Journal of Applied Informatics and Computing (JAIC)
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861

Evaluasi Performa TabTransformer Terhadap XGBoost dalam Klasifikasi Malware Android pada Dataset MH-1M

Ghiffar Alfin Faiz [1]*, Hamdani Arif [2]**
* Teknik Informatika, Politeknik Negeri Batam
** Rekayasa Keamanan Siber, Politeknik Negeri Batam
ghiffar.4332201010@students.polibatam.ac.id [1], hamdaniarif@polibatam.ac.id [2]

Article Info

Article history:

Received …
Revised …
Accepted …

Keyword:

Android Malware Detection, Artificial Intelligence, Macchine Learning, Deep Learning, MH-1M, TabTransformer, XGBoost

ABSTRACT

Pertumbuhan eksponensial ekosistem Android mendorong evolusi malware seluler yang semakin canggih, sehingga menuntut mekanisme deteksi berbasis perilaku yang andal. Arsitektur *deep learning* tabular seperti TabTransformer memiliki potensi teoretis yang kuat dalam memodelkan interaksi fitur kompleks melalui *contextual embeddings*, namun efektivitasnya masih belum banyak dieksplorasi dalam domain keamanan siber berskala masif. Penelitian ini bertujuan untuk mengevaluasi performa TabTransformer dengan menempatkan algoritma *gradient boosting* XGBoost sebagai model pembanding utama pada dataset MH-1M. Hasil eksperimen menunjukkan bahwa meskipun TabTransformer mencatatkan performa yang sangat kompetitif dengan akurasi 96,66% dan nilai ROC-AUC 0,9907, XGBoost secara konsisten unggul tipis di seluruh metrik prediktif dengan akurasi 97,21% dan ROC-AUC 0,9948. Secara kuantitatif, XGBoost mengungguli TabTransformer pada rentang selisih skor 0,42 hingga 0,80 poin persentase, termasuk keunggulan 0,55% pada akurasi dan 0,60% pada F1-Score. Selisih paling signifikan ditemukan pada *False Positive Rate* (FPR), di mana TabTransformer mencatatkan FPR 0,54% lebih tinggi (2,05%) dibandingkan XGBoost (1,51%). Kesenjangan skor ini mengindikasikan bahwa karakteristik dataset MH-1M yang didominasi fitur biner dan *sparse* lebih sesuai dengan mekanisme pemisahan langsung pohon keputusan XGBoost, sehingga membuat keunggulan interaksi kontekstual dari mekanisme *self-attention* pada TabTransformer tidak dapat dimanfaatkan secara maksimal. Penelitian ini menyimpulkan bahwa untuk data tabular biner berskala masif, arsitektur *gradient boosting* tetap menjadi pilihan yang lebih optimal.

I. PENDAHULUAN

Sejak pertama kali diluncurkan pada tahun 2008, ekosistem Android telah tumbuh secara eksponensial hingga menguasai 86,6% pangsa pasar *smartphone* global pada tahun 2019 dan menampung lebih dari 2,8 juta aplikasi di Google Play Store pada tahun 2020 [1]. Dominasi masif ini secara langsung memperluas permukaan serangan bagi *malware* yang semakin canggih, sehingga mendesak pergeseran mekanisme deteksi menuju paradigma Kecerdasan Buatan (AI) yang lebih adaptif [2]. Ancaman modern seperti *ransomware* dan *banking trojan* kini rutin menggunakan teknik *obfuscation* untuk menghindari deteksi statis, yang mendorong adopsi metode berbasis perilaku (*behavior-based*) menggunakan *Machine Learning* (ML) dan *Deep Learning* (DL). Berbagai studi terdahulu telah mengimplementasikan algoritma ML seperti *Random Forest* dan *XGBoost*, serta arsitektur DL seperti *Convolutional Neural Network* (CNN) dan *Recurrent Neural Network* (RNN) [3], [4], [5], [6], [7], [8], [9]. Meskipun menjanjikan, mayoritas penelitian tersebut masih terbatas pada dataset berskala kecil dan usang seperti Drebin atau AMD yang tidak mampu merepresentasikan evolusi temporal dan kompleksitas *malware* masa kini. Sebagai respons terhadap keterbatasan ekologis ini, salah satu penelitian memperkenalkan dataset MH-1M yang memuat 1,34 juta APK dengan lebih dari 12 ribu atribut multidimensi sepanjang 14 tahun (2010–2024), menjadikannya *benchmark* yang jauh lebih komprehensif dan representative [10]. Sayangnya, pemanfaatan dataset komprehensif ini sejauh ini masih terbatas pada evaluasi algoritma *machine learning* konvensional, sehingga potensi penerapan arsitektur *deep learning* tabular mutakhir seperti TabTransformer yang secara empiris sangat handal dalam memodelkan interaksi fitur kompleks melalui *contextual embeddings* belum dieksplorasi secara optimal untuk mendeteksi *malware* Android berskala massif [11], [12], [13].

