Detection Using Machine Learning Algorithms. *Symmetry*, 14(11). https://doi.org/10.3390/sym14112304.

AL-RIMY, B.A.S., SAEED, F., AL-SAREM, M., ALBARRAK, A.M. AND QASEM, S.N., 2023. An Adaptive Early Stopping Technique for DenseNet169-Based Knee Osteoarthritis Detection Model. *Diagnostics*, 13(11). https://doi.org/10.3390/diagnostics13111903.

AZZAHRA NASUTION, D., KHOTIMAH, H.H. AND CHAMIDAH, N., 2019. *Perbandingan Normalisasi Data Untuk Klasifikasi Wine Menggunakan Algoritma K-NN*.

BAKAR, M. Y. A. AND ADIWIJAYA, A. (2021) 'Klasifikasi Teks Hadis Bukhari Terjemahan Indonesia Menggunakan Recurrent Convolutional Neural Network (CRNN)', *Jurnal Teknologi Informasi dan Ilmu Komputer*, 8(5), pp. 907–918.

CAHYANTO, T. A. *ET AL.* (2022) 'Intelligent ubiquitous technology as a precision agri-food framework: A proposed framework', *IOP Conference Series: Earth and Environmental Science*, 1041(1). doi: 10.1088/1755-1315/1041/1/012022.

FERNÁNDEZ, A., GARCÍA, S., HERRERA, F. AND CHAWLA, N. V, 2018. *SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary*. *Journal of Artificial Intelligence Research*, .

FUJO, S.W., SUBRAMANIAN, S. AND KHDER, M.A., 2022. Customer churn prediction in telecommunication industry using deep learning. *Information Sciences Letters*, 11(1), pp.185–198. https://doi.org/10.18576/isl/110120.

GARCIA, S., PARMISANO, A. AND ERQUIAGA, M. J. (2020) *IoT-23: A labeled dataset with malicious and benign IoT network traffic*. doi: http://doi.org/10.5281/zenodo.4743746.

JAIN, A., RATNOO, S. AND KUMAR, D. (2020) 'A novel multi-objective genetic algorithm approach to address class imbalance for disease diagnosis', *International Journal of Information Technology*, pp. 1–16.

LECUN, Y., BENGIO, Y. AND HINTON, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436–444. doi: 10.1038/nature14539.

LIANG, Y. AND VANKAYALAPATI, N., 2021. *Machine Learning and Deep Learning Methods for Better Anomaly Detection in IoT-23 Dataset Cybersecurity*.

LIBOVICKÝ, J., 2017. *Deep Learning for Natural Language processing Introduction to Natural Language Processing*.

PUTRA, J.W.G., 2020. Pengenalan Pembelajaran Mesin dan Deep Learning. pp.150–151.

RIZKI, M., BASUKI, S. AND AZHAR, Y., 2020. Implementasi Deep Learning Menggunakan Arsitektur Long Short Term Memory Untuk Prediksi Curah Hujan Kota Malang. *REPOSITOR*, 2(3), pp.331–338.

SABAH, S. *ET AL.* (2019) 'Big data with decision tree induction', in *2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 1–6.

SORRI, K., MUSTAFEE, N. AND SEPPÄNEN, M., 2022. Revisiting IoT definitions: A framework towards comprehensive use. *Technological Forecasting and Social Change*, 179. https://doi.org/10.1016/j.techfore.2022.121623.

SUPRAYOGI, C. AND MARWAN, M.A., 2022. Classification of Network Traffic Data Mirai Malware Attacks on Internet of Things Devices Using the K-Nearest Neighbor Method. *International Research Journal of Advanced Engineering and Science*, 7(4), pp.39–43.

VAN DARTEL, B., 2021. *Malware detection in IoT devices using Machine Learning*.

YUAN, T. *ET AL.* (2020) 'High performance CNN accelerators based on hardware and algorithm co-optimization', *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(1), pp. 250–263.

ZHANG, H. *ET AL.* (2019) 'Classification of ransomware families with machine learning based on N-gram of opcodes', *Future Generation Computer Systems*, 90, pp. 211–221. doi: 10.1016/j.future.2018.07.052.

ZOUMPEKAS, T., SALAMÓ, M. AND PUIG, A., 2022. Effective Early Stopping of Point Cloud Neural Networks. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Science and Business Media Deutschland GmbH. pp.156–167. https://doi.org/10.1007/978-3-031-13448-7_13.

Halaman ini sengaja dikosongkan