*Jurnal Sistem Komputer dan Informatika (JSON)*
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


# Deteksi Malware Berbasiskan Analisis Statis Menggunakan Algoritma Convolutional Neural Network (CNN)

**Tri Wiyono*, Edrian Hadinata, Ahmad Zakir, Andi Marwan Elhanafi**
Fakultas Teknik dan Komputer, Sistem Informasi, Universitas Harapan Medan, Medan, Indonesia
Email: [1,*]tri82.wiyono@gmail.com, [2]edrianhadinata@gmail.com,[3]suratzakir@gmail.com, [4]andimarwanelhanafi@gmail.com
Email Penulis Korespondensi: tri82.wiyono@gmail.com*
Submitted: **28/07/2025**; Accepted: **25/08/9999**; Published: **08/09/2025**

**Abstrak–**Sistem informasi berperan penting dalam menjaga kelancaran operasional organisasi sekaligus melindungi data dari ancaman siber, termasuk malware. Pertumbuhan akses internet di Indonesia yang mencapai lebih dari 215 juta pengguna pada 2023 dan penetrasi smartphone Android sebesar 132,7 juta pada 2025 meningkatkan potensi risiko keamanan. Rendahnya kesadaran pengguna Android terhadap keamanan informasi (rata-rata 37%) menjadi faktor dominan munculnya insiden spam, phishing, dan malware. Dalam konteks ini, analisis statis menjadi salah satu pendekatan efektif untuk mendeteksi malware melalui izin aplikasi (permissions) yang tertera pada file AndroidManifest.xml. Namun, kelemahannya terletak pada keterbatasan mendeteksi kode berteknik obfuscation. Penelitian ini mengembangkan model deteksi malware berbasis Convolutional Neural Network (CNN) dengan memanfaatkan fitur statis APK. Dataset yang digunakan mencakup aplikasi berbahaya seperti Undangan Pernikahan.apk serta data publik dari Kaggle. Proses meliputi ekstraksi izin aplikasi, normalisasi, pembentukan vektor 256 fitur, serta klasifikasi biner malware dan benignware. Hasil uji menunjukkan akurasi 92% dengan precision tinggi (0,93) namun recall pada malware relatif rendah (0,77), mengindikasikan masih adanya false negative signifikan. Temuan ini menegaskan bahwa CNN efektif untuk deteksi berbasis izin, tetapi peningkatan recall diperlukan agar sistem lebih andal. Pengembangan pendekatan hibrida dengan menggabungkan analisis statis dan dinamis disarankan untuk memperkuat deteksi malware Android yang semakin kompleks. Dengan demikian, penelitian ini diharapkan memberikan kontribusi signifikan terhadap pengembangan sistem keamanan siber yang adaptif, andal, serta relevan bagi tantangan digital masa depan.

**Kata Kunci:** Android; Malware; CNN; Phishing; Permissions

**Abstract–**Information systems play a crucial role in ensuring organizational operations while protecting data from cyber threats, including malware. The rapid growth of internet users in Indonesia, reaching over 215 million in 2023, and the penetration of 132.7 million Android smartphones in 2025, have heightened security risks. Low awareness of information security among Android users (average 37%) is a dominant factor in the prevalence of spam, phishing, and malware incidents. In this context, static analysis provides an efficient approach for detecting malware through application permissions listed in the AndroidManifest.xml file, although its limitation lies in detecting obfuscated code. This research develops a malware detection model based on Convolutional Neural Network (CNN) using static APK features. The dataset includes malicious samples such as Undangan Pernikahan.apk along with public datasets from Kaggle. The process involves extracting application permissions, normalization, forming 256-feature vectors, and conducting binary classification between malware and benignware. Experimental results show an accuracy of 92%, with high precision (0.93) but relatively low recall for malware (0.77), indicating the presence of significant false negatives. The findings confirm that CNN is effective for permission-based detection, but recall improvement is necessary to enhance system reliability. A hybrid approach combining static and dynamic analysis is recommended to strengthen detection against increasingly complex Android malware. Thus, this study is expected to contribute significantly to the development of adaptive and reliable cybersecurity systems, providing relevance for addressing future digital challenges.

**Keywords**: Android; Malware; CNN; Phishing; Permissions

## 1. PENDAHULUAN

Sistem informasi adalah fondasi utama untuk operasional dan pengambilan keputusan di berbagai sektor, mengintegrasikan teknologi informasi untuk menjaga kelancaran dan perlindungan data organisasi. Salah satu elemen penting dalam keamanan ini adalah deteksi malware, yang berfungsi melindungi integritas, ketersediaan, dan kerahasiaan [1] data dari berbagai ancaman cyber seperti malware, ransomware, dan trojan [2]. Menurut data Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), pada tahun 2023, jumlah pengguna internet di Indonesia mencapai 215,63 juta jiwa atau sekitar 78% [3] dari total populasi [4]. Pertumbuhan akses internet ini mendorong besarnya penggunaan layanan daring dalam sektor publik, pendidikan, keuangan, komersial, dan sosial. Namun, perluasan infrastruktur dan penggunaan internet yang masif juga membuka celah bagi berbagai ancaman siber, seperti malvertising**,** phishing dan serangan digital lainnya [5]. Ancaman tersebut akan terjadi pada jutaan pengguna platform Android yang terdapat di Indonesia. Data menunjukkan bahwa penetrasi smartphone Android di Indonesia mencapai 132,7 juta perangkat pada awal 2025 [6]. Penelitian di Indonesia menyebutkan bahwa pengguna Android memiliki tingkat kesadaran terhadap keamanan informasi yang masih rendah (rata-rata 37%) menjadi faktor dominan hadirnya insiden seperti spam, phishing, spoofing, dan malware [7]. Terlebih pada

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


serangan malware yang kerap terjadi dimasyarakat penting untuk memperhatikan dampak signifikan yang di timbulkan.

Serangan malware memiliki potensi menimbulkan dampak yang signifikan terhadap berbagai entitas, baik pada tingkat individu maupun institusional. Risiko yang ditimbulkan dapat mencakup kerusakan sistem, kehilangan integritas data, hingga kebocoran informasi sensitif seperti serangan mobile banking yang menggunakan metode phishing dan mengincar data sensitif seperti kode OTP, PIN, serta Password akun sosial media hingga akun bank [8]. Oleh sebab itu proses analisis penting dilakukan agar dapat menghindari berbagai ancaman yang akan terjadi.

