denar

by Denar 5

Submission date: 20-Jul-2023 02:29PM (UTC+0700)

Submission ID: 2133957124

File name: Analisis Ekstraksi Fitur dan Klasifikasi Data Keluarga Malware Menggunakan Convolutional Neural Network.pdf (488.28K)

Word count: 2593

Character count: 15157

Seminar Keinsinyuran 2023
ISSN (Cetak) 2798-0405
eISSN (Online) 2797-1775

Analisis Ekstraksi Fitur dan Klasifikasi Data Keluarga Malware Menggunakan *Convolutional Neural Network*

Denar Regata Akbi[1], Didin Suhardi[2]
[1] Universitas Muhammadiyah Malang
[2] Program Profesi Insinyur, Universitas Muhammadiyah Malang, Jl. Raya Tlogomas 246 Malang

Kontak Person:
Denar Regata Akbi
Jl. Raya Tlogomas 246 Malang
dnarregata@umm.ac.id

Malware merupakan perangkat lunak berbahaya yang dapat mengganggu kinerja dari suatu sistem, dan telah menjadi salah satu cyber threat yang perlu mendapat perhatian khusus. Semakin hari perkembangan malware semakin berbagai macam dan mengalami evolusi semakin canggih, sehingga mempunyai kemampuan untuk melindungi diri dari suatu acaman baik itu antivirus atau sistem pengamanan yang lain, Salah satu upaya awal yang dapat dilakukan adalah melakukan analisis terhadap malware – malware yang ada, analisis dalam hal ini merupakan suatu proses untuk melakukan identifikasi terhadap perilaku malware, mulai dari apa yang dilakukan, apa yang diinginkan, dan apa tujuan utama dari malware tersebut, deep learning yang merupakan cabang ilmu dari kecerdasan buatan untuk melakukan penelitian terhadap karakteristik malware, dengan melakukan analisis terhadap karakteristik dari suatu varian malware, seperti menggunakan metode klasifikasi diharapkan hal tersebut dapat memberikan referensi untuk pembuatan sistem pengamanan terhadap malware yang lebih baik. Pada penelitian yang akan dilakukan peneliti mencoba untuk melakukan analisis terhadap data malware yang diambil dari Canadian Institute for Cybersecurity. Dalam hasil analisis tersebut didapatkan hasil precission dan recal 75%.

Kata kunci: *kecerdasan buatan, malware, deep learning, akurasi*

1. PENDAHULUAN

Sistem Operasi perangkat mobile yang paling populer saat ini adalah Android. Kebutuhan pasar terkait hal tersebut semakin meningkat sekitar 84,7% [1]. Disamping kebutuhan yang semakin meningkat tersebut terdapat sisi lain yang harus menjadi perhatian pengguna, hal tersebut merupakan sisi keamanan dari Sistem Operasi tersebut, menurut Fortinet terdapat lebih kurang 1800 macam malware yang dapat menyerang Sistem Operasi Android [2], sedangkan menurut Tencent Mobile Security Lab's terdapat sekitar 1.829.188 malware dan diprediksi akan terus berkembang sampai 12 kali lebih banyak [1].

Untuk meminimalisir serangan terhadap Sistem Operasi Android tersebut, perlu dilakukan kajian, penelitian terhadap data – data dari malware tersebut agar pembuat antivirus ataupun antimalware dapat terus mengembangkan perangkat lunak yang dapat digunakan untuk meminimalisir terjadinya serangan – serangan malware.