Berdasarkan beberapa artikel tersebut, terdapat kesenjangan penelitian (*research gap*) yang signifikan karena studi yang memanfaatkan *benchmark* MH-1M sejauh ini hanya berfokus pada evaluasi performa untuk satu algoritma *machine learning* standar, yaitu XGBoost, tanpa menyertakan perbandingan dengan arsitektur *deep learning* berbasis mekanisme *self-attention*. Oleh karena itu, penelitian ini bertujuan untuk mengisi celah tersebut dengan mengevaluasi dan membandingkan secara langsung performa arsitektur *Deep Learning* TabTransformer terhadap algoritma *Machine Learning* XGBoost dalam mengklasifikasikan *malware* Android pada kondisi distribusi kelas yang sangat tidak seimbang. Melalui penerapan rangkaian metrik evaluasi yang meliputi *accuracy, precision, recall, F1-score, ROC-AUC*, dan *False Positive Rate* (FPR).

II. METODE

A. Desain dan Prosedur Penelitian

Penelitian ini menggunakan pendekatan eksperimental kuantitatif berbasis komputasi untuk mengevaluasi dan membandingkan performa arsitektur *deep learning* TabTransformer dengan algoritma *machine learning* XGBoost dalam mendeteksi *malware* Android. Pendekatan eksperimental dipilih untuk menguji hipotesis secara terstruktur melalui manipulasi variabel dan pengamatan dampaknya terhadap metrik klasifikasi. Prosedur penelitian ini dirancang secara sistematis dan berurutan, mencakup tiga tahapan utama: praproses data, pelatihan model, dan evaluasi. Pada tahap praproses data, dilakukan penanganan ketidakseimbangan kelas melalui teknik *undersampling* untuk mengubah rasio kelas dari 1:10 menjadi 1:2. Selanjutnya, pada tahap pelatihan model, dataset yang telah diproses dibagi dengan rasio 60% untuk data latih (*training*), 20% untuk data uji (*testing*), dan 20% untuk data validasi (*validation*) lalu dilatih secara terpisah menggunakan algoritma XGBoost dan TabTransformer. Terakhir, tahap evaluasi dilakukan untuk mengukur dan membandingkan kinerja kedua model menggunakan berbagai metrik yang komprehensif, meliputi *accuracy*, *precision*, *recall*, *F1-score*, *confusion matrix*, *ROC-AUC*, dan *False Positive Rate* (FPR). Rancangan ini diharapkan dapat memberikan gambaran empiris

e-ISSN: 2548-6861

yang objektif mengenai keunggulan dan keterbatasan masing-masing pendekatan dalam menangani dataset berskala besar.

B. Dataset Penelitian

Dataset yang digunakan dalam penelitian ini adalah MH-1M, sebuah dataset *malware* Android berskala besar yang terdiri dari 1.340.515 sampel APK dengan ribuan atribut statis. Dataset ini dikembangkan dari repositori AndroZoo dan mencakup rentang waktu empat belas tahun (2010–2024), sehingga mampu merepresentasikan evolusi *malware* Android secara komprehensif dan relevan dengan ancaman masa kini. Secara struktural, MH-1M merupakan dataset tabular dengan representasi fitur biner, di mana setiap atribut menunjukkan ada atau tidaknya suatu karakteristik statis tertentu, seperti pemanggilan API, *intent*, *opcode*, atau *permission*, pada suatu APK. Nilai 1 diberikan apabila karakteristik tersebut ditemukan diterapkan pada aplikasi, sedangkan nilai 0 diberikan apabila karakteristik tersebut tidak ditemukan. Dengan representasi ini, setiap sampel aplikasi dapat digambarkan sebagai kumpulan tanda ada atau tidaknya ribuan karakteristik statis sekaligus, sehingga pola perilaku aplikasi dapat dikenali cukup melalui data statis tanpa memerlukan analisis dinamis yang lebih kompleks dan mahal secara komputasi.

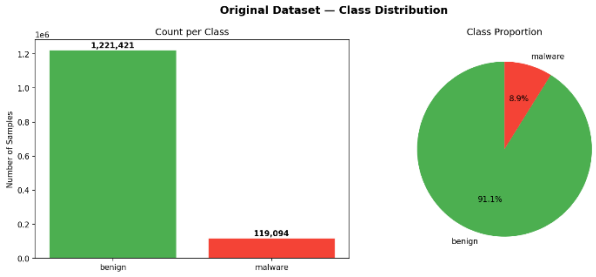
Dibandingkan dengan dataset *malware* Android yang lebih dahulu ada, seperti Drebin yang dirilis pada tahun 2014 dengan sekitar satu juta sampel, atau MH-100K yang dirilis pada akhir tahun 2023 dengan sekitar seratus ribu sampel namun minim metadata dan cakupan temporal, MH-1M unggul baik dari segi skala sampel, kekayaan metadata, maupun rentang waktu pengumpulan data [10]. Setiap sampel pada MH-1M juga dilengkapi hasil klasifikasi dari VirusTotal API yang mengintegrasikan puluhan mesin deteksi, sehingga label *malware* maupun *benign* yang dihasilkan lebih akurat dan andal. Dataset MH-1M dapat diakses secara publik melalui platform GitHub dan Figshare [14], [15], yang menyediakan beberapa varian dataset dengan komposisi dan jumlah fitur yang berbeda-beda untuk mendukung transparansi dan reproduktibilitas penelitian. Sebagai contoh, terdapat varian dataset yang hanya berfokus pada fitur *intents* secara eksklusif, serta varian mentah yang belum melalui proses seleksi fitur.