Dalam konteks upaya deteksi dan pencegahan, pendekatan analisis statis dikenal memiliki keunggulan dari segi kecepatan dan efisiensi proses [9], karena tidak memerlukan eksekusi kode secara langsung namun analisis ini memiliki kelemahan dimana hasil akan efektif jika malware tidak menggunakan teknik penyamaran kode sumber atau obfuscation. Keunggulannya yang lain adalah analisis statis melakukan proses analisis pada meta data dan dapat mengambil informasi awal seperti permission yang dapat digunakan untuk analisis selanjutnya.

Sebuah penelitian di Institut Bisnis dan Teknologi Indonesia (2023) melakukan reverse engineering terhadap sample malware yang menyamar sebagai aplikasi pesan instan. Hasil analisis statis menunjukkan bahwa malware tersebut meminta izin sensitif seperti akses SMS, lokasi dan kontak. Setelah mendapatkan izin, malware secara diam-diam mengirim data pribadi melalui API Telegram menggunakan token dan chat_id tertentu [10]. Temuan ini menyoroti bahwa aplikasi berbahaya tidak hanya menargetkan platform umum, tetapi juga memanfaatkan jalur pengiriman data yang canggih.

Jika ditinjau dari sisi platform, sifat Android yang merupakan open-source, ekosistem aplikasi pihak ketiga yang besar (termasuk situs dan toko aplikasi di luar Google Play), serta fleksibilitas pemakaian, menjadikannya target menarik bagi aktor kejahatan siber [11]. Android Package Kit (APK) tidak resmi atau diluar Playstore dan rekayasa APK ini sering dipakai untuk menyebarkan malware yang dapat mengambil konten SMS, mengirim SMS dan sebagainya pada perangkat pengguna [12], contohnya adalah aplikasi Undangan Pernikahan.APK. Dimana APK ini mengakses modul inbox yang ada pada smartphone pengguna tentunya sudah dipastikan dapat mengambil setiap SMS masuk ke smarphone pengguna. Tentunya ini sangat berbahaya dikarena semua SMS yang masuk tanpa terkecuali akan di terima oleh aplikasi malware seperti SMS OTP, Password, CVV, Kredit Card dan data kredensial lainya. Oleh sebab itu, metode analisis statis diperlukan untuk mengambil fitur android yang terdapat pada file AndroidManifest.xml dan menemukan data user permissions sebagai langkah awal untuk proses analisis lanjutan.

Untuk melengkapi hasil analisis statis, proses analisis lanjutan menggunakan teknik modern dapat meningkatkan akurasi pada pendeteksian malware. Seperti penggunaan metode machine learning dalam memprediksi malware dan menyebutkan bahwa penggunaan teknik machine learning cukup efektif dalam mendeteksi malware yang terdapat pada perangkat Android [12]. Seperti pada penelitian [13]. menggunakan model deep learning untuk mendeteksi sebuah malware yang di sembunyikan pada file PDF [13]. [14] menggunakan analisis statis, dynamic dan hybrid untuk mendeteksi malware [14] dan menggunakan Support Vector Machine (SVM) dalam proses analisisnya. Sementara penelitian lain menggunakan CNN dalam proses deteksi malware [15]. Dengan demikian penting memanfaatkan teknik dalam machine learning dalam proses pendeteksian dan penggunaan CNN dinyatakan cukup efektif.

Pada tahun 2021[16] mengembangkan HomDroid, metode deteksi malware “convert” yang memanfaatkan analisis call-graph untuk mengidentifikasi bagian jahat terselubung dalam APK. Metode ini mampu mendeteksi 96,8% malware tersembunyi, jauh lebih baik dibanding sistem lain seperti Drebin dan MaMaDroid. Pendekatan ini memperlihatkan pentingnya analisis lain berbasis machine learning dibandingkan fitur statis saja.

Seperti yang telah disebutkan bahwa metode berbasis deep learning, khususnya Convolutional Neural Network (CNN) [17], mulai diterapkan dalam bidang keamanan siber. CNN memiliki kemampuan tinggi dalam mengenali pola dan struktur data, termasuk dalam menganalisis fitur statis seperti permission serta perilaku dinamis seperti akses jaringan atau proses berbahaya. Namun, pemanfaatan fitur yang diperoleh dari analisis statis masih belum dimanfaatkan untuk mendeteksi file APK dalam kasus Undangan Pernikahan.apk yang di kombinasikan dengan CNN. Padahal CNN memiliki keunggulan tertentu yang mampu mendeteksi malware pada file APK.

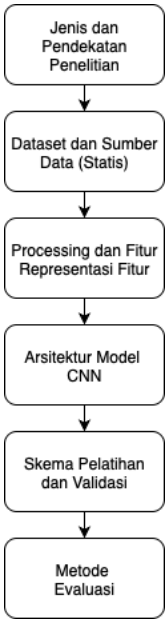
Untuk mengatasi kelemahan tersebut, penelitian ini mengusulkan pendekatan baru dengan mengombinasikan analisis statis dan Convolutional Neural Network (CNN). Analisis statis digunakan pada tahap awal untuk mengekstraksi fitur izin (permissions) dari file APK, sementara CNN dimanfaatkan untuk membangun model klasifikasi berbasis pembelajaran mendalam yang mampu mengenali pola kompleks dari data tersebut. Dengan memanfaatkan kemampuan CNN dalam mendeteksi struktur dan pola data, diharapkan metode ini mampu meningkatkan akurasi deteksi malware Android, termasuk pada kasus aplikasi berbahaya seperti Undangan Pernikahan.apk, yang memanfaatkan izin SMS, RECEIVE SMS, SEND SMS untuk mencuri data sensitif. Dengan demikian, perpaduan analisis statis dan CNN diharapkan dapat menjadi solusi efektif untuk menghadapi ancaman malware yang semakin canggih pada ekosistem Android.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


# 2. METODELOGI PENELITIAN

## 2.1 Jenis dan Pendekatan Penelitian

Penelitian ini merupakan penelitian terapan (applied research) dengan pendekatan kuantitatif dan bersifat eksperimental. Tujuan utama penelitian adalah membangun serta mengevaluasi model deteksi malware berbasis Android menggunakan algoritma Convolutional Neural Network (CNN) dengan memanfaatkan data fitur status hasil ekstraksi dari file APK.