Terdapat beberapa peneliti yang telah melakukan analisis terhadap data serangan – serangan malware dan menggunakan berbagai macam metode. Metode deep learning merupakan metode yang populer yang digunakan oleh beberapa peneliti. Pada penelitian terkait peneliti menggunakan metode *CNN (Convolutional Neural Network)* dan *DBN (Deep Belief Network)*, pada penelitian ini peneliti melakukan analisis data malware pada android, hal ini dilakukan karena Sistem Operasi Android merupakan Sistem Operasi Open Source sehingga malware bisa melakukan kamuflase menjadi aplikasi yang dapat dipercaya dan aman kemudian diletakkan pada market android[3]. Pada penelitian lain, peneliti juga menggunakan metode menggunakan metode *CNN*, pada penelitian ini peneliti mengambil data sample malware dari sensor *honeypot*, kemudian peneliti mencoba untuk melakukan pendeteksian malware menggunakan metode CNN, pada percobaan yang dilakukan metode CNN sangat efektif untuk mengidentifikasi *malware* yang tertanam ke dalam kode aplikasi yang aman, sehingga *malware* tidak memiliki tempat untuk bersembunyi[4]. Pada penelitian lainnya, peneliti melakukan deteksi dan klasifikasi malware family menggunakan Network Flow dan API-Calls. Algoritma machine learning yang digunakan adalah algoritma Random Forest. Hasil dari penelitian ini adalah metode yang digunakan sukses meningkatkan precision sebesar 95,3% pada malware Binary Classification , precision

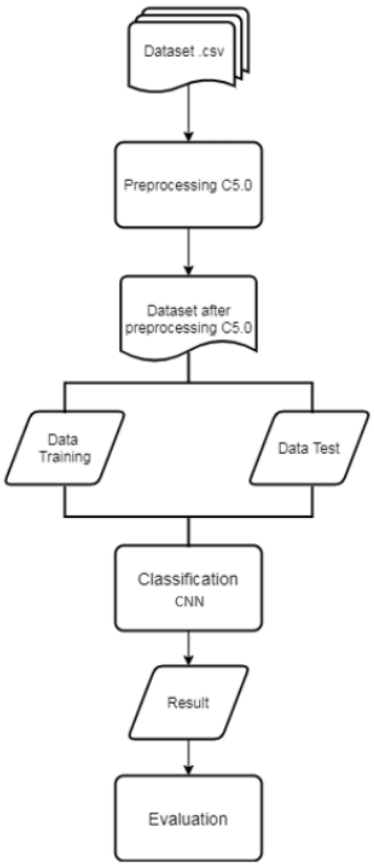
ISSN (Cetak) 2798-0405
eISSN (Online) 2797-1775

sebesar 83.3% pada Malware Category Classification dan precision sebesar 59,7% pada malware Family Classification [5].

Berdasarkan beberapa penelitian sebelumnya, peneliti mencoba untuk melakukan analisis data malware berdasarkan jenis familinya dengan data yang sama dengan yang digunakan oleh peneliti yang menggunakan data malware dengan fitur Network Flow dan API-Calls, dengan artikel yang berjudul "Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls" [5]. Pada penelitian ini penelitia mencoba untuk melakukan ekstraksi fitur dan dikombinasikan dengan metode CNN, ekstraksi fitur dilakukan bertujuan untuk mencoba mendapatkan hasil akurasi deteksi malware lebih baik.

2. METODE PENELITIAN

Tahapan penelitian yang dilakukan diantaranya melakukan pengumpulan data, pre-processing data, ekstraksi fitur serta klasifikasi data. Pada penelitian ini peneliti menggunakan dataset CICInvesAndMal2019. Gambar 1 merupakan gambar tahapan penelitian yang dilakukan oleh peneliti.



Gambar 1. Tahapan Penelitian

2.1 DATASET MALWARE

Dataset CICInvesAndMal2019 merupakan dataset yang didapat dari website Canadian Institute for Cybersecurity [5]. Dataset tersebut terdiri dari 305 sampel malware yang berasal dari 4 kategori malware dan 39 jenis keluarga malware family, terlihat pada tabel 1

ISSN (Cetak) 2798-0405
eISSN (Online) 2797-1775

Tabel 1 Dataset *CICInvesAndMal2019*

No	Category	Malware Family	Number of Sample
1	Adware	Dowgin family	6
		Ewind family	10
		Feiwo family	5
		Gooligan family	10
		Kemoge family	11
		Mobidash family	8
		Selfmite family	3
		Shuanet family	7
		Youmi family	10
2	Ransomware	Charger family	10
		Jisut family	8
		Koler family	7
		LockerPin family	3
		Simplocker family	3
		Pletor family	8
		PornDroid family	7
		RansomBO family	10
		Svpeng family	6
		WannaLocker family	6
3	Scareware	AndroidDefender	16