Dalam penelitian ini, digunakan varian dataset yang telah diseleksi menggunakan uji Chi-square dan terdiri dari 12.409 fitur paling relevan. Fitur-fitur tersebut secara spesifik mencakup 11.556 *API calls* yang merepresentasikan pemanggilan fungsi-fungsi sistem oleh aplikasi, 407 *intents* yang menggambarkan mekanisme komunikasi antar komponen aplikasi, 232 *opcodes* yang mencerminkan instruksi tingkat rendah pada *bytecode* aplikasi, serta 214 *permissions* yang menunjukkan hak akses yang diminta aplikasi terhadap sumber daya perangkat, seperti kamera, lokasi, kontak, dan penyimpanan.

Penggunaan varian yang telah terseleksi ini sangat krusial bagi penelitian ini karena kelengkapan metadata, seperti informasi *hash*, nama paket, dan laporan VirusTotal, serta rasio sebaran kelas yang merepresentasikan kondisi ekosistem nyata, menjadikannya *benchmark* modern yang ideal sekaligus efisien untuk pengujian model deteksi *malware*. Selain itu, hasil seleksi fitur menggunakan uji Chi-square memastikan bahwa fitur yang digunakan memiliki tingkat independensi dan relevansi statistik yang tinggi terhadap label kelas, sehingga dapat mengurangi *noise* serta beban komputasi tanpa mengorbankan informasi penting untuk proses klasifikasi.

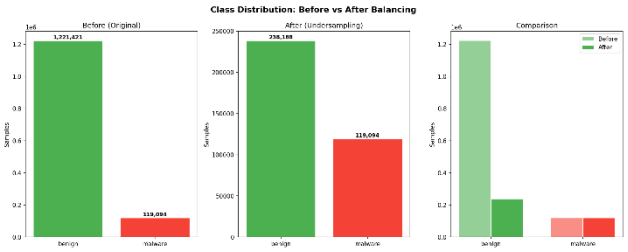
C. Praproses Data (Data Preprocessing)

Tahap praproses data merupakan langkah krusial untuk memastikan kualitas dataset sebelum masuk ke tahap pemodelan. Karakteristik asli dataset MH-1M memiliki ketidakseimbangan kelas yang sangat signifikan dengan rasio 1:10, yang terdiri dari sekitar 1.221.421 sampel aplikasi *benign* dan 119.094 sampel *malware*.



Gambar 1.Proporsi dan persentase sampel

Ketidakseimbangan ini dapat menyebabkan model menjadi bias terhadap kelas mayoritas (*benign*) dan mengabaikan pola pada kelas minoritas (*malware*). Oleh karena itu, penelitian ini menerapkan teknik *undersampling* pada kelas mayoritas untuk menyeimbangkan distribusi data. Rasio kelas disesuaikan menjadi 1:2 dengan mempertahankan seluruh 119.094 sampel *malware* dan mengambil sampel acak sebanyak 238.188 dari kelas *benign*.



Gambar 2.Perbandingan Proporsi sample sebelum dan sesudah undersampling

Pemilihan rasio 1:2 ini didasarkan pada pertimbangan untuk mempertahankan representasi kelas mayoritas yang cukup agar model tetap memahami karakteristik aplikasi *benign*, serta mencegah model dari dominasi prediksi kelas mayoritas yang berujung pada rendahnya *recall* pada deteksi *malware*. Selain itu, proses praproses juga mencakup normalisasi fitur kontinu dan *encoding* untuk fitur kategorikal agar kompatibel dengan arsitektur model yang digunakan.

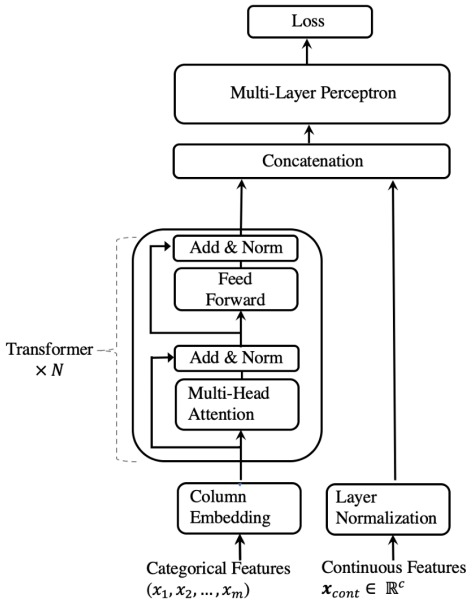
D. Pelatihan Model (Model Training)

Setelah melalui tahap praproses, dataset dibagi menjadi tiga subset dengan rasio 60% untuk data latih (training set), 20% untuk data validasi (validation set), dan 20% untuk data uji (test set). Penelitian ini melatih dua model utama, yaitu XGBoost dan TabTransformer, dengan konfigurasi yang dioptimalkan berdasarkan mekanisme arsitektur masing-masing.

XGBoost (Extreme Gradient Boosting) adalah salah satu algoritma machine learning berbasis ensemble learning yang sangat populer dan efektif, khususnya untuk data tabular. Algoritma ini bekerja dengan menggabungkan banyak decision tree yang dilatih secara berurutan. Setiap pohon baru dibangun untuk memperbaiki kesalahan dari pohon sebelumnya dengan memanfaatkan gradient dari fungsi loss [16]. Proses ini disebut boosting, di mana model secara bertahap menjadi lebih akurat karena setiap pohon berkontribusi dalam mengurangi error prediksi. Keunggulan utama XGBoost adalah kemampuannya menghasilkan akurasi tinggi, efisiensi komputasi, serta fleksibilitas dalam menangani berbagai jenis data dan masalah klasifikasi maupun regresi.

