Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 1 ; Februari 2024 ; Page 252-259
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index

Implementasi Neural Network 1-Dimensi Dalam Identifikasi Malware Android

Rofik Hidayat[1], Muhammad Koprawi[2], Pramudhita Ferdiansyah[3]

[1,2] Teknik Komputer, Universitas Amikom Yogyakarta, Sleman, Indonesia
[3] Teknik Informatika, Universitas Amikom Yogyakarta, Sleman, Indonesia
Email: [1]rofik.h@students.amikom.ac.id, [2,*]koprawi@amikom.ac.id, [3]ferdian@amikom.ac.id
Email Penulis Korespondensi: koprawi@amikom.ac.id

Article History:
Received Jan 10th, 2024
Revised Jan 30th, 2024
Accepted Feb 28th, 2024

Abstrak

Permasalahan *malware* setiap saat menjadi ancaman serius pada keamanan sistem informasi sehingga perlu adanya penanganan. Dengan meningkatnya penggunaan perangkat *mobile* ponsel pintar android menjadikannya target yang rentan terkena serangan *malware*. Untuk mencegah terjadinya serangan *malware* maka perlu adanya deteksi dini aplikasi yang berpotensi *malware*. Tiap aplikasi android memiliki *permission* untuk membuka hak akses agar aplikasi tersebut dapat mengakses informasi pada perangkat android, begitu juga dengan *malware* yang juga mempunyai permission tersebut. Teknologi *Machine Learning* dapat menyelesaikan masalah masalah yang rumit dengan meniru kecerdasan pada manusia. Salah satu teknologi tersebut adalah *Neural Network* yang merupakan Teknik *Machine learning* dengan strukturnya meniru cara kerja otak manusia, jenis *neural network* diantaranya adalah *Convolutional Neural Network* 1-Dimensi. Dengan menggunakan dataset sebanyak 510 aplikasi, model *Convolutional Neural Network* 1-Dimensi mampu mendapatkan 92.1% untuk tingkat akurasi, 93.4% untuk *recall* dan 89.5% untuk *precision*, dapat dikatakan model yang diusulkan sudah cukup baik dalam mengidentifikasi *malware* berdasarkan *permission*.
Kata Kunci : *Neural Network, Android Permission, Klasifikasi Malware*

Abstract

Malware problems at any time pose a severe threat to the security of information systems, so they need to be addressed. With the increasing use of Android smartphones, mobile devices make them vulnerable targets for malware attacks. There is a need for early detection of applications that have the potential to be Malware to prevent malware attacks. Every Android application has permission to open access rights so that the application can access information on the Android device, as well as Malware, which also has this permission. Machine Learning technology can solve complex problems by imitating human intelligence. One of these technologies is a Neural Network, a machine learning technique whose structure imitates how the human brain works. One type of neural network is a 1-dimensional Convolutional Neural Network. Using a dataset of 510 applications, the 1-dimensional Convolutional Neural Network model got 92.1% for accuracy, 93.4% for recall and 89.5% for precision. The proposed model is quite good at identifying Malware based on its permissions.
Keyword : *Neural Network, Android Permission, Malware Clasification*

1. PENDAHULUAN

Teknologi merupakan salah satu hal yang terus mengalami perkembangan yang pesat setiap tahunnya untuk mempermudah kehidupan manusia, salah satu perkembangannya adalah manusia dapat saling terhubung melalui internet menggunakan perangkat android. Android merupakan sistem operasi *mobile* yang paling banyak digunakan pada saat ini. Android menjadi kontributor utama pada pasar seluler dengan lebih dari 2 miliar perangkat aktif. Android memegang lebih dari 74% total pengguna smartphone dunia pada tahun 2019. Dengan popularitas yang dimiliki oleh Android dan ini memunculkan dampak negatif salah satunya adalah serangan *malware*[1][2].

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 1 ; Februari 2024 ; Page 252-259
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index

Malware atau *Malicious Software*, merupakan program jahat dengan maksud dan tujuan merugikan pihak lain[3], [4]. Sistem Android menjadi target utama *malware* seluler karena sistem operasi Android memungkinkan pengguna untuk menginstal aplikasi yang diunduh dari pasar pihak ketiga[5][6]. Android *malware* menyerang dengan berbagai cara dan salah satunya dengan menggunakan fitur *permission* yang ada pada aplikasi android[7][8]. Android *Permission* merupakan aturan dan batasan yang diterapkan pada sistem operasi android untuk menentukan apa yang dibolehkan dan tidak dibolehkan suatu aplikasi android dalam mengakses sumber daya dan fitur pada perangkat android[9]. Banyak pengguna yang tidak mengerti apa arti dari setiap *permission* dan memberikan *permission* tersebut tanpa memikirkan resiko yang memungkinkan aplikasi tersebut mengakses informasi sensitif pengguna[10].