		AndroidSpy.277 family	6
		AV for Android family	9
		AVpass family	10
		FakeApp family	4
		FakeAV family	10
		FakeJobOffer family	8
		FakeTaoBao family	7
		Penetho family	10
		VirusShield family	9
4	SMS Malware	BeanBot family	10
		Biige family	3
		FakeInst family	6
		FakeMart family	9
		Jifake family	10
		Mazarbot family	10
		Nandrobox family	10
		Plankton family	6
		SMSsniffer family	9
		Zsone family	10

2.2 PREPROCESSING

Tahap *Preprocessing* dilakukan untuk membersihkan data, reduksi data, serta diskretisasi data. Fase ini bertujuan untuk membuat dataset lebih presisi serta menghilangankan *noise* dan pengurangan atribut [6]. Metode yang digunakan dalam proses *preprocessing* adalah metode C5.0.

2.3 ALGORITMA C5.0

Algoritma C5.0 merupakan algoritma klasifikasi dan menggunakan Teknik decision tree. Algoritma ini pengembangan dari algoritma C4.5[7]. Nilai terbesar dari suatu atribut akan dijadikan suatu inti node selanjutnya, informasi tersebut diperlukan untuk melakukan klasifikasi tiap sample yang digunakan dengan menggunakan persamaan (1), kemudian nilai himpunan bagian dari atribut A akan dihintung menggunakan persamaan (2), setelah itu informasi akan didapat menggunakan perhitungan pada persamaan (3)[8].

$$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} p_i \log_2 (p_i) \quad (1)$$

$$E(A) = \sum_{j=1}^{y} \frac{s_{1j}+\cdots+s_{mj}}{s} I(s_{1j}, \ldots, s_{mj}) \quad (2)$$

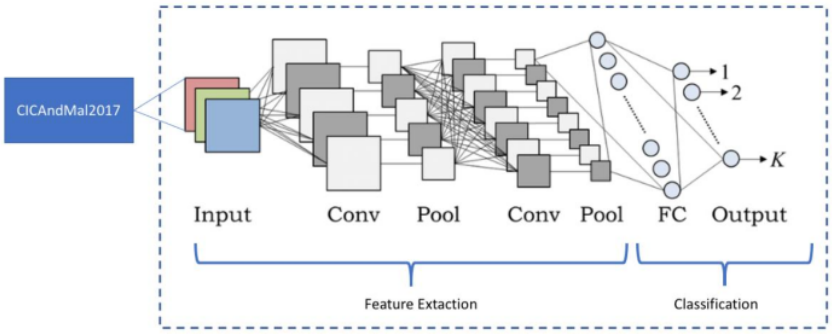
$$Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A) \quad (3)$$

Keterangan :

S	: jumlah data sampel
Si	: jumlah sampel dalam S di Ci class
m	: jumlah atribut class
pi	: pembagian class dan estimasi menggunakan s i/s
E(A)	: nilai bagian dari himpunan atribut A
Sj	: sampel dari S yang didapatkan dari nilai A
Sij	: jumlah sampel di class Ci dari himpunan Sj
Gain (A)	: informasi yang didapatkan dari atribut A

2.4 CONVOLUTIONAL NEURAL NETWORK

Secara umum arsitektur metode CNN yang akan digunakan pada penelitian yang akan dilakukan terlihat pada Gambar 2. Langkah-langkah penggunaan algoritma CNN memasukkan dataset yang telah dilakukan pada proses preprocessing, CNN otomatis akan memisahkan tiap fitur sesuai dengan label, memasukkan fitur yang telah dipisahkan ke dalam *array*, melakukan proses max-pooling untuk mengecilkan ukuran *array*, dan membuat prediksi indikator dari setiap *malware* sesuai dengan *family*nya. Pengujian dilakukan untuk mengetahui hasil akurasi dari algoritma yang telah dilakukan..



