

Komparasi Algoritma Neural Network dan K-Nearest Neighbor Dalam Mendeteksi Malware Android

Andi Ramadhan, Lindawati*, Martinus Mujur Rose

Teknik Elektro, Sarjana Terapan Teknik Telekomunikasi, Politeknik Negeri Sriwijaya, Palembang, Indonesia

Email: ¹andiramadhan23@gmail.com, ²*lindawati@polsri.ac.id, ³mujurrose@yahoo.com

Email Penulis Korespondensi: lindawati@polsri.ac.id

Submitted: 29/05/2023; Accepted: 27/06/2023; Published: 29/06/2023

Abstrak—Laporan Badan Siber dan Sandi Negara (BSSN) mencatat adanya sekitar 100 juta kasus serangan siber di Indonesia hingga bulan April 2022, dengan ransomware dan malware sebagai jenis serangan yang paling umum terdeteksi. Dalam konteks ini, perkembangan malware Android yang semakin canggih dan sulit terdeteksi menjadi ancaman serius, terutama dengan penetrasi yang berhasil ke Google Play Store. Oleh karena itu, deteksi dini terhadap malware Android menjadi penting. Penelitian ini bertujuan untuk membandingkan performa dua algoritma machine learning, yaitu Neural Network dan K-Nearest Neighbors (KNN), dalam mendeteksi malware pada platform Android. Dataset telah diproses dan dibagi menjadi data training dan testing. Kedua algoritma dilatih menggunakan data training, dan hasilnya divalidasi dan dievaluasi. Hasil penelitian menunjukkan bahwa Neural Network mencapai performa terbaik dengan akurasi 97%, presisi 97%, recall 97%, dan F1-score 97%. Sementara itu, KNN memiliki performa yang sedikit lebih rendah dengan akurasi 95%, presisi 96%, recall 95%, dan F1-score 95%. Kesimpulannya, Neural Network memiliki performa yang lebih baik daripada KNN dalam mendeteksi malware Android berdasarkan akurasi dan konsistensi klasifikasi. Saran penelitian selanjutnya melibatkan penggunaan algoritma lain, penggunaan dataset yang lebih luas dan representatif, serta penambahan fitur dan optimisasi parameter. Penelitian ini memberikan kontribusi dalam pengembangan solusi yang akurat dan efektif untuk mendeteksi dan memisahkan aplikasi Android yang berpotensi berbahaya.

Kata Kunci: Malware Android; Deteksi Dini; Neural Network; K-Nearest Neighbors (KNN); Performa algoritma

Abstract—The report from the State Cyber and Cryptography Agency (BSSN) recorded approximately 100 million cases of cyber attacks in Indonesia until April 2022, with ransomware and malware being the most commonly detected types of attacks. In this context, the increasingly sophisticated and hard-to-detect Android malware poses a serious threat, especially with successful penetrations into the Google Play Store. Therefore, early detection of Android malware is crucial. This research aims to compare the performance of two machine learning algorithms, Neural Network and K-Nearest Neighbors (KNN), in detecting malware on the Android platform. The dataset has been processed and divided into training and testing data. Both algorithms are trained using the training data, and their results are validated and evaluated. The research findings show that Neural Network achieves the best performance with an accuracy of 97%, precision of 97%, recall of 97%, and F1-score of 97%. Meanwhile, KNN performs slightly lower with an accuracy of 95%, precision of 96%, recall of 95%, and F1-score of 95%. In conclusion, Neural Network outperforms KNN in detecting Android malware based on accuracy and classification consistency. Further research suggestions involve the use of other algorithms, broader and more representative datasets, as well as the addition of features and parameter optimization. This research contributes to the development of accurate and effective solutions for detecting and identifying potentially harmful Android applications.

Keywords: Android Malware; Early Detection; Neural Network; K-Nearest Neighbors (KNN); Algorithm Performance

1. PENDAHULUAN

Kemajuan teknologi informasi di era sekarang telah memberikan dampak positif yang signifikan di berbagai sektor, salah satunya pada perangkat bergerak seperti *smartphone*. Sebuah *smartphone* pada dasarnya menyediakan sistem operasi dan aplikasi yang mempermudah penggunaannya. Beberapa sistem operasi yang umum digunakan meliputi iOS, Blackberry OS, dan Android. Dari berbagai pilihan tersebut, Android menjadi salah satu sistem operasi yang paling banyak digunakan. Menurut CEO Google, Erich Schmidt, sekitar 1,4 juta perangkat Android diaktifkan setiap hari [1]. Android menggunakan salah satu sistem operasi diantaranya sistem berbasis linux. Hingga saat ini, Android terus mengalami perkembangan baik dari segi sistem maupun aplikasi. Ketersediaan banyak *smartphone* Android yang diproduksi telah mendorong peningkatan fitur-fitur di dalamnya. Salah satu perkembangan yang signifikan adalah fitur keamanan yang semakin pesat. Sebelumnya, fitur keamanan hanya terbatas pada PIN unlock, namun sekarang semakin beragam dengan adanya fitur seperti *face unlock*, *face and voice unlock*, dan *pattern unlock* [2].

Malware merupakan istilah yang menggabungkan kata "*malicious*" (jahat) dan "*software*" (perangkat lunak), mengacu pada perangkat lunak berbahaya yang sengaja dirancang untuk merusak sistem, mencuri data, dan mendapatkan akses tanpa izin pada suatu sistem. Penyebaran *malware* dapat terjadi melalui berbagai metode, termasuk serangan phishing melalui email, rekayasa sosial, dan pengunduhan perangkat lunak yang tidak aman. Tujuan dari serangan *malware* ini adalah untuk mengambil informasi rahasia dari pemilik seperti kata sandi dan email, serta menyebarkan spam. Salah satu contoh umum dari serangan *malware* adalah melalui aplikasi berbahaya yang berusaha mendapatkan akses ke perangkat tanpa izin dari pengguna atau sistem operasi. Mengingat risiko dan dampak negatif yang ditimbulkan oleh serangan *malware*, diperlukan analisis yang lebih lanjut untuk melawan ancaman ini [3].