Untuk algoritma XGBoost dalam penelitian ini, proses pencarian hiperparameter terbaik dilakukan menggunakan framework Optuna yang dipilih karena menggunakan pendekatan optimasi Bayesian (Tree-structured Parzen Estimator) yang secara signifikan lebih efisien dan efektif dalam menjelajahi ruang pencarian hiperparameter yang luas dibandingkan metode Grid Search, sehingga mampu menemukan konfigurasi parameter yang menghasilkan generalisasi model terbaik tanpa membuang sumber daya komputasi secara percuma [17].

TabTransformer adalah sebuah arsitektur deep learning yang dirancang khusus untuk pemodelan data tabular, yaitu data yang tersusun dalam bentuk tabel dengan kombinasi fitur kategorikal dan kontinu [11]. Berbeda dengan pendekatan tradisional seperti Gradient Boosted Decision Trees (GBDT) atau Multi-Layer Perceptron (MLP), TabTransformer memanfaatkan mekanisme self-attention dari Transformer untuk mengubah embedding parametris dari fitur kategorikal menjadi contextual embeddings. Arsitektur TabTransformer terdiri dari tiga komponen utama: column embedding layer, stack of Transformer layers, dan multi-layer perceptron (MLP). Pertama, setiap fitur kategorikal diubah menjadi embedding parametris melalui column embedding. Embedding ini kemudian diproses oleh serangkaian Transformer layers yang menggunakan multi-head self-attention untuk menghitung hubungan antar fitur, sehingga menghasilkan contextual embeddings yang lebih bermakna. Selanjutnya, embedding kontekstual tersebut digabungkan dengan fitur kontinu dan diteruskan ke MLP untuk melakukan prediksi target. Proses ini dilakukan secara end-to-end dengan optimisasi menggunakan fungsi loss. Keunggulan utama TabTransformer adalah kemampuannya menghasilkan representasi yang lebih interpretable dibandingkan MLP, serta lebih robust terhadap data yang hilang atau terkontaminasi noise.



Gambar 3.Arsitektur TabTransformer

Tantangan utama pada TabTransformer terletak pada jumlah fitur yang sangat besar yang menyebabkan inefisiensi komputasi dan lonjakan penggunaan memori (Out-of-Memory) saat memproses contextual embeddings jika memperlakukan untuk seluruh 12.409 fitur. Untuk mengatasi hal tersebut, penulis melakukan seleksi fitur dengan mengambil Top K sebanyak 300 fitur paling penting menggunakan uji Chi-square. Uji Chi-square dipilih karena mampu mengevaluasi tingkat independensi dan ketergantungan statistik antara setiap fitur dengan variabel target, sehingga hanya fitur yang memiliki korelasi terkuat terhadap indikasi malware ataupun benign yang dipilih, dan sisa dari fitur tersebut akan dijadikan sebagai fitur kotinu yang nantinya akan digabungkan dan diproses melalui layer MLP. Hal ini secara drastis mereduksi noise dan mempercepat waktu pelatihan tanpa mengorbankan akurasi. Konfigurasi arsitektur TabTransformer yang digunakan mengadopsi kombinasi 6 layer Transformer dan 8 attention heads. Berdasarkan literatur asli TabTransformer, kedalaman 6 layer terbukti optimal untuk menangkap interaksi non-linear yang kompleks pada data tabular tanpa menyebabkan overfitting. Jumlah 8 heads ini disesuaikan dengan dimensi embedding total sebesar 64, di mana setiap head memproses ruang dimensi berukuran 8 (8 heads x 8 dimensi = 64 dimensi). Penggunaan dimensi 64 dipilih karena memberikan keseimbangan ideal antara kapasitas representasi model dalam memetakan interaksi fitur yang kompleks dan efisiensi penggunaan memori komputasi, menjadikannya sangat sesuai untuk klasifikasi malware berskala besar.

E. Evaluasi Model

Evaluasi kinerja model dilakukan secara komprehensif menggunakan metrik kuantitatif yang diturunkan dari tabel *Confusion Matrix*, yang memetakan hasil prediksi ke dalam empat kategori: *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN). *Confusion Matrix* berfungsi sebagai fondasi visual dan matematis untuk mengevaluasi tingkat kesalahan dan kebenaran klasifikasi model.

	Predicted Result	
True Class	Positive	Negative
Positive	*TP*	*FN*
Negative	*FP*	*TN*

Gambar 4.Tabel Confusion Matrix

Berdasarkan matriks tersebut, evaluasi dijabarkan melalui beberapa metrik berikut:

- ***Accuracy*** mengukur proporsi keseluruhan prediksi yang benar dari total sampel, memberikan gambaran umum kinerja model secara menyeluruh.

$$A_{cc} = \frac{TP + TN}{TP + TN + FP + FN}$$

Gambar 5.Rumus Accuracy

- ***Precision*** menilai tingkat keandalan model saat memprediksi kelas positif (*malware*), yang krusial untuk meminimalisasi *false alarm* atau peringatan palsu.

$$P = \frac{TP}{TP + FP}$$

Gambar 6.Rumus Precision

- ***Recall*** mengukur kemampuan model dalam mendeteksi seluruh sampel *malware* yang sebenarnya, sebuah metrik yang sangat vital dalam keamanan siber agar tidak ada ancaman berbahaya yang terlewat.

$$R = \frac{TP}{TP + FN}$$

Gambar 7.Rumus Recall

- ***F1-Score*** merupakan rata-rata harmonik dari *Precision* dan *Recall*, memberikan evaluasi yang seimbang dan *robust*, terutama pada kondisi di mana kedua metrik tersebut sama-sama penting untuk dioptimalkan.