Gambar 2. Arsitektur metode CNN secara umum

ISSN (Cetak) 2798-0405
eISSN (Online) 2797-1775

3. HASIL DAN PEMBAHASAN

Proses pre-processing yang dilakukan pada penelitian ini menghasilkan pengurangan fitur pada dataset yang sebelumnya terdapat 918 fitur menjadi 129 fitur. 129 fitur tersebut berpengaruh pada hasil klasifikasi yang dapat terlihat pada Tabel 2.

Tabel 2 Atribut dataset setelah *preprocessing*

No	Percentage	*attribute*
1.	100.00%	X2Grammed_APICalls_.7
2.	100.00%	X2Grammed_APICalls_.53
3.	100.00%	X2Grammed_APICalls_.416
4.	100.00%	Min_Packet_Length
5.	100.00%	PSH_Flag_Count
6.	100.00%	Down.Up_Ratio
7.	97.30%	Bwd_Packets.s
8.	94.05%	X2Grammed_APICalls_.142
9.	94.05%	X2Grammed_APICalls_.411
10.	94.05%	Fwd_Packets.s
11.	82.70%	X2Grammed_APICalls_.357
12.	81.08%	X2Grammed_APICalls_.41
13.	81.08%	X2Grammed_APICalls_.50
14.	81.08%	X2Grammed_APICalls_.329
15.	80.00%	X2Grammed_APICalls_.30
16.	75.68%	X2Grammed_APICalls_.301
17.	71.89%	X2Grammed_APICalls_.59
18.	66.49%	X2Grammed_APICalls_.263
19.	65.41%	X2Grammed_APICalls_.125
20.	64.32%	X2Grammed_APICalls_.342
21.	63.78%	X2Grammed_APICalls_.364
22.	61.62%	Fwd_PSH_Flags
23.	57.84%	Total_Length_of_Fwd_Packets
24.	57.84%	Idle_Min
25.	55.68%	X2Grammed_APICalls_.109
26.	55.14%	X2Grammed_APICalls_.217
27.	53.51%	X2Grammed_APICalls_.796
28.	52.97%	Fwd_IAT_Min
29.	52.43%	X2Grammed_APICalls_.90
30.	49.19%	X2Grammed_APICalls_.1
31.	47.57%	X2Grammed_APICalls_.255
32.	44.32%	X2Grammed_APICalls_.532
33.	40.00%	act_data_pkt_fwd
34.	38.92%	X2Grammed_APICalls_.48
35.	35.68%	X2Grammed_APICalls_.160
36.	35.68%	Idle_Std
37.	35.14%	X2Grammed_APICalls_.408
38.	34.05%	X2Grammed_APICalls_.4
39.	34.05%	X2Grammed_APICalls_.89
40.	33.51%	X2Grammed_APICalls_.331
41.	31.89%	X2Grammed_APICalls_.14
42.	31.89%	X2Grammed_APICalls_.38
43.	31.35%	X2Grammed_APICalls_.116
44.	31.35%	Active_Std
45.	29.19%	X2Grammed_APICalls_.138