Berdasarkan laporan Badan Siber dan Sandi Negara (BSSN) hingga bulan April 2022, terjadi sekitar 100 juta kasus serangan siber di Indonesia. *Ransomware* dan *malware* merupakan jenis serangan siber yang paling umum terdeteksi oleh BSSN. Mengingat pernyataan tersebut, terlihat bahwa *malware* Android semakin berkembang dalam

menghindari sistem keamanan dan berhasil menyusup ke Google Play Store. Karena tingkat kecanggihan *malware* semakin meningkat dan sulit terdeteksi, sangat penting untuk melakukan deteksi terhadap *malware* tersebut. Dikarenakan adanya jumlah serangan *malware* yang signifikan dan populasi pengguna Android yang sangat luas, hal ini menjadi sebuah masalah bagi pengguna Android. Oleh karena itu, penulis bertujuan untuk melakukan penelitian menggunakan sistem machine learning guna mengidentifikasi metode klasifikasi yang mampu memberikan performa tinggi dalam mendeteksi serangan *malware* secara dini [4].

Machine learning (ML) adalah bentuk penerapan dari Kecerdasan Buatan (*Artificial Intelligence*, AI) yang bertujuan untuk mengembangkan sistem yang mampu belajar secara mandiri tanpa memerlukan pemrograman berulang kali. ML bergantung pada penggunaan data (data pelatihan) dalam proses pembelajarannya sebelum menghasilkan output yang diinginkan [5]. *Machine learning* adalah sebuah konsep komputasi yang melibatkan penggunaan algoritma matematika dan komputer untuk mempelajari pola-pola yang terdapat dalam data dan menghasilkan prediksi untuk masa depan. Proses pembelajaran ini melibatkan dua proses tahapan yang penting, yaitu pelatihan (*training*) dan pengujian (*testing*). Terdapat tiga kategori utama di dalam *machine learning*, yaitu *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning* [6]. Pembelajaran mesin (Machine Learning/ML) termasuk dari kecerdasan buatan yang menggunakan algoritma secara teratur untuk menggabungkan hubungan-hubungan yang mendasari antara data dan informasi. Sebagai contoh, sistem ML dapat diinstruksikan menggunakan sistem pengenalan ucapan otomatis (seperti Siri pada iPhone) untuk mengubah informasi suara menjadi struktur yang mengandung makna yang diungkapkan dalam bentuk urutan kata-kata [7].

Dalam beberapa penelitian terdahulu yang relevan, ditemukan beberapa temuan penting. Salah satunya, sebuah penelitian yang dilakukan oleh Alzaylae, Yerima dan Sezer, pada tahun 2020 [8] mengevaluasi suatu sistem bernama DL-Droid yang menggunakan pembelajaran mendalam (*deep learning*) untuk mendeteksi *malware* Android dengan menggunakan perangkat nyata. Hasil penelitian menunjukkan bahwa DL-Droid berhasil mencapai tingkat deteksi sebesar 97,8% (dengan fitur dinamis saja) dan 99,6% (dengan fitur dinamis + statis), yang lebih unggul dibandingkan dengan teknik pembelajaran mesin tradisional. Penelitian ini juga membandingkan metode generasi input berbasis keadaan dengan pendekatan stateless yang umum digunakan dalam sistem pembelajaran mendalam. Perbedaan antara penelitian yang dilakukan terletak pada aspek fokus, metode yang digunakan, dan hasil yang ditemukan. Dalam penelitian yang dilakukan mengenai komparasi antara algoritma NN dan K-NN dalam mendeteksi *malware* Android menggunakan dataset Android permission. Penelitian yang dilakukan lebih berfokus pada perbandingan performa dua algoritma tersebut dalam mendeteksi *malware*. Sementara itu, penelitian yang dilakukan oleh [8] fokus pada evaluasi sistem DL-Droid yang menggunakan pembelajaran mendalam untuk mendeteksi *malware* Android dengan perangkat nyata. Mereka membandingkan performa DL-Droid dengan teknik pembelajaran mesin tradisional dan mempertimbangkan metode generasi input berbasis keadaan dengan pendekatan stateless.

Penelitian lainnya yang dilakukan oleh Diana, Indrajit dan Dazki, pada tahun 2022 [9], melakukan komparasi antara algoritma *Naïve Bayes*, *Logistic Regression*, dan *Support Vector Machine* dalam mengklasifikasikan *malware* Android berdasarkan file *Application Package Kit* (APK). Hasil pengujian menunjukkan bahwa algoritma *Naive Bayes* memiliki akurasi sebesar 97,75%, sedangkan *Logistic Regression* memiliki akurasi sebesar 88,75% dan *Support Vector Machine* mencapai akurasi sebesar 96,75%. Perbedaan antara penelitian yang dilakukan dan penelitian yang dibuat oleh penulis [9] terletak pada poin fokus, pendekatan metodologi, dan dataset yang digunakan. Penelitian yang dilakukan yaitu membandingkan algoritma NN dan K-NN dalam mendeteksi *malware* Android menggunakan dataset Android permission, lebih terfokus pada perbandingan dua algoritma tersebut. Peneliti menggunakan fitur statis dari aplikasi Android dalam upaya mendeteksi *malware*. Penelitian yang dilakukan oleh penulis [9] membandingkan algoritma *Naïve Bayes*, *Logistic Regression*, dan *Support Vector Machine* dalam mengklasifikasikan *malware* Android berdasarkan file APK. Fokus penelitian mereka adalah pada perbandingan antara tiga algoritma tersebut dalam hal akurasi klasifikasi.

Penelitian oleh R.B. Hadiprakoso, W.R. Aditya, F.N. Pramitha pada tahun 2022 [1] melakukan analisis deteksi *malware* pada platform Android menggunakan pendekatan supervised machine learning. Hasil penelitian menunjukkan bahwa penggunaan model klasifikasi SVM memberikan kinerja terbaik dengan akurasi mencapai 96,94% dan AUC (Area Under Curve) sebesar 95%. Penelitian yang akan dilakukan memiliki perbedaan dengan penelitian [1] dalam hal metode yang digunakan, fokus penelitian, dan hasil temuan yang diharapkan. Penelitian yang akan dilakukan akan berfokus pada perbandingan antara algoritma NN dan K-NN dalam mendeteksi *malware* pada platform Android dengan menggunakan dataset Android permission. Penelitian ini akan membandingkan performa kedua algoritma menggunakan confusion matrix dalam hal deteksi *malware*. Sementara itu, penelitian yang dilakukan oleh penulis [1] lebih menitikberatkan pada analisis deteksi *malware* Android secara umum dengan menggunakan algoritma supervised machine learning. Fokus penelitian tersebut terutama berada pada penerapan algoritma SVM sebagai model klasifikasi dalam mendeteksi *malware* Android.