Banyak metode dan pendekatan yang digunakan dalam mengidentifikasi dan mendeteksi *malware* android salah satunya adalah dengan menggunakan bantuan algoritma-algoritma *Machine Learning*[11]. *Machine Learning* merupakan serangkaian teknik yang dapat membantu dalam menangani dan memprediksi data dengan cara mengajarkan komputer untuk memiliki kecerdasan layaknya manusia, *machine learning* memiliki cabang algoritma yang mengambil inspirasi dari otak manusia bernama *Deep Learning*[12]. Salah satu algoritma *Deep Learning* yang paling umum digunakan dalam menyelesaikan masalah adalah algoritma *Convolutional Neural Network*. *Convolutional Neural Network* ini adalah hasil pengembangan algoritma *neural network* yang dirancang untuk mengelola data dua dimensi[13].

Convolutional Neural Network memiliki algoritma turunan bernama *Convolutional Neural Network 1-Dimension*. Pembeda utamanya terletak pada dimensinya, dimana *Convolutional Neural Network 1-Dimension* menggunakan *array 1D* untuk menggantikan matriks *2D* pada CNN pada umumnya[14]. Keunggulan utama dari *Convolutional Neural Network* 1-Dimensi dibanding *Convolutional Neural Network 2-Dimension* adalah mengurangi kompleksitas komputasi karena hanya menggunakan urutan data[15][16]. Karena persyaratan komputasinya yang rendah inilah *Convolutional Neural Network 1-Dimension* cocok untuk diimplementasikan pada aplikasi perangkat seluler[17]. Pada penelitian ini akan mencoba untuk mengimplementasikan algoritma *Convolutional Neural Network 1-Dimension* untuk mengidentifikasi *malware* berdasarkan *permission*nya, dan mengetahui seberapa efektifkah penerapan algoritma ini.

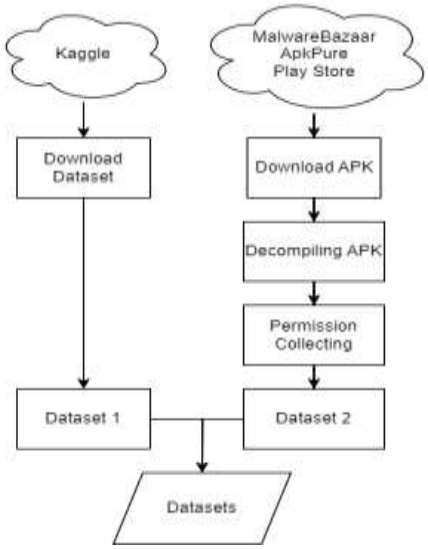
2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini memiliki beberapa tahapan pengolahan data hingga sampai pembuatan *model Convolutional Neural Network* 1-Dimensi untuk dapat mengidentifikasi *malware* android.

1. Data *Collection*

   Pada penelitian ini, pengumpulan data dilakukan dengan mengunduh dataset pada website https://www.kaggle.com/datasets/xwolf12/datasetandroidpermissions serta pengunduhan sampel *malware* dan aplikasi android melalui website https://bazaar.abuse.ch/, *ApkPure* dan *Playstore*. Selanjutnya sampel-sampel *malware* dan aplikasi android tersebut akan dilakukan proses dekompilasi menggunakan JADX untuk membongkar dan mengambil *permission* yang ada pada file *AndroidManifest.xml* dan mendatanya serta pemberian label sesuai dengan tipenya untuk membuat dataset kedua. Kedua dataset tersebut nantinya akan digabung menjadi satu dataset seperti pada Gambar 1.