ISSN (Cetak) 2798-0405
eISSN (Online) 2797-1775

46.	28.65%	X2Grammed_APICalls_.22
47.	28.65%	X2Grammed_APICalls_.34
48.	28.11%	X2Grammed_APICalls_.111
49.	27.57%	X2Grammed_APICalls_.5
50.	27.57%	X2Grammed_APICalls_.35
51.	27.03%	X2Grammed_APICalls_.86
52.	25.95%	X2Grammed_APICalls_.193
53.	25.95%	X2Grammed_APICalls_.222
54.	25.95%	X2Grammed_APICalls_.353
55.	25.41%	X2Grammed_APICalls_.121
56.	25.41%	X2Grammed_APICalls_.296
57.	25.41%	X2Grammed_APICalls_.344
58.	24.86%	X2Grammed_APICalls_.240
59.	24.86%	X2Grammed_APICalls_.283
60.	24.86%	X2Grammed_APICalls_.303
61.	23.78%	X2Grammed_APICalls_.13
62.	23.78%	X2Grammed_APICalls_.363
63.	23.24%	X2Grammed_APICalls_.99
64.	23.24%	X2Grammed_APICalls_.320
65.	22.70%	X2Grammed_APICalls_.2
66.	22.16%	X2Grammed_APICalls_.12
67.	22.16%	Flow_Packets.s
68.	21.62%	X2Grammed_APICalls_.24
69.	21.62%	X2Grammed_APICalls_.66
70.	21.62%	X2Grammed_APICalls_.309
71.	21.62%	Flow_Bytes.s
72.	21.62%	Init_Win_bytes_backward
73.	21.08%	X2Grammed_APICalls_.213
74.	21.08%	Fwd_Packet_Length_Min
75.	21.08%	Idle_Max
76.	20.54%	X2Grammed_APICalls_.56
77.	20.00%	X2Grammed_APICalls_.264
78.	19.46%	X2Grammed_APICalls_.101
79.	19.46%	X2Grammed_APICalls_.108
80.	19.46%	X2Grammed_APICalls_.343
81.	18.92%	X2Grammed_APICalls_.265
82.	18.38%	X2Grammed_APICalls_.77
83.	17.30%	X2Grammed_APICalls_.0
84.	17.30%	X2Grammed_APICalls_.74
85.	17.30%	X2Grammed_APICalls_.170
86.	16.22%	X2Grammed_APICalls_.225
87.	15.68%	X2Grammed_APICalls_.3
88.	15.68%	X2Grammed_APICalls_.211
89.	15.14%	X2Grammed_APICalls_.28
90.	14.59%	X2Grammed_APICalls_.17
91.	14.59%	X2Grammed_APICalls_.40
92.	14.59%	X2Grammed_APICalls_.192
93.	13.51%	X2Grammed_APICalls_.8
94.	12.97%	X2Grammed_APICalls_.18
95.	12.97%	X2Grammed_APICalls_.165
96.	12.97%	X2Grammed_APICalls_.197
97.	12.97%	X2Grammed_APICalls_.293
98.	12.97%	URG_Flag_Count
99.	12.43%	Bwd_IAT_Mean

100.	11.89%	X2Grammed_APICalls_.311
101.	10.27%	X2Grammed_APICalls_.221
102.	10.27%	Flow_Duration
103.	9.73%	ACK_Flag_Count
104.	9.19%	X2Grammed_APICalls_.36
105.	9.19%	X2Grammed_APICalls_.393
106.	9.19%	X2Grammed_APICalls_.524
107.	8.65%	X2Grammed_APICalls_.19
108.	8.65%	X2Grammed_APICalls_.64
109.	8.65%	X2Grammed_APICalls_.84
110.	8.11%	X2Grammed_APICalls_.281
111.	7.57%	X2Grammed_APICalls_.137
112.	7.57%	X2Grammed_APICalls_.209
113.	7.57%	X2Grammed_APICalls_.220
114.	7.03%	X2Grammed_APICalls_.43
115.	7.03%	X2Grammed_APICalls_.91
116.	7.03%	X2Grammed_APICalls_.230
117.	7.03%	X2Grammed_APICalls_.328
118.	7.03%	Bwd_IAT_Total
119.	6.49%	X2Grammed_APICalls_.171
120.	5.95%	X2Grammed_APICalls_.51
121.	5.95%	X2Grammed_APICalls_.87
122.	5.95%	X2Grammed_APICalls_.205
123.	5.95%	X2Grammed_APICalls_.368
124.	5.41%	X2Grammed_APICalls_.212
125.	4.32%	X2Grammed_APICalls_.39
126.	4.32%	X2Grammed_APICalls_.229
127.	4.32%	Fwd_IAT_Total
128.	3.78%	X2Grammed_APICalls_.275
129.	3.78%	X2Grammed_APICalls_.663

3.1 Menambahkan *code* pada proses *split* data awal

Pengujian ini akan dilakukan penambahan *code* "shuffle=False", dimana akan menghasilkan akurasi dengan hasil yang sangat stabil diangka 0 karena dampak yang dihasilkan adalah data tidak acak sehingga data tidak bisa dijadikan validasi untuk dilakukan klasifikasi. Hasil tersebut tidak hanya berlaku untuk akurasi saja melainkan juga berdampak nilai precission dan recall.