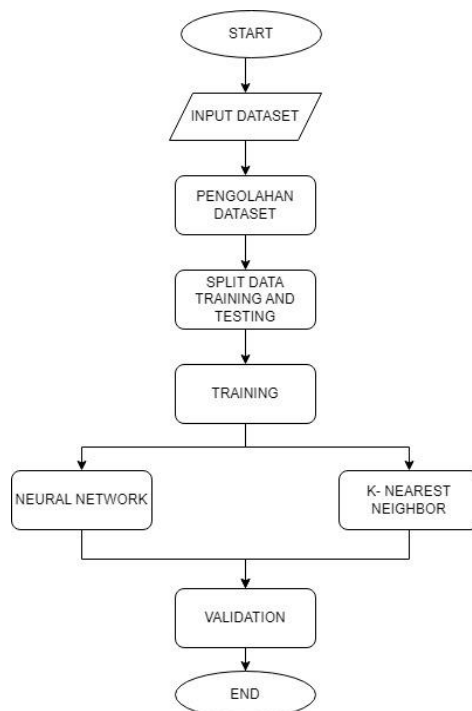
Penelitian lainnya, yang dilakukan oleh R. B. Hadiprakoso, N. Qomariasih, dan R. N. Yasa, pada tahun 2021 [10], menggunakan pendekatan analisis hibrida dengan deep learning untuk mengidentifikasi *malware* Android. Hasil validasi model dengan menggunakan algoritma LSTM memiliki akurasi sebesar 98,7% dan recall sebesar 97,9%. Perbedaan antara penelitian yang dilakukan dengan penelitian yang disebutkan dalam [10] terletak pada pendekatan dan metode yang digunakan. Penelitian [10] menerapkan sebuah pendekatan analisis hibrida yang menggabungkan deep learning, khususnya algoritma LSTM, dalam mengidentifikasi *malware* Android. Sementara itu, penelitian yang peneliti lakukan memfokuskan pada perbandingan antara algoritma NN dan K-NN dalam mendeteksi *malware*

Android menggunakan dataset Android *permission*. Terakhir, penelitian oleh Sitorus, Sukarno dan Mandala, pada tahun 2021 [11] membandingkan metode *Support Vector Machine* (SVM) dan *Random Forest* dalam analisis deteksi *malware* Android. Hasilnya menunjukkan bahwa SVM memiliki nilai presisi, *recall*, dan *F1-score* sebesar 97%, serta akurasi sebesar 96,23%. Sementara itu, metode *Random Forest* menghasilkan nilai presisi, *recall*, dan akurasi sebesar 99% dan 98,99% secara berturut-turut. Perbedaan antara penelitian yang dilakukan terletak pada metode yang dibandingkan dan hasil yang ditemukan. Penelitian [11] membandingkan SVM dan *Random Forest* dalam analisis deteksi *malware* Android, sedangkan penelitian yang peneliti lakukan dengan membandingkan algoritma NN dan KNN. Hasil penelitian [11] menunjukkan bahwa metode *Random Forest* memiliki performa yang lebih baik dengan presisi, *recall*, dan akurasi yang lebih tinggi dibandingkan dengan SVM.

Penelitian ini bertujuan untuk membandingkan performa dua metode deteksi, yaitu *Neural Network* dan *K-Nearest Neighbor*, dalam mendeteksi *malware* pada dataset aplikasi Android. Fokus utama penelitian ini adalah pada analisis statis, yang merupakan pendekatan efisien dan sederhana untuk mengamati *malware* tanpa harus menjalankan secara langsung aplikasi *malware* tersebut. Dalam penelitian ini, penulis menggunakan dataset yang terdiri dari aplikasi Android yang telah dideteksi sebagai benign (tidak berbahaya) atau *malware*. Dua metode deteksi yang diuji adalah *Neural Network* dan *K-Nearest Neighbor*. Tujuan penelitian ini adalah untuk menentukan dari kedua algoritma tersebut yang memiliki hasil yang bagus dalam melakukan deteksi *malware* menggunakan dataset yang digunakan pada penelitian ini. Harapan penelitian ini adalah dapat memberikan kontribusi penting dalam pengembangan sistem keamanan Android dan memperkuat perlindungan terhadap serangan *malware*. Dengan menentukan metode deteksi yang optimal, diharapkan penelitian ini dapat menjadi dasar bagi pengembangan solusi yang lebih efektif dalam mendeteksi dan mencegah ancaman *malware* pada perangkat Android, sehingga pengguna dapat lebih aman dan terlindungi saat menggunakan aplikasi-aplikasi tersebut.

2. METODOLOGI PENELITIAN

Proses perancangan penelitian dimulai dengan membuat suatu diagram blok sistem yang mencakup keseluruhan sistem. Flowchart memiliki peran yang sangat penting dalam perancangan studi, karena memberikan gambaran tentang bagaimana rangkaian penelitian beroperasi secara keseluruhan. Melalui flowchart ini, kita dapat memahami langkah-langkah yang diperlukan dalam metode penelitian yang sedang dilakukan. Dengan demikian, diagram blok penelitian yang lengkap membantu kita menciptakan sebuah sistem yang dapat berfungsi dengan baik. Pada Gambar 1, merupakan tahapan-tahapan yang terlibat dalam metode penelitian yang sedang dijalankan.