**Gambar 1.** Metodologi Penelitian

## 2.2 Dataset dan Sumber Data

Dataset yang digunakan dalam penelitian ini terdiri dari atas sumber utama. Pertama, file APK Undangan Pernikahan.apk yang dikategorikan sebagai sample malware berbasis Android dan dijadikan sebagai studi kasus. Kedua, dataset publik yang diperoleh dari platform Kaggle, yang berisi kumpulan fitur-fitur aplikasi Android hasil ekstraksi statis. Dataset tersebut telah terklasifikasi menjadi dua kategori, yaitu malware (label 0) dan benignware (label 1)[18]. Penggabungan kedua sumber data ini dimaksudkan untuk memperkaya variasi sample serta meningkatkan validitas proses pelatihan dan pengujian model deteksi malware yang dikembangkan.

Dalam penelitian ini digunakan analisis statis**,** yaitu metode deteksi yang dilakukan tanpa mengeksekusi aplikasi. Analisis statis memanfaatkan struktur internal aplikasi Android, terutama berkas AndroidManifest.xml, untuk mengekstraksi fitur-fitur seperti izin (permissions) yang diminta oleh aplikasi. Informasi tersebut kemudian direpresentasikan dalam bentuk vektor fitur yang menjadi input bagi model Convolutional Neural Network (CNN). Pendekatan ini dipilih karena relatif efisien, cepat, dan mampu memberikan gambaran awal mengenai potensi perilaku berbahaya dari sebuah aplikasi Android [19].

## 2.4 Processing dan Representasi Fitur

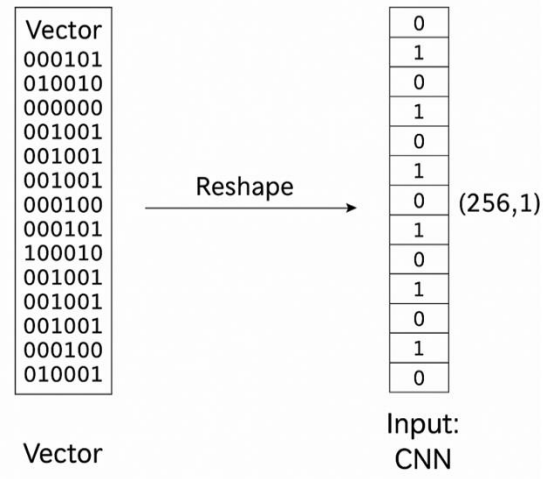
Tahap processing data merupakan langkah krusial dalam membangun model deteksi malware berbasis CNN proses ini bertujuan untuk menyiapkan data agar sesuai dengan kebutuhan model dan meningkatkan akurasi prediksi. Seperti penggabungan dataset yang diperoleh fitur statis dan dinamis, melakukan normalisasi data, konversi ke format input CNN, pembagian dataset. Data yang digunakan pada penelitian ini data yang berbentuk vektor 256 elemen yaitu 0; 0; 0; 0; 0; 0; 1; 0; ... ; 0; 1; 0; 0; 0; ... ; 0, ukuran 256 fitur penggabungan antara analisis statis dan dinamis. Adapun bentuk input ke CNN yang digunakan dalam model klasifikasi biner (malware dan benignware). Adapun contoh vektor fitur 256 elemen

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 1, 0,

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
.....
0, 1, 0, 0, 1, 0, 0, 0, 0, 0] ← total: 256 elemen

Agar bisa diproses oleh Conv1D, vector ini di-reshape menjadi (256,1) contoh visualisasi sederhana [[0],[0],[0],[0],[1],[0],...[1]]. Bentuk input batch (samples, 256, 1). Misalnya jika ada 1000 data X.shape = (1000, 256, 1).




**Gambar 2.** Bentuk Vector & Reshape

### 2.5 Arsitektur Model

Berikut adalah penjelasan rinci dari arsitektur model CNN satu dimensi (1D) yang digunakan untuk klasifikasi biner (malware dan benignware), serta contoh perhitungan manual dalam konteks CNN.

**Tabel 1.** Arsitektur CNN

| Layer | Konfigurasi | Fungsi dan Penjelasan |
|---|---|---|
| Input | (256,1) | Input berupa vector satu dimensi dengan panjang 256 dan 1 saluran (channel). Vektor ini mewakili 256 fitur dari data permission hasil analisis statis dan dinamis. |
| Conv1D | 32 Filter, kernel = 3, stride = 1, padding = valid | Melakukan konvolusi sepanjang sumbu fitur. Tiap filter menghasilkan fitur baru berdasarkan pola lokal sepanjang 3 elemen input. Output shape: (256 − 3 + 1) = (254, 32). |
| ReLU | Aktivasi nonliner | Fungsi aktivasi non-linear ReLU (Rectified Linear Unit), mendefinisikan output sebagai f(x) = max(0, x) untuk memperkenalkan non-linearitas pada jaringan. |
| MaxPooling1D | Pool size = 2 | Melakukan downsampling dengan memilih nilai maksimum dari setiap dua elemen bertetangga. Output shape: (254 / 2) = 127 (dibulatkan jika perlu). |
| Flatten | Mengubah ke vector 1D | Mengubah representasi tensor 3D (batch, timestep, channel) menjadi vector 1D agar dapat diproses oleh lapisan Dense (fully connected). |
| Dense | 64 Neuron, aktivasi ReLU | Lapisan penuh dengan 64 unit. Menerima input hasil flatten dan mengaplikasikan aktivasi ReLU untuk menghasilkan representasi tinggi dari fitur. |
| Dropout | 50% (training) | Mengabaikan (drop) secara acak 50% neuron saat pelatihan untuk mencegah overfitting. |
| Ouput | 1 neuron, aktivasi Sigmoid | Menghasilkan nilai antara 0 dan 1 yang mengindikasikan probabilitas kelas. Sigmoid: σ(x) = 1 / (1 + e^(-x)). Digunakan untuk klasifikasi biner. |

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


Contoh perhitungan manual Conv1D misalkan input [1, 2, 3, 4, 5], filter [0.2, 0.5, -0.3] stride = 1, padding = valid. Adapun langkah konvolusi manual adalah sebagai berikut :

a. Posisi 1 : (1 × 0.2) + (2 × 0.5) + (3 × -0.3) = 0.2 + 1.0 - 0.9 = 0.3
b. Posisi 2 : (2 × 0.2) + (3 × 0.5) + (4 × -0.3) = 0.4 + 1.5 - 1.2 = 0.7
c. Posisi 3 : (3 × 0.2) + (4 × 0.5) + (5 × -0.3) = 0.6 + 2.0 - 1.5 = 1.1

Output dari Conv1D [0.3, 0.7, 1.1] kemudian akan diproses oleh ReLU max(0,x) dan output ReLU [0.3, 0.7, 1.1].