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

Gambar 8.Rumus F1-Score

- ***False Positive Rate*** **(FPR)** menghitung persentase aplikasi *benign* yang secara keliru diklasifikasikan sebagai *malware*, yang berdampak langsung pada pengalaman pengguna akibat pemblokiran aplikasi yang sah.

$$FPR = \frac{FP}{FP + TN}$$

Gambar 9.Rumus False Positive Rate

e-ISSN: 2548-6861

- ***ROC-AUC* (*Receiver Operating Characteristic - Area Under Curve*)** mengevaluasi kemampuan diskriminasi model secara keseluruhan pada berbagai ambang batas (*threshold*) klasifikasi, di mana nilai AUC yang lebih tinggi mengindikasikan model yang lebih handal dalam membedakan antara aplikasi *malware* dan *benign* secara independen dari *threshold* yang ditentukan.

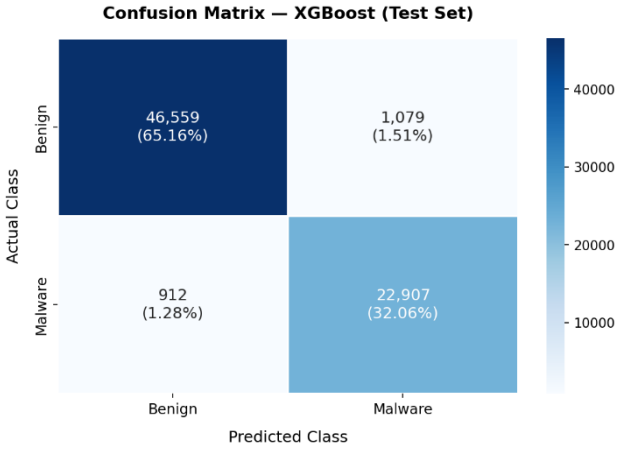
III. Hasil dan Pembahasan

A. Gambaran Umum

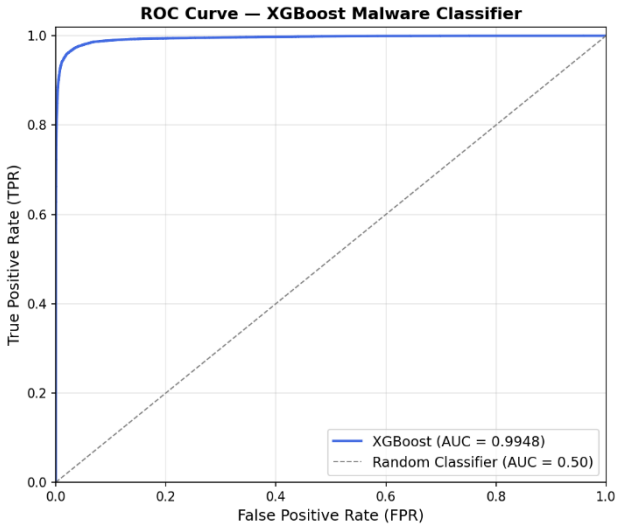
Eksperimen dalam penelitian ini dilakukan menggunakan varian dataset MH-1M yang telah melalui tahap praproses, meliputi penyeimbangan kelas dengan teknik *undersampling* (rasio 1:2). Dataset akhir yang digunakan untuk pelatihan dan evaluasi terdiri dari 357.282 sampel dengan 12.409 fitur multidimensi. Data tersebut kemudian dipartisi menjadi tiga subset dengan rasio 60% untuk data latih (214.368 sampel), 20% untuk data uji (71.457 sampel), dan 20% untuk data validasi (71.457 sampel). Dua model dievaluasi secara komparatif, yaitu XGBoost dan TabTransformer, untuk mengukur efektivitasnya dalam mengklasifikasikan *malware* Android berdasarkan metrik *Accuracy, Precision, Recall, F1-Score, ROC-AUC, False Positive Rate*, serta analisis *Confusion Matrix*.

B. Hasil Kinerja Performa XGBoost

Berdasarkan hasil eksperimen, algoritma XGBoost menunjukkan performa klasifikasi yang sangat solid dan konsisten pada data uji. Model ini berhasil mencatatkan tingkat akurasi (Accuracy) sebesar 97,21% dan nilai ROC-AUC yang sangat tinggi yaitu 0,9948, yang mengindikasikan kemampuan diskriminasi model yang luar biasa dalam membedakan kelas malware dan benign. Dari sisi metrik presisi dan sensitivitas, XGBoost mencapai Precision makro sebesar 96,79% dan Recall makro sebesar 96,95%, yang berujung pada F1-Score sebesar 96,87%.