	precision	recall	f1-score	support
13	0.00	0.00	0.00	0.0
14	0.00	0.00	0.00	0.0
15	0.00	0.00	0.00	0.0
17	0.00	0.00	0.00	0.0
19	0.00	0.00	0.00	0.0
23	0.00	0.00	0.00	0.0
25	0.00	0.00	0.00	0.0
26	0.00	0.00	0.00	0.0
35	0.00	0.00	0.00	1.0
36	0.00	0.00	0.00	3.0
37	0.00	0.00	0.00	3.0
38	0.00	0.00	0.00	6.0
accuracy			0.00	13.0
macro avg	0.00	0.00	0.00	13.0
weighted avg	0.00	0.00	0.00	13.0

Gambar 3. Hasil Pengujian *code* "shuffle=False"

Seminar Keinsinyuran 2023
ISSN (Cetak) 2798-0405
eISSN (Online) 2797-1775

3.2 Pengujian *learning rate*

	precision	recall	f1-score	support
0	0.50	1.00	0.67	1
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	2
9	1.00	1.00	1.00	1
12	1.00	1.00	1.00	1
13	1.00	1.00	1.00	1
14	0.00	0.00	0.00	0
20	0.00	0.00	0.00	0
22	1.00	0.50	0.67	2
25	1.00	1.00	1.00	1
27	1.00	1.00	1.00	1
28	1.00	1.00	1.00	1
34	0.00	0.00	0.00	1
36	1.00	1.00	1.00	1
accuracy			0.69	13
macro avg	0.61	0.61	0.60	13
weighted avg	0.73	0.69	0.69	13

Gambar 4. Learning Rate 2

	precision	recall	f1-score	support
0	0.50	1.00	0.67	1
5	0.00	0.00	0.00	2
7	0.00	0.00	0.00	0
9	0.00	0.00	0.00	1
12	0.00	0.00	0.00	1
13	0.00	0.00	0.00	1
18	0.00	0.00	0.00	0
22	0.00	0.00	0.00	2
25	0.00	0.00	0.00	1
27	0.00	0.00	0.00	1
28	0.00	0.00	0.00	1
33	0.00	0.00	0.00	0
34	0.00	0.00	0.00	1
36	0.00	0.00	0.00	1
accuracy			0.08	13
macro avg	0.04	0.07	0.05	13
weighted avg	0.04	0.08	0.05	13

Gambar 5. Learning Rate 1

3.3 Pengujian Dense

Dense merupakan suatu *hidden layer* dimana proses ini juga yang akan menentukan hasil dan waktu komputasi karena semakin banyak *dense/hidden layer* maka proses yang dilakukan akan semakin detail Gambar 6.

3.4 Pengujian *split test size*

Proses pengujian akan dilakukan setelah proses *split* data selesai dilakukan, dan untuk hasil dari penggantian *split* data. pada Gambar 7 yang menghasilkan hasil akurasi yang lebih besar dikarenakan

ISSN (Cetak) 2798-0405
eISSN (Online) 2797-1775

data train yang digunakan lebih banyak sehingga memungkinkan computer lebih mengenal bermacam data dan membuat akurasi menjadi lebih baik dibanding dengan split data yang sebesar 0.2 atau 20% untuk data test.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
5	0.50	0.50	0.50	2
9	0.00	0.00	0.00	1
12	1.00	1.00	1.00	1
13	0.00	0.00	0.00	1
14	0.00	0.00	0.00	0
18	0.00	0.00	0.00	0
19	0.00	0.00	0.00	0
22	0.00	0.00	0.00	2
25	0.50	1.00	0.67	1
27	1.00	1.00	1.00	1
28	1.00	1.00	1.00	1
29	0.00	0.00	0.00	0
34	0.00	0.00	0.00	1
36	0.00	0.00	0.00	1
accuracy			0.46	13
macro avg	0.33	0.37	0.34	13
weighted avg	0.42	0.46	0.44	13