### 2.7. Skema Pelatihan dan Validasi

Model CNN yang dikembangkan dalam penelitian ini dilatih menggunakan dataset yang telah melalui tahap praproses. Proses pelatihan dilakukan dengan menerapkan konfigurasi hyperparamete yang umum digunakan dalam klasifikasi biner. Optimasi parameter jaringan dilakukan menggunakan algoritma Adam Optimizer, sedangkan fungsi kerugian yang digunakan adalah binary cross-entropy karena sesuai untuk permasalahan klasifikasi dua kelas. Ukuran batch (batch size) ditentukan berdasarkan kapasitas komputasi yang tersedia, umumnya berkisar antara 32 hingga 64 sampel per iterasi. Jumlah epoch ditetapkan secara iteratif hingga model mencapai kondisi convergence atau tidak terjadi peningkatan signifikan pada performa validasi. Untuk menjaga generalisasi model, dataset dibagi menjadi tiga subset, yaitu training set untuk proses pembelajaran parameter model, validation set untuk memantau performa selama pelatihan dan mencegah overfitting, serta testing set yang digunakan secara eksklusif untuk mengevaluasi kinerja akhir model. Pembagian ini dimaksudkan agar hasil evaluasi dapat mencerminkan kemampuan model dalam mendeteksi malware pada data yang belum pernah dilihat sebelumnya.

### 2.8 Metode Evaluasi

Evaluasi dilakukan untuk menilai kinerja model CNN dalam mengklasifikasikan aplikasi Android ke dalam dua kategori, yaitu malware dan benignware. Proses evaluasi menggunakan sejumlah metrik performa yang umum diterapkan pada tugas klasifikasi biner.

1. Akurasi (Accuracy)**,** digunakan untuk mengukur tingkat ketepatan prediksi model secara keseluruhan dengan membandingkan jumlah prediksi benar terhadap total prediksi.
2. Presisi **(**Precision**)**, menggambarkan tingkat ketepatan model dalam mengidentifikasi malware, yakni proporsi prediksi positif yang benar dari seluruh prediksi positif.
3. Recall **(**Sensitivity**)**, menunjukkan kemampuan model dalam mendeteksi seluruh sampel malware yang ada pada dataset.
4. F1-Score, berfungsi sebagai ukuran harmonisasi antara presisi dan recall, sehingga dapat memberikan penilaian yang lebih seimbang ketika distribusi kelas tidak merata.
5. Confusion Matrix, memberikan informasi detail mengenai distribusi hasil klasifikasi, baik prediksi yang benar maupun kesalahan prediksi pada masing-masing kelas.

Selain itu, penggunaan mekanisme dropout serta pemanfaatan validation set selama pelatihan dimaksudkan untuk mengurangi risiko overfitting dan memastikan bahwa performa model tidak hanya tinggi pada data latih, tetapi juga mampu melakukan generalisasi pada data baru. Evaluasi berbasis metrik-metrik tersebut diharapkan dapat memberikan gambaran yang komprehensif mengenai efektivitas model dalam mendeteksi malware pada platform Android.

## 3. HASIL DAN PEMBAHASAN

### 3.1 Persiapan Dataset dan Processing

Penelitian ini mengimplementasikan algoritma CNN satu dimensi untuk mendeteksi malware pada aplikasi Android (APK) berdasarkan izin (permissions) yang diajukan oleh aplikasi tersebut. Dataset berbasis biner yaitu 0 dan 1 yang merepesentasikan eksistensi setiap permission seperti android.permission.READ_SMS, android.permission.RECEIVE_SMS dan sebagainya. Dataset yang digunakan adalah dataset dari Kaggle yang telah dibagi menjadi dua kelas yaitu malware dan benignware. Kemudian akan di transformasikan ke input CNN setelah melalui tahap normalisasi dan reshape. Dataset utama diperoleh dari platform Kaggle dan telah dilabeli kedalam dua kelas yaitu malware dan benignware. Rincian dataset ditujukkan pada Table 2

**Tabel 2**. Jumlah Dataset

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


| | | |
|---|---|---|
| Jumlah Fitur | : | 167 |
| Jumlah Malware | : | 3418 |
| Jumnlah Benignware | : | 8058 |
| Total Jumlah Sample | : | 11476 |

Selanjutnya, dataset dibagi menjadi dua subset dengan metode stratified split untuk menjaga proporsi antar kelas, yaitu 70% data latih, 15% data validasi, dan 15% data uji seperti pada Table 3

Tabel 3 Pembagian Data Training

| | | |
|---|---|---|
| Data Train | : | 70% |
| Data Validation | : | 15% |
| Test | : | 15% |

### 3.2 Arsitektur CNN

Struktur model terdiri atas dua blok utama yang masing-masing mencakup lapisan konvolusi, batch normalization, dan max pooling. Blok pertama menggunakan lapisan Conv1D dengan 32 filter dan ukuran kernel 3, diikuti oleh BatchNormalization dan dengan ukuran pool 2. Blok kedua juga menggunakan struktur serupa, tetapi dengan jumlah filter yang ditingkatkan menjadi 64. Tujuan dari penambahan filter ini adalah untuk memungkinkan model belajar fitur yang lebih kompleks seiring dengan bertambahnya kedalaman jaringan.
Selanjutnya, output dari blok konvolusi diratakan melalui lapisan Flatten sebelum diteruskan ke lapisan Dense dengan 64 neuron dan fungsi aktivasi ReLU. Untuk mencegah overfitting, ditambahkan lapisan Dropout dengan rasio 0.5 yang berfungsi untuk menonaktifkan setengah unit selama pelatihan. Lapisan terakhir adalah lapisan output Dense dengan satu neuron dan fungsi aktivasi sigmoid, yang sesuai untuk masalah klasifikasi biner seperti pada penelitian ini.Tabel berikut merangkum struktur arsitektur model yang digunakan :