Gambar 10.Grafik Confusion Matrix Model XGBoost

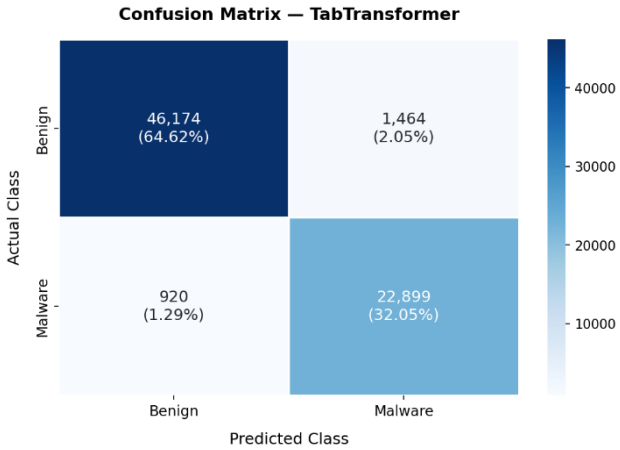


Gambar 11.Grafik ROC Curve Model XGBoost

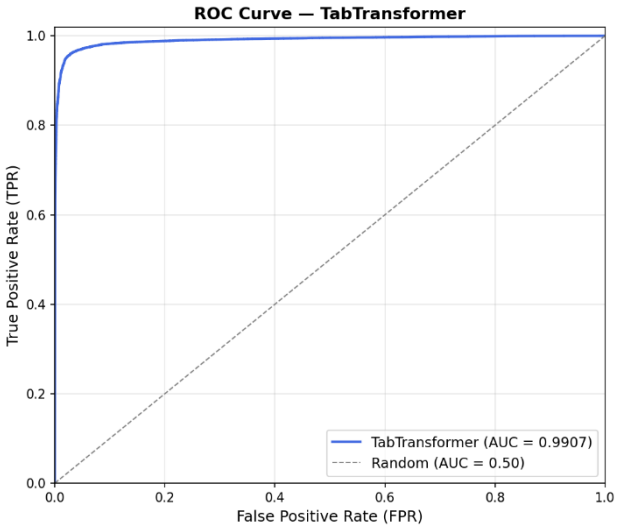
Analisis mendalam melalui Confusion Matrix memperlihatkan bahwa dari total 71.457 sampel uji, model berhasil mendeteksi 22.907 True Positives (aplikasi malware yang terdeteksi dengan benar) dan 46.559 True Negatives (aplikasi benign yang diklasifikasikan dengan benar). Tingkat kesalahan klasifikasi model ini sangat minimal, yang dibuktikan dengan hanya 912 False Negatives (1,28%) dan 1.079 False Positives (1,51%). Rendahnya angka False Positives ini sangat krusial dalam konteks keamanan siber, karena berarti model jarang menghasilkan peringatan palsu (false alarm) yang dapat memblokir aplikasi sah milik pengguna.

C. Hasil Kinerja Performa TabTransformer

Di sisi lain, arsitektur Deep Learning TabTransformer juga mencatatkan hasil yang sangat kompetitif. Model dengan konfigurasi 6 layer, 8 heads, dan dimensi 64 ini menghasilkan akurasi sebesar 96,66% dengan nilai ROC-AUC mencapai 0,9907. Secara teoritis, mekanisme multi-head self-attention pada TabTransformer dirancang untuk menangkap interaksi kontekstual antar-fitur, yang tercermin dari perolehan Precision makro sebesar 96,02% dan Recall makro sebesar 96,53%, serta F1-Score sebesar 96,27%.



Gambar 12.Grafik Confusion Matrix Model TabTransformer



Gambar 13.Grafik ROC Curve Model TabTransformer

Berdasarkan Confusion Matrix, TabTransformer berhasil memprediksi 22.899 True Positives dan 46.174 True Negatives. Namun, model ini menghasilkan sedikit lebih banyak kesalahan klasifikasi dibandingkan XGBoost, dengan catatan 920 False Negatives (1,29%) dan 1.464 False Positives (2,05%). Peningkatan jumlah False Positives ini mengindikasikan bahwa meskipun contextual embeddings mampu memetakan pola yang kompleks, model cenderung sedikit lebih agresif dalam mengklasifikasikan beberapa aplikasi benign yang memiliki atribut statis menyerupai malware sebagai ancaman.

D. Analisis Komparatif Kinerja dan Efisiensi Komputasi

Untuk memberikan gambaran yang komprehensif mengenai perbandingan kapabilitas kedua model, berikut adalah tabel yang merangkum metrik evaluasi utama dari arsitektur XGBoost dan TabTransformer pada data uji:

TABEL 1
PERBANDINGAN METRIK EVALUASI XGBOOST DAN TABTRANSFORMER

Metrik Evaluasi	XGBoost	TabTransformer
Accuracy	97,21%	96,66%
Precision (Macro)	96,79%	96,02%
Recall (Macro)	96,95%	96,53%
F1-Score (Macro)	96,87%	96,27%
ROC-AUC	0,9948	0,9907
False Positive Rate	1,51%	2,05%