Gambar 1. Tahapan Penelitian

Gambar diatas dimulai dengan langkah "Start", yang menunjukkan awal dari proses. Kemudian, dilakukan langkah "Input Dataset", di mana dataset yang akan digunakan dimasukkan ke dalam sistem. Setelah itu, langkah "Pengolahan Dataset" dilakukan untuk membersihkan dan mengubah dataset menjadi format yang lebih sesuai untuk analisis. Langkah selanjutnya yaitu "Split Data Training Dataset" dilakukan. Pada langkah ini, dataset dibagi dalam dua bagian yaitu data pelatihan (*training data*) dan data pengujian (*testing data*). Data pelatihan digunakan untuk melakukan pelatihan terhadap algoritma *Neural Network* (NN) dan KNN (*K-Nearest Neighbors*). Langkah selanjutnya

adalah "*Training* Algoritma NN dan KNN". Pada tahap ini, algoritma NN dan KNN dilatih menggunakan data pelatihan yang telah dipersiapkan sebelumnya. Pelatihan ini bertujuan untuk mengembangkan model yang dapat melakukan prediksi atau klasifikasi berdasarkan fitur-fitur yang ada pada dataset. Setelah proses pelatihan selesai, langkah "*Validation*" dilakukan. Pada tahap ini, model yang telah dilatih diuji menggunakan data pengujian yang sebelumnya dipisahkan. Tujuan dari validasi ini adalah untuk mengevaluasi sejauh mana model yang dikembangkan dapat menghasilkan prediksi yang akurat dan dapat diandalkan. Terakhir, setelah semua tahapan sebelumnya selesai dilakukan, flowchart berakhir dengan langkah "Selesai" yang menunjukkan bahwa proses telah selesai dan dapat dilanjutkan ke langkah-langkah berikutnya jika ada. Secara keseluruhan, flowchart ini menggambarkan langkah-langkah yang harus diikuti untuk melatih dan menguji algoritma neural network dan KNN menggunakan dataset yang telah diinput dan dipreproses. Gambar tersebut memberikan panduan visual yang jelas tentang urutan tindakan yang harus diambil dalam proses tersebut.

2.1 Data Mining

Data mining merupakan suatu proses eksplorasi data yang bertujuan untuk mendapatkan informasi baru yang tersembunyi dalam kumpulan data. Proses ini menggunakan metode dan algoritma tertentu untuk menggali pola, melakukan prediksi, membuat estimasi, atau mengelompokkan data. Penerapan data mining sangat umum digunakan dalam berbagai disiplin ilmu untuk memfasilitasi dalam menyelesaikan masalah yang berkaitan dengan pengolahan data. Dalam konteks ini, data mining berperan penting dalam mengungkap wawasan baru dan memanfaatkan potensi data yang ada [12].

2.2 Dataset

Kami mengumpulkan satu set data dari sumber [13]. Data ini terdiri dari 1436 atribut fitur yang diekstrak dari 100.000 aplikasi. Dataset ini mencakup atribut *permission* yang diekstrak dari setiap aplikasi. Setiap aplikasi diberi nilai biner (1/0) berdasarkan keberadaan *permission* pada aplikasi tersebut, dengan nilai 1 menunjukkan adanya *permission* dan nilai 0 menunjukkan tidak adanya *permission*. Dataset ini akan digunakan sebagai data pelatihan dan data uji untuk algoritma *machine learning*.

2.3 Pengolahan Data

Kami melakukan langkah awal dalam pengolahan data dengan maksud mengurangi data yang tidak relevan atau yang memiliki atribut yang tidak lengkap. Selain itu, kami juga melakukan transformasi terhadap nilai-nilai yang redundan atau memiliki variasi yang terlalu besar. Hal ini memungkinkan kami untuk mengelompokkan data ke dalam kelompok yang lebih terstruktur. Tahap ini merupakan bagian penting dalam proses data mining, karena tidak semua data atau atribut yang tersedia akan digunakan dalam analisis yang kami lakukan. Tujuan utama dari tahap persiapan data ini adalah untuk memastikan bahwa data yang kami gunakan sesuai dengan kebutuhan analisis yang kami lakukan, serta mempersiapkan data agar siap digunakan pada tahap selanjutnya.

2.4 Split Data Training dan Data Testing

Pada tahap ini, dataset awal akan dipisahkan menjadi dua bagian, yaitu dataset pelatihan (*training*) dan dataset pengujian (*testing*). Dataset pelatihan akan berisi data yang lengkap dengan atribut prediktor dan kelas, dan akan digunakan untuk melatih model agar dapat melakukan klasifikasi dengan akurasi yang tinggi sesuai dengan kelas yang sesuai. Sementara itu, dataset pengujian akan berisi data baru yang digunakan untuk menguji model yang telah dibangun dan untuk mengevaluasi performa model dalam melakukan klasifikasi. Pembagian dataset menjadi data pelatihan dan data pengujian dilakukan secara proposional, dengan umumnya menggunakan perbandingan 80% untuk dataset pelatihan dan 20% untuk dataset pengujian. Pembagian ini dilakukan untuk memastikan penggunaan yang seimbang antara kedua jenis data tersebut, dan tidak ada overlap atau tumpang tindih antara keduanya.

2.5 Training

Proses pelatihan dalam pembelajaran mesin melibatkan penggunaan algoritma yang mengubah parameter internalnya agar sesuai dengan data yang diberikan selama pelatihan. Analoginya mirip dengan otak manusia yang mengalami perubahan saat manusia belajar. Pelatihan dalam pembelajaran mesin sangat penting untuk memastikan bahwa model yang dihasilkan memiliki kinerja yang baik. Dalam pembelajaran mesin, model akan belajar atau melatih diri dari data yang diberikan untuk memahami informasi yang terkandung dalam data tersebut.

2.6 Neural Network

Neural Network adalah sistem yang digunakan untuk memproses informasi dan dianalogikan sebagai suatu bentuk kotak yang menerima masukan dan menghasilkan keluaran. *Neural Network* terdiri dari beberapa komponen pemrosesan dan bobot yang saling terhubung satu sama lain [14]. Metode-metode algoritma jaringan saraf untuk pembelajaran mesin terinspirasi oleh struktur dan pergerakan jaringan neuron dalam otak. Algoritma-algoritma ini menggunakan model-model neuron yang sangat idealis. Namun, prinsip dasarnya tetap sama: jaringan saraf buatan belajar dengan mengubah koneksi antara neuron-neuronnya. Jaringan semacam ini memiliki kemampuan untuk menjalankan berbagai tugas pemrosesan informasi yang beragam [15]. *Convolutional Neural Network* (CNN)

merupakan jenis dari penerapan dari *Neural Network* yang termasuk dalam kategori *Deep Neural Network* karena memiliki struktur dengan tingkat kedalaman yang tinggi [16]. CNN adalah metode yang dapat digunakan untuk berbagai aplikasi, termasuk melakukan identifikasi pengenalan bentuk wajah, klasifikasi dokumen, dan sebagainya [17]. Dalam algoritma ini, terdapat dua metode yang digunakan, yaitu *backpropagation* untuk pembelajaran dan *feedforward* untuk melakukan klasifikasi. CNN dirancang khusus untuk memproses data dengan struktur grid, seperti data deret waktu dalam bentuk grid 1D atau data gambar dalam bentuk grid piksel 2D. Bagian-bagian yang terdapat dalam CNN mencakup lapisan masukan (*input layer*), lapisan keluaran (*output layer*), dan beberapa lapisan yang tersembunyi. Lapisan tersembunyi atau disebut *hidden layer* ini meliputi lapisan konvolusi, lapisan penggabungan (*pooling layer*), lapisan normalisasi, lapisan ReLU, lapisan *fully connected*, dan lapisan *loss*[18].