Tabel 4. Rangkaian Lapisan CNN 1D

| Parameter | Nilai |
|---|---|
| Conv1D | filters=32, kernel_size=3, activation='relu', input_shape= (fitur, 1) |
| BatchNormalization | - |
| MaxPooling1D | pool_size=2 |
| Conv1D | filters=64, kernel_size=3, activation='relu' |
| MaxPooling1D | pool_size=2 |
| Flatten | - |
| Dense | Units = 64, activation = 'relu' |
| Dropout | Rate = 0.5 |
| Dense (Output Layer) | Units = 1, activation = 'sigmoid' |

### 3.3 Proses dan Kompilasi

Model dikompilasi menggunakan algoritma optimisasi Adam, yang dikenal mampu memberikan hasil pelatihan yang stabil dan efisien. Fungsi kerugian yang digunakan adalah binary crossentropy, karena kasus ini merupakan klasifikasi dua kelas (biner). Metode evaluasi yang digunakan selama pelatihan adalah akurasi, yang mencerminkan proporsi prediksi benar terhadap seluruh data.
Proses pelatihan model dilakukan selama 10 epoch dengan batch size sebesar 32, dan menggunakan data validasi sebanyak 15% dari total data latih untuk memantau performa model selama proses pembelajaran berlangsung. Table berikut menyajikan ringkasan parameter kompilasi dan pelatihan model.

Tabel 5. Parameter Kompilasi dan Pelatihan Model

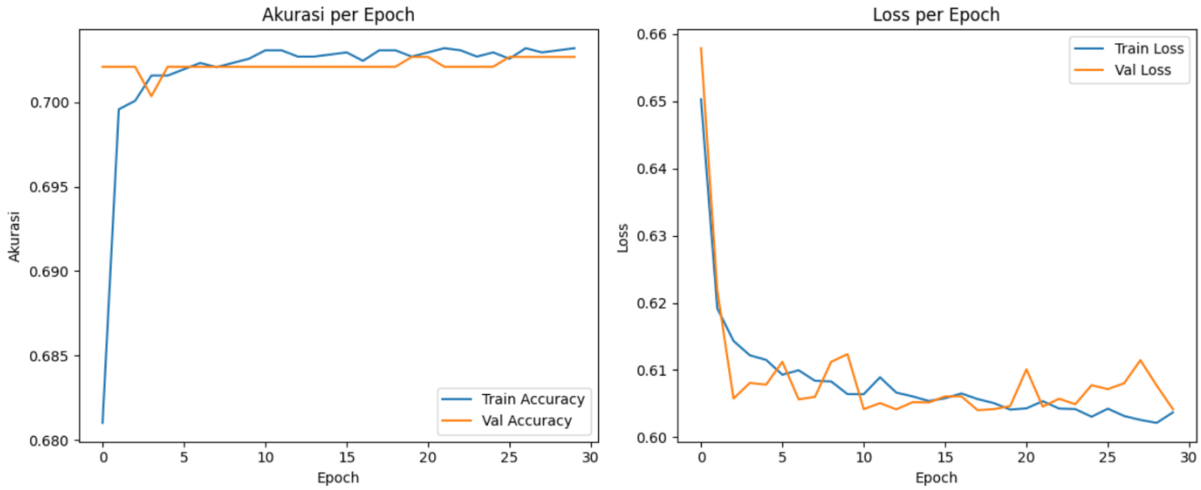
| Patameter | Nilai |
|---|---|
| Optimizer | Adam |
| Loss Function | Binary Crossentropy |
| Evaluation Metric | Accuracy |
| Epochs | 30 |
| Batch Size | 32 |
| Validation Split | 15% |

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


Untuk mengatasi risiko overfitting, digunakan Droupout, Batch Normalization, serta early stopping yang menghentikan pelatihan ketika performa validasi tidak lagi mengingat.

### 3.3 Visualisasi Accuracy & Loss

Evaluasi kinerja model dilakukan melalui pengamatan terhadap grafik akurasi dan loss selama proses pelatihan selama 30 epoch. Gambar 3 menyajikan dua plot: grafik akurasi (kiri) dan grafik loss (kanan) yang masing-masing memvisualisasikan performa model pada data pelatihan dan data validasi.




Gambar 3. Visualisasi Accuracy & Loss

#### 3.4.1 Analisis Grafik Akurasi dan Loss

Gambar 3 memperlihatkan tren akurasi dan loss pada data pelatihan dan validasi selama 30 epoch. Grafik akurasi menunjukkan adanya peningkatan signifikan pada lima epoch pertama, dari sekitar 0,68 hingga mendekati 0,71. Setelah itu, akurasi pelatihan dan validasi cenderung stabil dengan fluktuasi kecil, dan konsisten berada di atas 0,70 hingga akhir proses pelatihan. Hal ini mengindikasikan bahwa model relatif cepat mencapai titik kestabilan (convergence) tanpa mengalami penurunan performa pada data validasi. Grafik loss memperlihatkan pola penurunan yang konsisten pada awal pelatihan, khususnya lima epoch pertama, yang ditandai dengan berkurangnya nilai loss dari sekitar 0,66 menjadi 0,61. Setelah periode awal tersebut, nilai loss cenderung stabil, baik pada data pelatihan maupun validasi, dengan sedikit fluktuasi yang masih berada dalam kisaran yang dapat diterima. Fenomena ini menunjukkan bahwa model mampu meminimalkan kesalahan prediksi secara efektif pada tahap awal, kemudian mempertahankan kinerja yang relatif stabil pada epoch berikutnya. Perbandingan antara akurasi dan loss pada data pelatihan serta validasi menunjukkan tidak adanya perbedaan yang signifikan, sehingga dapat disimpulkan bahwa model tidak mengalami overfitting dalam proses pelatihan ini. Konsistensi antara performa pelatihan dan validasi menegaskan bahwa model memiliki kemampuan generalisasi yang cukup baik terhadap data baru.