Secara kuantitatif, XGBoost unggul atas TabTransformer pada seluruh metrik prediktif, meskipun dengan selisih yang tergolong tipis, yaitu pada rentang 0,42 hingga 0,80 poin persentase. Pada metrik *Accuracy*, yang merepresentasikan proporsi keseluruhan prediksi benar terhadap total sampel uji, XGBoost mencatatkan skor 0,55 poin persentase lebih tinggi (97,21% berbanding 96,66%). Pada F1-Score, yang mencerminkan keseimbangan harmonik antara *Precision* dan *Recall* sehingga menggambarkan kemampuan model menjaga *trade-off* antara *false alarm* dan *malware* yang lolos deteksi, XGBoost unggul 0,60 poin persentase (96,87% berbanding 96,27%). Selisih paling signifikan ditemukan pada *False Positive Rate* (FPR), yaitu proporsi aplikasi *benign* yang keliru diklasifikasikan sebagai *malware* terhadap seluruh aplikasi *benign* yang diuji. TabTransformer mencatatkan FPR 0,54 poin persentase lebih tinggi dibandingkan XGBoost (2,05% berbanding 1,51%). Dalam konteks keamanan siber, FPR yang lebih tinggi ini mengindikasikan bahwa TabTransformer berpotensi lebih sering menandai aplikasi sah sebagai ancaman, yang dapat mengganggu pengalaman pengguna meskipun selisih akurasi keseluruhannya tidak besar.

Apabila performa kedua model diuraikan lebih lanjut per kelas, selisih tersebut dapat diketahui tidak berasal dari kemampuan mendeteksi *malware*, melainkan dari kemampuan mengenali aplikasi *benign* secara tepat. Berdasarkan Confusion Matrix, XGBoost berhasil mengidentifikasi 22.907 dari 23.819 sampel *malware* (*recall* kelas *malware* sebesar 96,17%), sedangkan TabTransformer mengidentifikasi 22.899 sampel (96,14%), dengan selisih hanya 8 sampel atau 0,03 poin persentase. Hasil ini menunjukkan bahwa mekanisme *self-attention* pada TabTransformer mampu menangkap pola karakteristik *malware* hampir setara dengan mekanisme *gradient boosting* pada XGBoost. Sebaliknya, pada kelas *benign*, XGBoost mencatatkan spesifisitas sebesar 97,73% (46.559 dari 47.638 sampel), sedangkan TabTransformer hanya mencapai 96,93% (46.174 dari 47.638 sampel), dengan selisih 385 sampel atau 0,80 poin persentase. Selisih pada kelas *benign* inilah yang menjadi kontributor utama perbedaan *Accuracy*, F1-Score, dan FPR secara keseluruhan.

Temuan ini mengindikasikan bahwa rendahnya sebagian metrik TabTransformer dibandingkan XGBoost kemungkinan besar bukan disebabkan oleh keterbatasan arsitektur *self-attention* itu sendiri, melainkan oleh karakteristik dataset yang digunakan. Seluruh fitur pada dataset MH-1M bernilai biner, yaitu hanya berupa angka 0 atau 1 yang menandakan tidak ada atau adanya suatu karakteristik pada aplikasi. Kondisi ini menghasilkan data dengan dimensi yang sangat tinggi namun sebagian besar nilainya bernilai 0, sehingga lebih sesuai dengan cara kerja XGBoost yang membagi data berdasarkan nilai fitur secara langsung melalui struktur pohon keputusan. Di sisi lain, TabTransformer dirancang untuk mengenali hubungan antar fitur yang lebih kompleks, kemampuan yang umumnya lebih bermanfaat pada data dengan fitur bernilai kontinu atau kategorikal dengan banyak variasi, bukan pada data biner yang sederhana seperti pada MH-1M. Oleh karena itu, keunggulan mekanisme *self-attention* pada TabTransformer belum dapat dimanfaatkan secara maksimal pada dataset ini. Dengan demikian, selisih performa yang ditemukan lebih mencerminkan kesesuaian antara karakteristik dataset dan cara kerja masing-masing algoritma, bukan bukti bahwa arsitektur Transformer secara umum kurang mampu dibandingkan pendekatan *gradient boosting* untuk tugas deteksi *malware* Android.

IV. KESIMPULAN

Penelitian ini telah mengevaluasi dan membandingkan performa arsitektur *Deep Learning* TabTransformer dengan algoritma *Machine Learning* XGBoost dalam mengklasifikasikan *malware* Android pada dataset berskala masif MH-1M dengan kondisi kelas yang tidak seimbang. Hasil eksperimen menunjukkan bahwa XGBoost secara konsisten unggul atas TabTransformer pada seluruh metrik evaluasi, meskipun dengan selisih yang tergolong tipis, yaitu pada rentang 0,42 hingga 0,80 poin persentase. XGBoost mencatatkan *Accuracy* sebesar 97,21%, unggul 0,55 poin persentase dibandingkan TabTransformer sebesar 96,66%. Pada F1-Score, XGBoost memperoleh 96,87%, unggul 0,60 poin persentase dibandingkan TabTransformer sebesar 96,27%. Sementara itu, XGBoost mencatatkan ROC-AUC sebesar 0,9948 berbanding 0,9907 pada TabTransformer, dengan *False Positive Rate* terendah sebesar 1,51%, unggul 0,54 poin persentase dibandingkan *False Positive Rate* TabTransformer sebesar 2,05%.