Gambar 1. *Data Collecting*

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 1 ; Februari 2024 ; Page 252-259
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index

2. *Pre-Processing*

Pre-Processing data merupakan tahapan persiapan dimana data akan diolah dan disesuaikan untuk digunakan dalam pembuatan model.

```
print("Dataset 1 Shape:", df1.shape)
print("Dataset 2 Shape:", df2.shape)
```

```
Dataset 1 Shape: (398, 331)
Dataset 2 Shape: (112, 330)
```

```
#@markdown Mencari Kolom yang berbeda dari kedua dataset.
df1.columns.symmetric_difference(df2.columns)
```

```
Index(['NAME', 'android', 'test_permission'], dtype='object')
```

```
#@markdown Membuang kolom yang tidak diperlukan.
df1 = df1.drop('test_permission', axis=1)
df1 = df1.drop('android', axis=1)
df2 = df2.drop('NAME', axis=1)
print(df1.shape, df2.shape)
```

```
(398, 329) (112, 329)
```

```
#@markdown Menggabungkan kedua dataset menjadi satu.
df = pd.DataFrame()
df = df1.append(df2, ignore_index=True)
print(df.shape)
```

```
(510, 329)
```

Gambar 2. *Pre-Processing Dataset*

Seperti pada Gambar 2. mulanya dataset pertama (dataset yang di unduh dari kaggle) berdimensi (398x331) dan dataset kedua (dataset tambahan) berdimensi (112x330) akan dibandingkan bentuk datasetnya dan dicari dimana variabel yang menjadi pembeda dari kedua dataset tersebut. Selanjutnya variabel yang berbeda tersebut akan di drop untuk menyelaraskan kedua dataset sehingga mempunyai dimensi yang sama. Kedua dataset tersebut digabung menjadi satu dataset dengan dimensi (510x329) yang selanjutnya dicari apakah terdapat *NaN* valuenya. Dataset yang baru tersebut berisi 256 *malware* dan 254 aplikasi jinak seperti pada gambar 3. yang berarti dataset bersifat *balance*.

```
#@markdown 256 total apk Malware dan 254 total apk jinak
df.type.value_counts()
```

```
1    256
0    254
Name: type, dtype: int64
```

```
#@markdown Melakukan cek value NaN
print(df.isnull().values.any())
```

```
False
```

Gambar 3. *Cleaning* Dataset

Pada gambar 4. memperlihatkan proses berikutnya, yaitu memisahkan variabel dengan labelnya yang selanjutnya di *convert* dari *dataframe* menjadi *array numpy*. Dari sini akan dilakukan proses *splitting* data, yang berarti membagi antara data untuk proses *training* dan data untuk proses *testing*. penulis menggunakan *testsize 0.20* yang berarti membagi dataset menjadi 80% data *training* dan 20% data *testing*.

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 1 ; Februari 2024 ; Page 252-259
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index

```
#@markdown Memisahkan label dengan variabel
X = df.iloc[:,:-1]
y = df['type']
print(X.shape, y.shape)
```

```
(510, 328) (510,)
```

```
#@markdown Merubah dataframe menjadi numpy array
X = X.to_numpy()
y = y.to_numpy()
```

```
#@markdown Merubah dimensi numpy array
X = X.reshape(510, 328, 1)
y = y.reshape(510)
print(X.shape)
print(y.shape)
```