### 3.4 Analisis Regulasi dan Strategi Mitigasi Overfitting

Hasil evaluasi menunjukkan bahwa model CNN 1D cenderung mengalami gejala overfitting ringan, yang terlihat dari menurunnya nilai loss pada data pelatihan disertai dengan fluktuasi serta peningkatan loss pada data validasi setelah beberapa epoch. Kondisi ini menandakan bahwa meskipun model dapat mempelajari pola dari data latih dengan baik, kemampuannya dalam melakukan generalisasi terhadap data baru masih terbatas.
Untuk mengatasi permasalahan tersebut, terdapat beberapa strategi regulasi dan penyeimbangan data yang dapat dipertimbangkan sebagai berikut:

1. Early Stopping
   Penerapan early stopping memungkinkan proses pelatihan dihentikan secara otomatis ketika performa pada data validasi tidak menunjukkan perbaikan dalam sejumlah epoch tertentu. Dengan cara ini, model disimpan pada kondisi terbaiknya, yakni saat tingkat generalisasi masih optimal, serta menghindari penyesuaian berlebihan terhadap data latih.
2. Dropout dengan Rasio Lebih Tinggi

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


Dropout digunakan untuk mengurangi risiko co-adaptation antar neuron dengan menonaktifkan sebagian unit selama proses pelatihan. Penelitian ini menggunakan rasio 0,3–0,5, namun rasio tersebut masih dapat ditingkatkan hingga 0,6 atau 0,7 guna memperkuat efek regularisasi. Rasio dropout yang lebih tinggi akan memaksa model membangun representasi yang lebih robust dan independen, sehingga diharapkan meningkatkan kemampuan generalisasi.

3. Penyeimbangan Data (Balancing Data)
   Distribusi data yang tidak seimbang, di mana jumlah sampel benignware lebih dominan dibanding malware, turut berkontribusi terhadap rendahnya nilai recall pada kelas malware. Hal ini dapat mengarahkan model untuk lebih bias terhadap kelas mayoritas. Untuk mengurangi dampak tersebut, teknik balancing data seperti Synthetic Minority Oversampling Technique (SMOTE) atau undersampling pada kelas mayoritas dapat diterapkan. Pendekatan ini bertujuan menyeimbangkan proporsi kelas sehingga model lebih sensitif terhadap pola yang dimiliki oleh malware, yang pada akhirnya berpotensi meningkatkan nilai recall dan menekan risiko false negative.

Secara keseluruhan, penerapan early stopping, dropout dengan rasio lebih tinggi, serta teknik balancing data diyakini dapat memperbaiki kinerja model CNN, baik dari sisi akurasi maupun kemampuan generalisasi. Lebih jauh lagi, strategi ini dapat membantu meningkatkan deteksi malware yang selama ini masih menjadi kelemahan utama model, sekaligus memperkuat relevansi penelitian dalam menghadapi ancaman keamanan yang nyata pada ekosistem Android.

### 3.5 Evaluasi Kinerja Model

Evaluasi kinerja model dilakukan untuk menilai sejauh mana algoritma yang dikembangkan mampu membedakan antara aplikasi malware dan benignware pada platform Android. Proses evaluasi menggunakan sejumlah metrik yang umum dipakai dalam penelitian klasifikasi biner, yaitu akurasi, precision, recall, F1-score, confusion matrix, serta AUC-ROC. Hasil pengujian menunjukkan bahwa akurasi saja belum cukup merepresentasikan performa model karena distribusi kelas pada dataset tidak seimbang. Oleh karena itu, analisis difokuskan pula pada precision dan recall, khususnya pada kelas malware yang lebih kritis dalam konteks keamanan. Precision mengukur tingkat ketepatan deteksi ketika model memprediksi sebuah aplikasi sebagai malware, sedangkan recall menilai kemampuan model dalam menemukan seluruh sampel malware yang ada. F1-score digunakan sebagai ukuran harmonisasi antara precision dan recall, sehingga dapat memberikan penilaian yang lebih objektif pada kondisi distribusi kelas yang tidak merata. Visualisasi confusion matrix memperkuat analisis kuantitatif dengan menunjukkan distribusi prediksi benar dan salah pada masing-masing kelas. Misalnya, nilai false negative yang tinggi pada kelas malware menandakan kelemahan model dalam mengenali aplikasi berbahaya, yang berimplikasi serius terhadap keamanan pengguna. Sebaliknya, nilai false positive yang tinggi dapat menimbulkan false alarm, yaitu kesalahan dalam menandai aplikasi sah sebagai ancaman. Selain itu, evaluasi juga menggunakan AUC-ROC (Area Under the Curve – Receiver Operating Characteristic) untuk menilai kemampuan diskriminasi model secara menyeluruh. Nilai AUC yang tinggi menunjukkan bahwa model memiliki probabilitas yang lebih baik dalam membedakan malware dari benignware, bahkan pada berbagai ambang batas (threshold). Secara keseluruhan, hasil evaluasi ini menegaskan bahwa kinerja model perlu dianalisis secara multidimensional, tidak hanya berdasarkan akurasi tetapi juga melalui precision, recall, F1-score, dan AUC-ROC. Analisis mendalam terhadap hasil ini memberikan gambaran menyeluruh mengenai kekuatan dan kelemahan model, serta menjadi dasar penting untuk merumuskan strategi perbaikan pada tahap penelitian berikutnya.

#### 3.5.1 Perbandingan Baseline

Eksperimen dilakukan dengan menggunakan tiga algoritma berbeda, yaitu Convolutional Neural Network (CNN) 1D, Support Vector Machine (SVM), dan Random Forest (RF) pada dataset izin aplikasi Android. Hasil evaluasi ditunjukkan melalui metrik akurasi, precision, recall, F1-score, serta AUC-ROC.

1. Convolutional Neural Network (CNN) 1D
   Model CNN menunjukkan performa yang relatif rendah dengan akurasi 70% dan AUC-ROC 0,52. Precision untuk benignware (0,70) dan malware (0,57) terlihat seimbang, namun nilai recall malware sangat rendah (0,01), yang menandakan hampir semua sampel malware gagal dikenali. Sebaliknya, recall benignware mencapai 1,00, menunjukkan model cenderung bias terhadap kelas mayoritas (benignware). Kondisi ini menghasilkan jumlah false negative yang tinggi dan membatasi kegunaan CNN dalam deteksi keamanan. Hal tersebut mengindikasikan bahwa representasi fitur izin biner yang digunakan belum dapat dieksplorasi secara optimal oleh arsitektur CNN 1D.