Analisis lebih lanjut terhadap Confusion Matrix menunjukkan bahwa kedua model memiliki kemampuan yang hampir setara dalam mendeteksi sampel *malware*, dengan selisih *recall* kelas *malware* yang hanya sebesar 0,03 poin persentase. Selisih performa yang lebih nyata justru ditemukan pada kemampuan mengklasifikasikan aplikasi *benign* secara tepat, di mana TabTransformer menghasilkan lebih banyak *false positive* dibandingkan XGBoost. Temuan ini mengindikasikan bahwa kesenjangan performa antara kedua model lebih dipengaruhi oleh karakteristik dataset, khususnya representasi fitur biner yang menghasilkan ruang fitur berdimensi tinggi dan *sparse*, dibandingkan keterbatasan mendasar dari mekanisme *self-attention* pada TabTransformer. Dengan demikian, dapat disimpulkan bahwa untuk data tabular berskala besar dengan karakteristik fitur biner dan *sparse* seperti pada deteksi *malware* Android, kerangka kerja *gradient boosting* seperti XGBoost tetap menjadi pilihan yang lebih optimal untuk menghasilkan performa klasifikasi terbaik, tanpa mengesampingkan potensi TabTransformer pada skenario dataset dengan fitur yang lebih kaya secara kontekstual.

e-ISSN: 2548-6861

DAFTAR PUSTAKA

[1] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A Review of Android Malware Detection Approaches Based on Machine Learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020, doi: 10.1109/ACCESS.2020.3006143.

[2] S. J. Altaha, A. Aljughaiman, and S. Gul, "A Survey on Android Malware Detection Techniques Using Supervised Machine Learning," *IEEE Access*, vol. 12, pp. 173168–173191, 2024, doi: 10.1109/ACCESS.2024.3485706.

[3] N. Z. Gorment, A. Selamat, L. K. Cheng, and O. Krejcar, "Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges, and Future Directions," *IEEE Access*, vol. 11, pp. 141045–141089, 2023, doi: 10.1109/ACCESS.2023.3256979.

[4] S. Nethala, P. Chopra, K. Kamaluddin, S. Alam, S. Alharbi, and M. Alsaffar, "A Deep Learning-Based Ensemble Framework for Robust Android Malware Detection," *IEEE Access*, vol. 13, pp. 46673–46696, 2025, doi: 10.1109/ACCESS.2025.3551152.

[5] M. Alshoulie and A. Mehmood, "Deep Learning Approaches for Malware Detection: A Comprehensive Review of Techniques, Challenges, and Future Directions," *IEEE Access*, vol. 13, pp. 118652–118677, 2025, doi: 10.1109/ACCESS.2025.3582875.

[6] O. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," *IEEE Access*, vol. 9, pp. 87936–87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[7] R. Gutierrez, W. Villegas-Ch, L. Naranjo Godoy, A. Mera-Navarrete, and S. Luján-Mora, "Application of Deep Learning Models for Real-Time Automatic Malware Detection," *IEEE Access*, vol. 12, pp. 107742–107756, 2024, doi: 10.1109/ACCESS.2024.3436588.

[8] C. Negi, P. Mishra, P. Chaudhary, and H. Vardhan, "A Review and Case Study on Android Malware: Threat Model, Attacks, Techniques and Tools," *Journal of Cyber Security and Mobility*, Mar. 2021, doi: 10.13052/jcsm2245-1439.1018.

[9] S. Almarri, A. Bodokhi, and M. Frikha, "A Review of the Recent Trends in Mobile Malware Evolution, Detection, and Analysis," *IEEE Access*, vol. 13, pp. 108415–108445, 2025, doi: 10.1109/ACCESS.2025.3582086.

[10] H. Bragança, D. Kreutz, V. Rocha, J. Assolin, and E. Feitosa, "MH-1M: A 1.34 Million-Sample Multi-Feature Android Malware Dataset with Rich Metadata," *Sci. Data*, vol. 13, no. 1, p. 153, Dec. 2025, doi: 10.1038/s41597-025-06469-5.

[11] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, "TabTransformer: Tabular Data Modeling Using Contextual Embeddings," Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.06678

[12] Haoyu Wu and Jingwen Zhang, "TabTransformer-Based Credit Default Prediction for Structured Economic Data," *Information Systems and Economics*, vol. 6, no. 2, 2025, doi: 10.23977/infse.2025.060210.

[13] K. Shi and J. Li, "Leveraging Self-Attention for Heterogeneous Data Integration: A TabTransformer Validation on a Biomedical Case Study," in *Proceedings of the 19th International Joint Conference on Biomedical Engineering Systems and Technologies*, SCITEPRESS - Science and Technology Publications, 2026, pp. 680–687. doi: 10.5220/0014482500004070.

[14] "MH-1M Dataset," Jun. 2025. [Online]. Available: https://figshare.com/articles/dataset/_b_MH-1M_Dataset_b_/28355897

[15] H. Bragança and V. Rocha, Eds., "GitHub - Malware-Hunter/MH-1M," 2025. [Online]. Available: https://github.com/Malware-Hunter/MH-1M

[16] A. Moore and M. Bell, "XGBoost, A Novel Explainable AI Technique, in the Prediction of Myocardial Infarction: A UK Biobank Cohort Study," *Clin. Med. Insights Cardiol.*, vol. 16, Jan. 2022, doi: 10.1177/11795468221133611.

[17] V. Siva Kumar, M. P. Chinnasamy, and S. R. Kumar, "Deep Learning Framework With Optuna-Based Hyperparameter Tuning for Predicting Dry Turning Process Performance of 42CrMo4 Steel," *IEEE Access*, vol. 14, pp. 2119–2133, 2026, doi: 10.1109/ACCESS.2025.3649243.