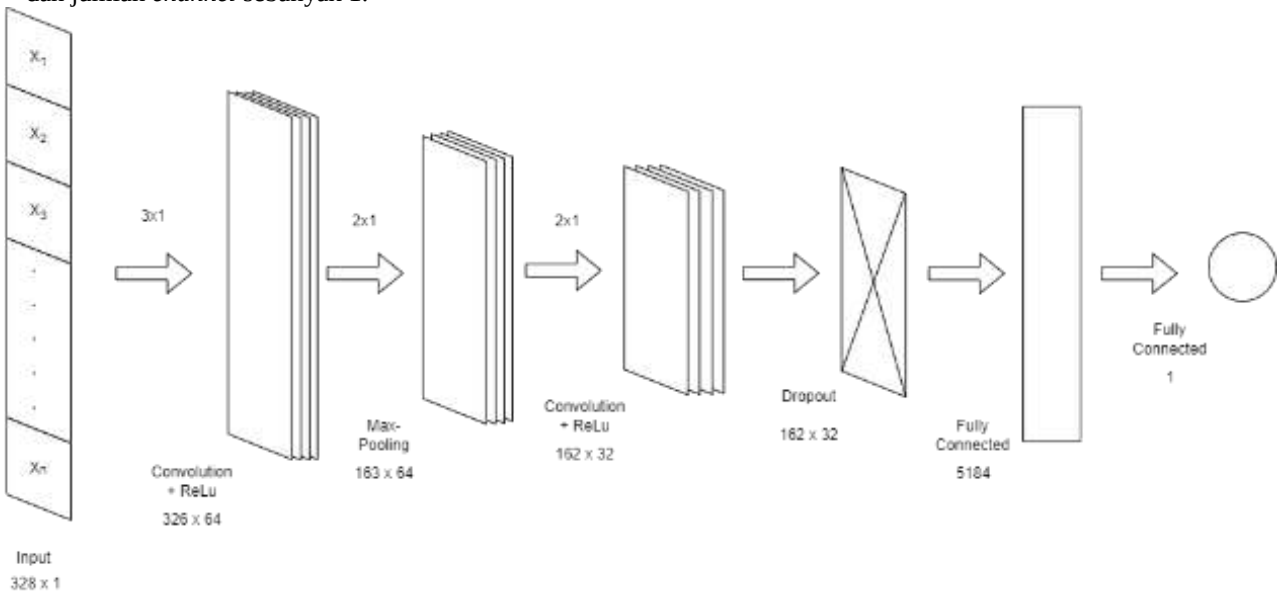
```
(510, 328, 1)
(510,)
```

```
#@markdown Membagi dataset menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20
print(X_train.shape)
print(y_train.shape)
```

```
(408, 328, 1)
(408,)
```

Gambar 4. *Splitting* Dataset

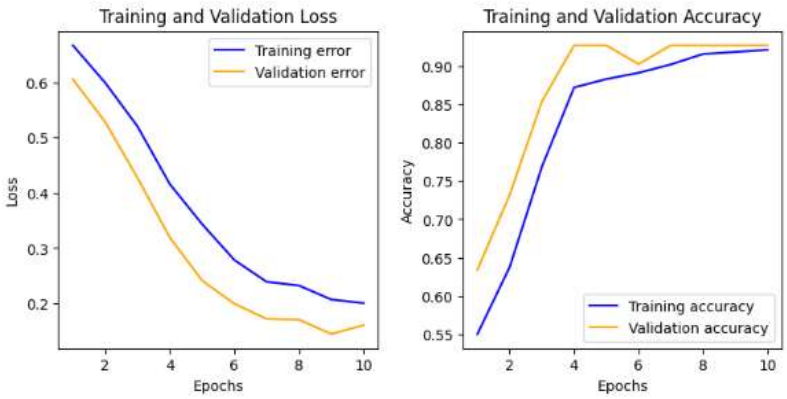
3. Pembuatan Model *CNN-1D*
   The Untuk melakukan identifikasi *malware* android metode yang digunakan adalah *Convolutional Neural Network* 1-Dimensi. Pada gambar 5 model tersebut menggunakan input *shape* (328 x 1). *Dalam Convolutional Neural Network* 1 dimensi, input data memiliki representasi 2 dimensi dengan rincian (panjang urutan, jumlah *channel*) dan *kernel* berjalan/bergeser dengan 1 dimensi, dengan demikian maka input shape pada gambar 5 memiliki ukuran panjang 328 dan jumlah *channel* sebanyak 1.



Gambar 5. Arsitektur *CNN-1D*

Input akan melalui 2 layer konvolusi yang menggunakan fungsi RELU dan *filter* 3x1 dan 2x1, dengan di antara layer konvolusi terdapat layer *Max-Pooling* 2x1, selanjutnya akan dilakukan *dropout* terhadap 60% *neuron* secara acak untuk mengurangi potensi *overfit*, kemudian masuk layer *fully connected* yang akan menghasilkan *Feature Vector* sebanyak 1. Pada layer *fully connected* ini terdapat optimasi Adam dengan *learning rate* 0.001 untuk mengurangi *loss* dan meningkatkan kualitas model.

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 1 ; Februari 2024 ; Page 252-259
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index

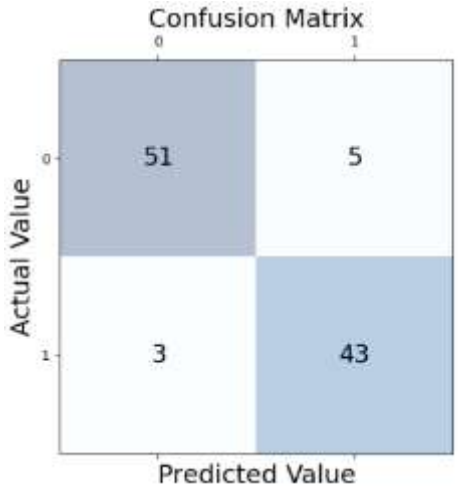


Gambar 6. *Loss* dan Akurasi proses *Training*.