2.7 K-Nearest Neighbor

Algoritma *K-Nearest Neighbor* (KNN) merupakan algoritma untuk melakukan klasifikasi yang memanfaatkan perhitungan jarak antara data uji dan data pelatihan. KNN bekerja dengan mencari k data pelatihan yang memiliki jarak terdekat dengan data uji, dan kemudian melakukan klasifikasi berdasarkan mayoritas kelas dari data terdekat tersebut. Dalam penghitungan jarak, digunakan rumus jarak seperti jarak *Euclidean* dan jarak *Minkowski*. Umumnya, rumus jarak *Euclidean* banyak digunakan karena memiliki tingkat akurasi yang tinggi [19]. KNN termasuk dalam algoritma klasifikasi sederhana dalam bidang pembelajaran mesin. Algoritma ini menggunakan pendekatan *supervised*, di mana sampel uji yang baru akan diklasifikasikan berdasarkan kategori yang paling banyak muncul dari tetangga terdekat dalam k-NN. Keakuratan algoritma KNN dipengaruhi oleh ketersediaan data yang relevan dan pengukuran bobot fitur yang sesuai dengan klasifikasi yang dilakukan [20]. Dalam KNN, data latihan beserta kelasnya disimpan sebagai aturan. Ketika ada data masukan baru, algoritma ini akan mengklasifikasikan objek baru berdasarkan nilai atributnya dan data latihan yang ada. Tujuannya adalah melakukan klasifikasi objek baru berdasarkan atribut-nilai dan informasi yang terdapat pada data latihan [21].

2.8 Evaluasi Kinerja Sistem

Tujuan evaluasi sistem secara keseluruhan adalah untuk menguji kemampuan sistem dalam mendeteksi *malware* pada platform Android menggunakan teknologi *machine learning* dan bahasa pemrograman Python. Evaluasi ini bertujuan untuk memverifikasi apakah program berjalan sesuai dengan perancangan yang telah dirancang dan untuk mengukur tingkat kesalahan program yang mungkin terjadi. Dalam penelitian ini, kami melakukan penghitungan akurasi, presisi, dan *F1-score* pada setiap metode klasifikasi.

Tabel 1. *Confusion Matrix*

	FALSE POSITIVE (FP)	FALSE NEGATIVE (FN)
TRUE POSITIVE	TP	FN
TRUE NEGATIVE	FP	TN

Confusion matrix adalah komponen dasar untuk menemukan akurasi, presisi, dan *recall*. Akurasi merupakan perbandingan dari prediksi yang benar yang diklasifikasikan oleh sistem. Rumus dasar untuk akurasi terdapat pada persamaan 1.

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (1)$$

Presisi digunakan untuk membandingkan prediksi benar positif dengan hasil prediksi positif secara keseluruhan. Persamaan berikutnya, yaitu persamaan 2, merupakan rumus untuk Presisi.

$$\text{Presisi} = \frac{TP}{TP+FP} \times 100\% \quad (2)$$

Recall dihitung untuk menemukan hubungan antara prediksi benar positif dan seluruh data positif yang sebenarnya. Persamaan 3 merupakan rumus dasar untuk Recall.

$$\text{Recall} = \frac{TP}{TP+FN} \times 100\% \quad (3)$$

Skor F1 merupakan nilai yang dihasilkan dari *harmonic mean* antara presisi dan *recall*. Nilai Skor F1 dihitung menggunakan persamaan 4.

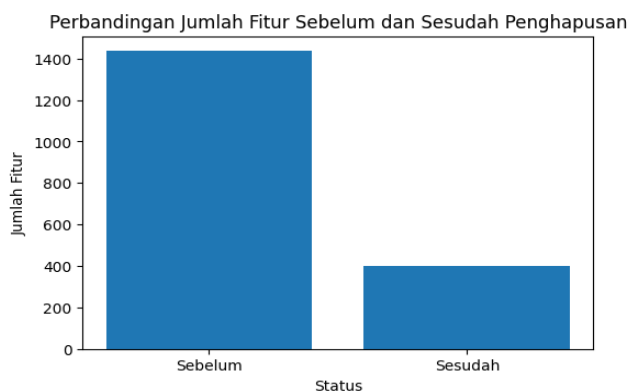
$$F1 - score = \frac{1}{2} \left(\frac{1}{Precision} + \frac{1}{Recall} \right) \quad (4)$$

3. HASIL DAN PEMBAHASAN

3.1 Hasil Pengolahan Data

Dalam penelitian ini, saya menggunakan sebuah dataset yang awalnya terdiri dari 100 ribu aplikasi dan 1436 fitur *permission*. Namun, setelah melakukan analisis awal, saya menemukan bahwa sebagian besar aplikasi memiliki fitur-

fitur dengan kolom kosong atau tidak memiliki informasi yang relevan. Untuk memastikan kualitas dan validitas data yang digunakan dalam analisis, saya melakukan pemilihan subset dari dataset awal tersebut. Dalam proses ini, saya memutuskan untuk menggunakan 30 ribu aplikasi yang memiliki data yang lebih lengkap dan informatif dibandingkan dengan aplikasi lainnya. Pemilihan subset ini dilakukan dengan hati-hati untuk memastikan representativitas data. Saya mempertimbangkan faktor-faktor seperti keberagaman aplikasi, ukuran populasi, dan relevansi dengan tujuan penelitian. Selain itu, saya juga berusaha untuk menghindari terjadinya bias yang signifikan dalam analisis dengan pemilihan subset. Setelah berhasil memilih subset yang terdiri dari 30 ribu aplikasi dengan fitur-fitur yang memiliki data yang lengkap, saya melanjutkan dengan melakukan pembersihan data. Langkah ini melibatkan penghapusan fitur-fitur yang memiliki kolom kosong pada semua aplikasi dalam subset tersebut. Tujuan dari pembersihan data ini adalah untuk memastikan bahwa hanya fitur-fitur dengan data yang lengkap yang digunakan dalam analisis selanjutnya. Dengan menggunakan subset yang lebih kecil namun memiliki fitur-fitur yang informatif dan lengkap, saya dapat melanjutkan analisis lebih lanjut dalam penelitian ini. Dataset yang telah disusun dengan teliti ini memberikan dasar yang solid untuk mendapatkan wawasan dan data yang relevan dengan tujuan penelitian.