Gambar 6. Dense

	precision	recall	f1-score	support
9	1.00	1.00	1.00	1
13	0.00	0.00	0.00	0
16	1.00	1.00	1.00	1
24	1.00	1.00	1.00	1
37	0.00	0.00	0.00	1
accuracy			0.75	4
macro avg	0.60	0.60	0.60	4
weighted avg	0.75	0.75	0.75	4

Gambar 7. Split Test Size

4. KESIMPULAN

Penelitian ini menghasilkan model bahwa penggunaan CNN tidak hanya terbatas untuk proses gambar melainkan juga memungkinkan untuk klasifikasi *malware* sesuai dengan *family* dapat dilakukan dan juga memiliki tingkat akurasi yang cukup tinggi apabila data dilakukan *split* sebesar 0.1 atau 10% untuk data *test* dan 90% untuk data *train*.

Proses dari klasifikasi menggunakan CNN ini dapat disimpulkan bahwa penggunaan *activation, optimizer*, jumlah dense dan pengaturan *split* data sangat berpengaruh terhadap hasil seperti *recall, precision* dan juga akurasi.

ISSN (Cetak) 2798-0405
eISSN (Online) 2797-1775

REFERENSI

[1] C. Da, H. Zhang, and X. Zhang, "Detection of Android malware security on system calls," *Proc. 2016 IEEE Adv. Inf. Manag. Commun. Electron. Autom. Control Conf. IMCEC 2016*, pp. 974–978, 2017.Casadei D, Serra G, Tani K. Implementation of a Direct Control Algorithm for Induction Motors Based on Discrete Space Vector Modulation. *IEEE Transactions on Power Electronics*. 2007; 15(4): 769-777. (*pada contoh ini Vol.15, Issues 4, and halaman 769-777*)

[2] M. Jaiswal, Y. Malik, and F. Jaafar, "Android gaming malware detection using system call analysis," *6th Int. Symp. Digit. Forensic Secur. ISDFS 2018 - Proceeding*, vol. 2018-Janua, pp. 1–5, 2018.

[3] C. Hasegawa and H. Iyatomi, "One-dimensional convolutional neural networks for Android malware detection," *Proc. - 2018 IEEE 14th Int. Colloq. Signal Process. its Appl. CSPA 2018*, no. March, pp. 99–102, 2018.

[4] D. Kornish, J. Geary, V. Sansing, S. Ezekiel, L. Pearlstein, and L. Njilla, "Malware Classification using Deep Convolutional Neural Networks," *Proc. - Appl. Imag. Pattern Recognit. Work.*, vol. 2018-Octob, pp. 1–6, 2018.

[5] Taheri, L., Kadir, A. F. A., & Lashkari, A. H. (2019, October). *Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls*. In 2019 International Carnahan Conference on Security Technology (ICCST) (pp. 1-8). IEEE.

[6] Chandrasekar, P., & Qian, K. (2016, June). *The impact of data preprocessing on the performance of a naive bayes classifier*. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 2, pp. 618-619). IEEE

[7] Kurniawan, E., Nhita, F., Aditsania, A., & Saepudin, D. (2019, July). *C5. 0 algorithm and synthetic minority oversampling technique (SMOTE) for rainfall forecasting in Bandung regency*. In 2019 7th *International Conference on Information and Communication Technology* (ICoICT) (pp. 1-5). IEEE

[8] Sandag, G. A., Leopold, J., & Ong, V. F. (2018). K*lasifikasi Malicious Websites Menggunakan Algoritma K-NN Berdasarkan Application Layers dan Network Characteristics*. CogITo Smart Journal, 4(1), 37-45.

denar

ORIGINALITY REPORT

11%	10%	3%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.umm.ac.id Internet Source	9%
2	Mahshid Gohari, Sattar Hashemi, Lida Abdi. "Android Malware Detection and Classification Based on Network Traffic Using Deep Learning", 2021 7th International Conference on Web Research (ICWR), 2021 Publication	1%
3	Atul Kumar, Ishu Sharma, Avinash Sharma. "Understanding the Behaviour of Android SMS Malware Attacks With Real Smartphones Dataset", 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), 2023 Publication	1%
4	lppm.stikma.ac.id Internet Source	1%

Exclude quotes	Off	Exclude matches	< 1%

Exclude bibliography On