Pada proses *training* menggunakan validation_split sebesar 10%, yang berarti model akan memisahkan 10% dari data latih/*training* untuk tidak digunakan sebagai pelatihan namun data ini digunakan untuk validasi *loss* dan akurasi pada setiap iterasi pelatihan, dengan menggunakan *epoch* sebanyak 10 iterasi pada proses pelatihan akan mendapatkan nilai *loss* sebesar 0.2007, akurasi sebesar 0.9210, validasi *loss* sebesar 0.1608, dan validasi akurasi sebesar 0.9268. Berdasarkan pada gambar grafik 6. penulis berpendapat model tersebut memiliki nilai akurasi cukup baik dalam memvalidasi dikarenakan setiap iterasi *epoch*, model mengalami penurunan *loss* dan peningkatan akurasi.

3. HASIL DAN PEMBAHASAN

Pengujian dilakukan dengan melakukan prediksi menggunakan dataset *testing* yang didapatkan dari proses *splitting* dataset sebelumnya dengan total sebanyak 102 data. Hasil pengujian penelitian ini didapatkan dengan dihitung menggunakan parameter pengujian *Precision*, *Recall*, *F1-Score* dan Akurasi pada *Confusion Matrix* seperti pada gambar 7.



Gambar 7. *Confution Matrix*

Parameter pengujian dihitung menggunakan persamaan 1-4,

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

$$F1\text{- }Score = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall} \quad (3)$$

$$\text{Akurasi} = \frac{TP+TN}{TP+FN+TN+FP} \quad (4)$$

Berdasarkan hasil dari tabel 1. penulis simpulkan model ini memiliki nilai *Recall* yang lebih tinggi daripada nilai *Precision*nya sehingga model ini bersifat *High Recall, Low Precision.* Berdasarkan sifatnya yang *High Recall* ini berarti model mampu mengidentifikasi dengan benar sebagian besar sample *malware* (*True Positive*) dari seluruh sample *malware* yang ada di dataset (*True Positive + False Negative*) sifat ini sangat penting dalam deteksi *malware*, karena melewatkan satu *malware* saja dapat menimbulkan konsekuensi yang buruk. Namun, model ini juga memiliki sifat *Low Precision* yang berarti model ini juga dapat mengkategorikan beberapa aplikasi jinak sebagai *malware* (*False Positive*). Hal ini dapat menyebabkan alarm palsu atau tindakan yang tidak perlu dilakukan terhadap aplikasi tersebut bagi sistem atau pengguna.

Tabel 1. Hasil pengujian

Parameter	**CNN 1D**
Precision	0.895833
Recall	0.934783
F1-Score	0.914894
Akurasi	0.921569