Gambar 2. Grafik Hasil Pengolahan Data Pengurangan Fitur

3.2 Hasil Pengujian Neural Network

Pada penelitian ini, dilakukan penggunaan model CNN 1D untuk melakukan klasifikasi data. Dataset awal dibagi menjadi data latih dan data uji dengan rasio 80:20. Data latih dan data uji kemudian diubah bentuknya agar sesuai dengan input yang dibutuhkan oleh model CNN 1D. Data latih dan data uji di-expand dimensi menjadi data tiga dimensi dengan menambahkan satu dimensi yang merepresentasikan kanal. Model CNN 1D yang digunakan terdiri dari beberapa layer konvolusi dan maxpooling. Conv1D digunakan untuk melakukan konvolusi satu dimensi pada data input. MaxPooling1D digunakan untuk melakukan operasi max pooling pada output dari layer konvolusi. Kemudian dilanjutkan dengan beberapa layer konvolusi dan max pooling lagi untuk melakukan ekstraksi fitur secara bertingkat. Setelah itu, output dari layer max pooling terakhir di-flatten untuk dihubungkan ke beberapa layer fully connected dengan fungsi aktivasi ReLU. Akhirnya, layer output dengan fungsi aktivasi sigmoid digunakan untuk menghasilkan prediksi. Model tersebut dikompilasi dengan optimiser Adam dan learning rate yang lebih kecil. Fungsi loss yang digunakan adalah binary crossentropy, karena tugas klasifikasi yang dilakukan adalah biner (dua kelas). Selain itu, metrik akurasi juga dihitung selama pelatihan. Model dilatih dengan data latih dalam beberapa epoch. Setiap epoch, data latih diambil dalam batch-batch dengan ukuran tertentu. Selama pelatihan, dilakukan validasi menggunakan data uji untuk melihat performa dari model pada data yang tidak digunakan dalam proses pelatihan. Setelah pelatihan selesai, model dievaluasi menggunakan data uji untuk mengukur loss dan akurasi yang dihasilkan oleh model. Selanjutnya, dilakukan prediksi pada data uji menggunakan model yang telah dilatih. Hasil prediksi tersebut kemudian digunakan untuk membuat laporan klasifikasi (classification report) yang mencakup metrik-metrik seperti presisi, recall, dan f1-score untuk setiap kelas.

Tabel 2. Classification Report Neural Network

	Precision	Recall	F1-Score	Accuracy
Weighted Avg	97%	97%	97%	97%

3.3 Hasil Pengujian K- Nearest Neighbor

Kami membangun model K- Nearest Neighbors (KNN) dengan melakukan optimasi parameter menggunakan GridSearchCV. GridSearchCV digunakan untuk mencari parameter terbaik dalam algoritma KNN yang dapat meningkatkan performa model. Parameter yang kami optimalkan dalam GridSearchCV mencakup jumlah tetangga (n_neighbors) dan metrik jarak (metric) yang digunakan dalam perhitungan jarak antara tetangga. Kami melakukan penelusuran grid dengan validasi silang (cross-validation) sebanyak 5 kali dan menggunakan metrik akurasi sebagai skoring. Setelah melakukan optimasi parameter, kami menampilkan parameter terbaik yang ditemukan oleh GridSearchCV. Kemudian, kami membangun model KNN baru dengan menggunakan parameter terbaik tersebut.

Model ini kemudian dilatih menggunakan data pelatihan. Setelah melatih model, kami menggunakan model tersebut untuk memprediksi label dari data pengujian. Hasil prediksi tersebut dibandingkan dengan label sebenarnya pada data pengujian. Kami menggunakan metrik evaluasi klasifikasi seperti akurasi, presisi, *recall*, dan *F1-score* untuk mengevaluasi performa model kami. Hasil evaluasi model kami ditampilkan dalam bentuk classification report yang mencakup nilai akurasi, presisi, *recall*, dan *F1-score* untuk setiap kelas, serta metrik evaluasi keseluruhan. Informasi ini membantu kami dalam mengevaluasi sejauh mana model KNN yang telah dioptimasi bekerja dengan baik dalam melakukan klasifikasi pada dataset yang digunakan.

Tabel 3. Classification Report K- Nearest Neighbor

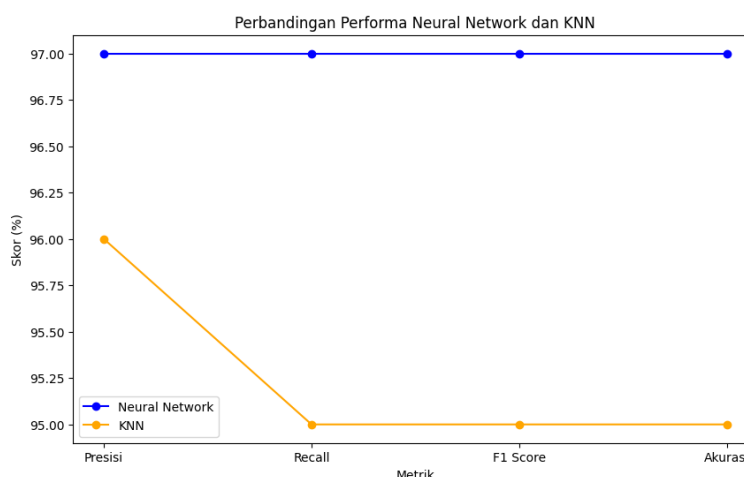
	Precision	Recall	F1- Score	Accuracy
Weighted Avg	96%	95%	95%	95%