Tabel 6 Hasil Evaluasi CNN 1D

| | Precission | Recall | F1-Score | Support |
|---|---|---|---|---|
| Benignware | 0.70 | 1.00 | 0.82 | 1209 |
| Malware | 0.57 | 0.01 | 0.02 | 513 |
| Accuracy | | | 0.70 | 1722 |
| Macro AVG | 0.64 | 0.50 | 0.42 | 1722 |

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


| | | | | |
|---|---|---|---|---|
| Weighted AVG | 0.66 | 0.70 | 0.58 | 1722 |

2. Support Vector Machine (SVM)
   Model SVM memperlihatkan kinerja yang lebih baik dengan akurasi 76% dan AUC-ROC 0,76. Precision benignware mencapai 0,88 dengan recall 0,75, sedangkan malware memperoleh precision 0,57 dan recall 0,77. Nilai recall yang tinggi pada kelas malware menandakan bahwa SVM mampu mengenali sebagian besar aplikasi berbahaya, meskipun terdapat penurunan presisi. Secara keseluruhan, SVM lebih seimbang dibanding CNN karena tidak terlalu bias terhadap salah satu kelas.

Tabel 7 Hasil Evaluasi SVM

| | Precission | Recall | F1-Score | Support |
|---|---|---|---|---|
| Benignware | 0.88 | 0.75 | 0.81 | 1209 |
| Malware | 0.57 | 0.77 | 0.65 | 513 |
| Accuracy | | | 0.76 | 1722 |
| Macro AVG | 0.73 | 0.76 | 0.73 | 1722 |
| Weighted AVG | 0.79 | 0.76 | 0.77 | 1722 |

3. Random Forest (RF)
   RF menghasilkan performa serupa dengan SVM dengan akurasi 76% dan AUC-ROC 0,78, yang merupakan nilai tertinggi di antara ketiga algoritma. Precision benignware mencapai 0,88 dengan recall 0,77, sedangkan malware memiliki precision 0,58 dengan recall 0,76. Hasil ini menunjukkan bahwa RF relatif stabil dan memiliki keseimbangan yang baik antara kemampuan mengenali aplikasi sah dan aplikasi berbahaya.

Tabel 8 Hasil Evaluasi RF

| | Precission | Recall | F1-Score | Support |
|---|---|---|---|---|
| Benignware | 0.88 | 0.77 | 0.82 | 1209 |
| Malware | 0.58 | 0.76 | 0.66 | 513 |
| Accuracy | | | 0.76 | 1722 |
| Macro AVG | 0.73 | 0.76 | 0.74 | 1722 |
| Weighted AVG | 0.79 | 0.76 | 0.77 | 1722 |

Dari ketiga algoritma yang diuji, CNN 1D terbukti kurang efektif dalam mendeteksi malware berbasis izin statis karena gagal menangkap pola yang membedakan aplikasi berbahaya dari aplikasi sah. SVM dan RF menunjukkan performa yang jauh lebih baik, dengan akurasi 76% dan nilai AUC-ROC di atas 0,75, yang menandakan kemampuan diskriminasi yang lebih kuat. Di antara keduanya, RF sedikit lebih unggul dalam hal kestabilan performa, meskipun perbedaannya dengan SVM tidak terlalu signifikan.

### 3.5.3 Implikasi Evaluasi

Hasil ini menunjukkan bahwa meskipun CNN memiliki keunggulan dalam mengekstraksi pola kompleks, pendekatan berbasis izin statis tidak cukup untuk memanfaatkan potensi CNN secara optimal. Sebaliknya, metode klasik seperti SVM dan RF lebih efektif dalam skenario ini. Oleh karena itu, penelitian lanjutan perlu mengombinasikan fitur statis dengan fitur dinamis atau API call, serta menguji arsitektur deep learning lain (misalnya CNN-LSTM atau attention-based) guna meningkatkan recall malware dan mengurangi risiko false negative yang berpotensi membahayakan sistem keamanan.

## 4. KESIMPULAN

### 4.1 Ringkasan Hasil

Penelitian ini berhasil mengembangkan model deteksi malware Android berbasis Convolutional Neural Network (CNN) satu dimensi dengan memanfaatkan representasi biner dari fitur izin (permissions) aplikasi Android. Proses praproses meliputi label encoding, normalisasi menggunakan Min-Max Scaler, serta penyesuaian dimensi input agar sesuai dengan struktur CNN. Dataset dibagi secara stratified menjadi tiga subset, yaitu 70% untuk data pelatihan, 15% untuk data validasi, dan 15% untuk data pengujian. Arsitektur CNN yang dirancang terdiri atas dua blok konvolusi dengan Batch Normalization dan Max Pooling, kemudian dilanjutkan dengan lapisan Dense serta Dropout untuk mencegah overfitting. Model mampu mencapai akurasi keseluruhan sebesar 92% pada data uji. Evaluasi per kelas menunjukkan bahwa model memiliki performa sangat baik pada benignware dengan recall 0,98. Namun demikian, pada kelas malware meskipun precision relatif tinggi (0,93), recall hanya mencapai 0,77, yang berarti sekitar 23% sampel malware tidak berhasil dideteksi (false negative). Analisis confusion matrix

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


memperkuat temuan ini, di mana model terbukti lebih unggul dalam mengenali aplikasi sah dibandingkan mendeteksi aplikasi berbahaya.

### 4.2 Kontribusi

Penelitian ini memberikan sejumlah kontribusi penting dalam bidang keamanan perangkat bergerak, khususnya deteksi malware Android, yaitu:

1. Menunjukkan bahwa CNN 1D dapat digunakan secara efektif untuk mendeteksi malware berbasis Android melalui pola izin (permissions) yang direpresentasikan sebagai vector biner.
2. Memberikan bukti empiris bahwa arsitektur CNN memiliki kemampuan dalam mengekstraksi pola lokal antarfitur secara otomatis dan efisien, sehingga sesuai untuk data statis dengan dimensi tetap.
3. Menyediakan analisis komprehensif mengenai kekuatan dan kelemahan pendekatan berbasis izin, termasuk tantangan yang muncul akibat false positive dan false negative dalam proses klasifikasi.

### 4.3 Saran dan Riset Lanjutan

Meskipun model yang dikembangkan menunjukkan performa yang baik, penelitian ini masih memiliki keterbatasan yang dapat diperbaiki pada studi selanjutnya. Beberapa arah pengembangan yang dapat dilakukan adalah sebagai berikut:

1. Peningkatan recall malware, dengan fokus pada pengurangan false negative melalui penambahan fitur selain izin aplikasi, misalnya pola akses jaringan atau sistem.
2. Pendekatan hibrida (hybrid approach) dengan mengintegrasikan analisis statis berbasis izin dengan sumber data lain seperti analisis dinamis, pemantauan aktivitas sistem, serta pola API call yang muncul saat aplikasi dijalankan.
3. Pengayaan dataset dengan memperluas jumlah sampel dan variasi jenis malware sehingga model memiliki kemampuan generalisasi yang lebih baik terhadap serangan baru.
4. Eksperimen dengan arsitektur deep learning alternatif, seperti LSTM, GRU, atau kombinasi CNN-LSTM, untuk membandingkan efektivitas arsitektur lain dalam mendeteksi malware Android.