3.4 Perbandingan Hasil Pengujian

Setelah menyelesaikan pengujian dan mendapatkan laporan klasifikasi, langkah berikutnya adalah membandingkan hasil dari kedua laporan tersebut. Hasil perbandingan ini akan disajikan dalam Tabel 4 dan Gambar 3. Tabel 2 akan menampilkan perbandingan metrik kinerja utama antara dua algoritma, yaitu Neural Network dan K-Nearest Neighbors (KNN), dalam deteksi *malware*. Metrik kinerja yang akan dibandingkan meliputi akurasi, presisi, recall, dan F1-score. Dengan melihat perbandingan ini, kita dapat mengevaluasi kekuatan dan kelemahan masing-masing algoritma dalam menangani masalah deteksi *malware* pada dataset yang digunakan. Gambar 2 dan 3 akan memvisualisasikan hasil perbandingan dengan cara yang lebih jelas. Grafik ini akan menampilkan perbedaan kinerja antara Neural Network dan KNN secara visual yang mudah dipahami. Dengan menggunakan grafik ini, kita dapat melihat pola dan tren dari metrik kinerja yang dibandingkan, serta mendapatkan pemahaman yang lebih mendalam tentang perbedaan antara kedua algoritma tersebut dalam deteksi *malware*. Melalui perbandingan ini, kita akan mendapatkan pengetahuan lebih baik tentang kemampuan kedua algoritma dalam menyelesaikan masalah deteksi *malware* pada dataset yang diberikan. Selanjutnya, informasi ini dapat digunakan untuk memilih algoritma yang paling sesuai dalam situasi yang berbeda, sesuai dengan kebutuhan dan persyaratan spesifik yang dihadapi dalam deteksi dan penghapusan *malware*.

Tabel 4. Hasil Perbandingan Classification Report

	Precision	Recall	F1- Score	Accuracy
Neural Network	97%	97%	97%	97%
K- Nearest Neighbor	96%	95%	95%	95%



Gambar 3. Grafik Perbandingan Algoritma Neural Network dan KNN

Dalam penelitian ini, peneliti melakukan pengujian untuk mendeteksi Android *malware* menggunakan dua algoritma *machine learning* yang berbeda, *Neural Network* dan *K-Nearest Neighbors* (KNN). Peneliti ingin mengetahui perbandingan kinerja kedua algoritma ini dalam mengidentifikasi dan mengklasifikasikan *malware* pada sistem Android. Hasil pengujian kami menunjukkan perbedaan signifikan antara kedua algoritma tersebut. Algoritma *Neural Network* menunjukkan performa yang sangat baik dalam deteksi Android *malware*, dengan mencapai akurasi sebesar 97%, presisi 97%, *recall* 97%, dan *F1-score* 97%. Hasil ini menunjukkan bahwa model *Neural Network* mampu memberikan hasil klasifikasi yang sangat akurat dan konsisten dalam mengidentifikasi *malware*. Sementara itu, kami juga menguji algoritma *K-Nearest Neighbors* (KNN) untuk deteksi Android *malware*. Meskipun KNN masih memberikan performa yang baik, terdapat sedikit penurunan dibandingkan dengan *Neural Network*. Algoritma KNN menghasilkan akurasi sebesar 95%, presisi 95%, *recall* 95%, dan *F1-score* 95%. Berdasarkan hasil pengujian, baik *Neural Network* maupun KNN memiliki kemampuan dalam mendeteksi Android *malware*. Namun, *Neural Network*

secara konsisten unggul dalam hal akurasi, presisi, *recall*, dan *F1-score*. Dalam perbandingan hasil penelitian yang peneliti lakukan dengan penelitian yang dilakukan oleh Hadiprakoso, Aditya, dan Pramitha (2022), hasil penelitian yang peneliti lakukan menunjukkan performa yang lebih tinggi dalam deteksi Android malware menggunakan algoritma Neural Network (NN). Penelitian ini mencapai akurasi sebesar 97% dengan *Neural Network*, sementara penelitian yang dilakukan oleh Hadiprakoso, Aditya, dan Pramitha menggunakan model *klasifikasi Support Vector Machine* (SVM) dan mencapai akurasi sebesar 96,94%. Oleh karena itu, jika akurasi dan konsistensi klasifikasi menjadi faktor utama dalam deteksi Android *malware*, *Neural Network* mungkin merupakan pilihan yang lebih cocok untuk digunakan. Penelitian ini memberikan wawasan yang berharga dalam penggunaan algoritma *machine learning* untuk mendeteksi Android *malware*. Penerapan teknologi ini, diharapkan dapat meningkatkan keamanan sistem Android dan melindungi pengguna dari ancaman *malware* yang ada.

4. KESIMPULAN

Penelitian ini membandingkan performa dua algoritma machine learning, yaitu *Neural Network* (NN) dan *K-Nearest Neighbors* (KNN), dalam mendeteksi malware pada platform Android. Hasil penelitian menunjukkan bahwa *Neural Network* (NN) memiliki performa yang lebih baik daripada *K-Nearest Neighbors* (KNN). Model *Neural Network* mencapai akurasi, presisi, *recall*, dan *F1-score* sebesar 97%, menunjukkan kemampuan yang sangat baik dalam mengklasifikasikan aplikasi Android menjadi *benign* atau *malware*. Sementara itu, *K-Nearest Neighbors* (KNN) menghasilkan akurasi, presisi, *recall*, dan *F1-score* sebesar 95%, menunjukkan performa yang sedikit lebih rendah dibandingkan dengan *Neural Network*. Saran untuk penelitian selanjutnya termasuk melibatkan algoritma *machine learning* lainnya yang belum pernah digunakan dalam penelitian deteksi malware pada platform Android, menggunakan dataset yang lebih luas dan representatif, serta mempertimbangkan fitur-fitur tambahan yang berkaitan dengan perilaku aplikasi Android atau analisis struktur kode. Selain itu, optimisasi parameter untuk masing-masing algoritma dapat dilakukan guna meningkatkan performa klasifikasi. Dengan mengikuti saran-saran ini, penelitian selanjutnya di bidang deteksi malware pada platform Android dapat memberikan kontribusi lebih lanjut dalam pengembangan solusi yang lebih akurat dan efektif untuk mengidentifikasi dan membedakan jenis aplikasi Android yang berpotensi berbahaya.