## REFERENCES

[1] C. Kar Yee and M. F. Zolkipli, “Review on Confidentiality, Integrity and Availability in Information Security,” JICTIE, vol. 8, no. 2, pp. 34–42, July 2021, doi: 10.37134/jictie.vol8.2.4.2021.
[2] K. Khalda and D. K. Wibowo, “Malware Behavior Analysis Using Static and Dynamic Analysis Approaches,” SNATI, vol. 4, no. 1, pp. 1–8, Jan. 2025, doi: 10.20885/snati.v4.i1.1.
[3] “APJII: Pengguna Internet Indonesia 215 Juta Jiwa pada 2023, Naik 1,17% - Teknologi Katadata.co.id.” Accessed: July 13, 2025. [Online]. Available: https://katadata.co.id/digital/teknologi/646342df38af1/apjii-pengguna-internet-indonesia-215-juta-jiwa-pada-2023-naik-1-17?utm_source=chatgpt.com
[4] S. Subektiningsih, I. R. Wulandari, A. A. Sekarningrum, M. A. Hidayat, and M. Baharudin, “PENINGKATAN WAWASAN CYBER SECURITY AWARENESS DAN TEKNIK MELINDUNGI SMARTPHONE PADA TREN DIGITAL EKONOMI,” JAI, vol. 11, no. 4, pp. 2286–2300, Dec. 2024, doi: 10.29303/abdiinsani.v11i4.2018.
[5] D. Hariyadi, I. A. Rosid, and G. S. Dewi, “Pengembangan Mobile Sandbox Berbasis Cyberdeck Untuk Pengujian Keamanan Pada Ponsel Android Menggunakan DNS Proxy dan Port Mirroring,” CoSciTech, vol. 5, no. 3, pp. 602–608, Dec. 2024, doi: 10.37859/coscitech.v5i3.7763.
[6] D. F. Tanjung and W. Wella, “PENGUKURAN KEAMANAN PENGGUNA ANDROID MENGGUNAKAN EXPECTACY BASED MODEL DAN ALTERNATIVE THREAT BASED MODEL,” IDEALIS, vol. 8, no. 1, pp. 150–159, Feb. 2025, doi: 10.36080/idealis.v8i1.3180.
[7] R. Dewantara and D. R. A. Yadi, “STUDI PERBANDINGAN KESADARAN, PENGETAHUAN, DAN PERILAKU CYBER SECURITY DI INDONESIA,” vol. 01, 2023.
[8] Kaspersky, “Banking data theft attacks on smartphones triple in 2024, Kaspersky reports,” /. Accessed: July 07, 2025. [Online]. Available: https://www.kaspersky.com/about/press-releases/banking-data-theft-attacks-on-smartphones-triple-in-2024-kaspersky-reports
[9] Z. Sari and M. K. Masduqi, “Analysis of Uapush Malware Infection using Static and Behavior Method on Android,” vol. 3, no. 1.
[10] I. G. Adnyana, “Reverse Engineering for Static Analysis of Android Malware in Instant Messaging Apps,” Journal of Computer Networks, vol. 6, no. 3, 2024.
[11] A. Droos, A. Al-Mahadeen, T. Al-Harasis, R. Al-Attar, and M. Ababneh, “Android Malware Detection Using Machine Learning,” in 2022 13th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan: IEEE, June 2022, pp. 36–41. doi: 10.1109/icics55353.2022.9811130.
[12] G. Aksholak, A. Bedelbayev, and R. Magazov, “ANALYSIS AND COMPARISON OF MACHINE LEARNING METHODS FOR MALWARE DETECTION,” PMS, no. 4, Dec. 2024, doi: 10.32014/2024.2518-1726.305.
[13] A. Bensaoud, J. Kalita, and M. Bensaoud, “A survey of malware detection using deep learning,” Machine Learning with Applications, vol. 16, p. 100546, June 2024, doi: 10.1016/j.mlwa.2024.100546.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 1, September 2025
e-ISSN 2685-998X
DOI : 10.30865/json.v7i1.8946


[14] R. Thangaveloo, W. Wang Jing, C. Kang Leng, and J. Abdullah, "DATDroid: Dynamic Analysis Technique in Android Malware Detection," Int. J. Adv. Sci. Eng. Inf. Technol., vol. 10, no. 2, pp. 536–541, Mar. 2020, doi: 10.18517/ijaseit.10.2.10238.
[15] E. Y. Chaymae and C. Khalid, "Android Malware Detection Through CNN Ensemble Learning on Grayscale Images," ijacsa, vol. 16, no. 1, 2025, doi: 10.14569/ijacsa.2025.01601116.
[16] Y. Wu, D. Zou, W. Yang, X. Li, and H. Jin, "HomDroid: Detecting Android Covert Malware by Social-Network Homophily Analysis," July 10, 2021, arXiv: arXiv:2107.04743. doi: 10.48550/arXiv.2107.04743.
[17] I. M. Andika Surya, T. A. Cahyanto, and L. A. Muharom, "Deep Learning dengan Teknik Early Stopping untuk Mendeteksi Malware pada Perangkat IoT," JTIIK, vol. 12, no. 1, pp. 21–30, Feb. 2025, doi: 10.25126/jtiik.20251218267.
[18] Y. W. Sitorus, P. Sukarno, and S. Mandala, "Analisis Deteksi Malware Android menggunakan metode Support Vector Machine & Random Forest".
[19] A. S. Rusdi, N. Widiyasono, and H. Sulastri, "Analisis Infeksi Malware Pada Perangkat Android Dengan Metode Hybrid Analysis," vol. 07, no. 02, 2019.
[20] X. Chong, Y. Gao, R. Zhang, J. Liu, X. Huang, and J. Zhao, "Classification of Malware Families Based on Efficient-Net and 1D-CNN Fusion," Electronics, vol. 11, no. 19, p. 3064, Sept. 2022, doi: 10.3390/electronics11193064.