REFERENCES

- [1] R. B. Hadiprakoso, W. R. Aditya, F. N. Pramitha, and P. Siber, "Analisis Statis Deteksi Malware Android Menggunakan Algoritma Supervised Machine Learning," vol. 5, no. 1, pp. 1–5, 2022.
- [2] S. Mardiyati and L. Ariyani, "Perancangan Aplikasi Security Lock Berbasis Android," *JUST IT J. Sist. Informasi, Teknol. Inf. dan Komput.*, vol. 11, no. 2, p. 18, 2021, doi: 10.24853/justit.11.2.18-24.
- [3] B. Prasetyo, V. Suryani, and D. R. Anbiya, "Analisis Deteksi Malware pada Aplikasi Android Fintech berdasarkan Permissions dengan menggunakan Naive Bayes dan Random Forest," vol. 8, no. 5, pp. 9885–9897, 2021.
- [4] CNN INDONESIA, "BSSN Ungkap Ransomware Dominasi Serangan Siber di Indonesia," 2023. <https://www.cnnindonesia.com/teknologi/20230220152846-192-915441/bssn-ungkap-ransomware-dominasi-serangan-siber-di-indonesia> (accessed Apr. 09, 2023).
- [5] C. Chazar and B. Erawan, "Machine Learning Diagnosis Kanker Payudara Menggunakan Algoritma Support Vector Machine," *Inf. (Jurnal Inform. dan Sist. Informasi)*, vol. 12, no. 1, pp. 67–80, 2020, doi: 10.37424/informasi.v12i1.48.
- [6] A. Roihan, P. A. Sunarya, and A. S. Rafika, "Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper," *IJCIT (Indonesian J. Comput. Inf. Technol.)*, vol. 5, no. 1, pp. 75–82, 2020, doi: 10.31294/ijcit.v5i1.7951.
- [7] R. K. Mariette Awad, *Efficient Learning Machines*. Apress Berkeley, CA, 2015.
- [8] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Comput. Secur.*, vol. 89, 2020, doi: 10.1016/j.cose.2019.101663.
- [9] D. Diana, R. E. Indrajit, and E. Dazki, "Komparasi Algoritma Naïve Bayes, Logistic Regression Dan Support Vector Machine pada Klasifikasi File Application Package Kit Android Malware," *Jutisi J. Ilm. Tek. Inform. dan Sist. Inf.*, vol. 11, no. 1, p. 109, 2022, doi: 10.35889/jutisi.v11i1.815.
- [10] R. B. Hadiprakoso, N. Qomariasih, and R. N. Yasa, "Identifikasi Malware Android Menggunakan Pendekatan Analisis Hibrid Dengan Deep Learning," *J. Teknol. Inf. Univ. Lambung Mangkurat*, vol. 6, no. 2, pp. 77–84, 2021, doi: 10.20527/jtiulm.v6i2.82.
- [11] Y. W. Sitorus, P. Sukarno, and S. Mandala, "Analisis Deteksi Malware Android menggunakan metode Support Vector Machine & Random Forest," *e-Proceeding Eng.*, vol. 8, no. 6, pp. 12500–12518, 2021.
- [12] Nursobah, S. Lailiyah, B. Harpad, and M. Fahmi, "Penerapan Data Mining Untuk Prediksi Perkiraan Hujan dengan Menggunakan Algoritma K-Nearest Neighbor," *Build. Informatics, Technol. Sci.*, vol. 4, no. 3, pp. 1395–1400, 2022, doi: 10.47065/bits.v4i3.2564.
- [13] A. Mahindru, "Android permissions dataset," *Data Mendeley*, 2020. <https://data.mendeley.com/datasets/b4mxg7ydb7/2> (accessed Apr. 04, 2023).
- [14] N. Hadianto, H. B. Novitasari, and A. Rahmawati, "Klasifikasi Peminjaman Nasabah Bank Menggunakan Metode Neural Network," *J. Pilar Nusa Mandiri*, vol. 15, no. 2, pp. 163–170, 2019, doi: 10.33480/pilar.v15i2.658.
- [15] B. Mehlig, *Machine Learning with Neural Networks*. <https://arxiv.org/abs/1901.05639>, 2021.
- [16] P. A. Nugroho, I. Fenriana, and R. Arijanto, "Implementasi Deep Learning Menggunakan Convolutional Neural Network (Cnn) Pada Ekspresi Manusia," *Algor*, vol. 2, no. 1, pp. 12–21, 2020.
- [17] A. Antoni, T. Rohana, and A. R. Pratama, "Implementasi Algoritma Convolutional Neural Network Untuk Klasifikasi Citra



- Kemasan Kardus Defect dan No Defect,” vol. 4, no. 4, pp. 1941–1950, 2023, doi: 10.47065/bits.v4i4.3270.
- [18] F. P. Rachman, “Perbandingan Model Deep Learning untuk Klasifikasi Sentiment Analysis dengan Teknik Natural Language Processing,” *J. Teknol. dan Manaj. Inform.*, vol. 7, no. 2, pp. 113–121, 2021, doi: 10.26905/jtmi.v7i2.6506.
- [19] V. Alvian, D. Hidayatullah, A. Nilogiri, H. Azizah, and A. Faruq, “Klasifikasi Siswa Berprestasi Menggunakan Metode K-Nearest Neighbor (KNN) Pada SMA Negeri 2 Situbondo Classification Of Achieving Students Using K-Nearest Neighbor (KNN) Method At SMA Negeri 2 Situbondo,” *J. Smart Teknol.*, vol. 3, no. 6, pp. 2774–1702, 2022, [Online]. Available: <http://jurnal.unmuhjember.ac.id/index.php/JST>.
- [20] A. R. Yogaswara, “Klasifikasi Malware Family menggunakan Metode k-Nearest Neighbor (k-NN),” *J. Repos.*, vol. 3, no. 3, pp. 319–323, 2021, doi: 10.22219/repositor.v2i3.1313.
- [21] P. Putra, A. M. H. Pardede, and S. Syahputra, “Analisis Metode K-Nearest Neighbour (Knn) Dalam Klasifikasi Data Iris Bunga,” *J. Tek. Inform. Kaputama*, vol. 6, no. 1, pp. 297–305, 2022.