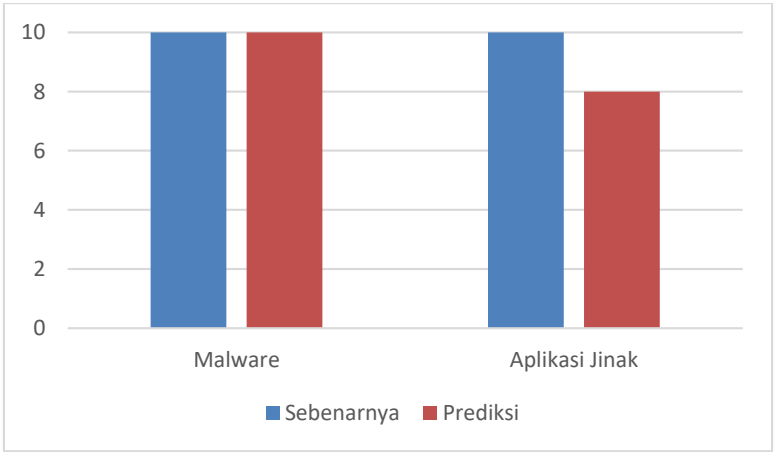
Model yang telah jadi dapat di implementasikan ke aplikasi untuk melakukan identifikasi atau prediksi *malware* dengan aplikasi jinak android berdasarkan *permission*nya. Disini peneliti mencoba membuat *prototype* program dengan menggunakan bahasa *python* untuk melakukan prediksi *malware* android dengan inputan aplikasi android.

```
1/1 [==============================] - 1s 628ms/step

PREDICTED

Malware |              5c01f7727c78dea9c89dccf92b01b4c45e69406e6462340779401497bf4d4589.apk
Malware |              634124330e5c60ab4838e61d83d74bb4cb59be86172f19f22d362d4ef0ceee6d.apk
Malware |              69b11e34e270deda7857e1bfa73a47c10843095c330e1e342ade8b20df370c9c.apk
Malware |              917e29ff91324a6c3630f0eb392a6d1a5c394d7112b35ef29e7cc0269e4c1445.apk
Malware |              9256586f657c81070a91f5ccb5a57ce89b188b3209906d592ccbd1b001c20f4c.apk
Benign  |              BosKos_Apkpure.apk
Benign  |              CalculatorVault.apk
Malware |              Carousell.apk
Benign  |              SoundCloud_Apkpure.apk
Benign  |              TelegramX_Apkpure.apk
Benign  |              Threads_Apkpure.apk
Benign  |              X.apk
Benign  |              YouTubeGo.apk
Malware |              a05497647a879afec62bc7e916005f729fbfee48cfd56423481e0600061678b6.apk
Malware |              aac13c7dec7b5acb804d4b896245962c395f1cd6dfce79bc9c96edcf65ae8e68.apk
Malware |              cbdc855825013478969d14d5943f0df6ce37cad338d02267bba31728112fd34b.apk
Malware |              dee1eaaa8879a7d321ef4e698203be7b23eeda80a6dea3c70cbf3138597b1800.apk
Malware |              e94920a1f3c3cfd2d901af1e21ad5d99676f02cb5c1464283b1919e7e853fbe1.apk
Benign  |              operamini.apk
Malware |              slither.io_Apkpure.apk
```

Gambar 8. *Output* Program

Dengan menggunakan 5 aplikasi yang diunduh melalui *ApkPure*, 5 aplikasi dari *Playstore* dan 10 *malware* dari *MalwareBazaar* maka diperoleh hasil output seperti pada gambar 8.

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 1 ; Februari 2024 ; Page 252-259
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index



Gambar 9. Hasil prediksi program

Berdasarkan hasil prediksi pada gambar 9, prediksi terhadap 20 aplikasi android menunjukkan hasil yang cukup baik dan efektif. Dari total 20 aplikasi ada 2 aplikasi yang diprediksi salah dengan begitu nilai akurasi yang didapat adalah 90%. Dengan rincian 10 malware dapat diprediksi dengan benar dan 8 aplikasi jinak dapat diprediksi dengan benar, aplikasi yang diprediksi salah merupakan 2 aplikasi jinak yang diprediksi menjadi *malware* (*False Positive*).

4. KESIMPULAN

Berdasarkan penelitian dan hasil pengujian metode *Convolutinal Neural Network* 1-Dimensi untuk identifikasi *malware* android berbasis *permission*, dapat diperoleh kesimpulan metode ini terbukti mampu dan cukup efektif untuk mendeteksi aplikasi *malware* dengan berdasarkan permissionnya dikarenakan sifatnya yang *High Recall*, dan *Low Precision*. Dengan menggunakan beberapa parameter pengujian, model dapat menghasilkan nilai 92.1% untuk tingkat akurasi, presisi 89.5% , dan *recall* 93.4%, dengan nilai *F1-Score* sebesar 91.4%. Hal ini menunjukkan bahwa model yang digunakan sudah cukup baik untuk mengidentifikasi *malware* berbasis *permission*. Namun model ini masih memiliki beberapa kekurangan pada segi presisi, diharapkan penelitian selanjutnya dapat mengembangkan dan mengoptimalkan lagi arsitektur model serta menggunakan dataset yang lebih besar dan lebih beragam untuk dapat meningkatkan kualitas dan akurasi model.

DAFTAR PUSTAKA

[1] A. Mathur, L. M. Podila, K. Kulkarni, Q. Niyaz, and A. Y. Javaid, "NATICUSdroid: A malware detection framework for Android using native and custom permissions," *Journal of Information Security and Applications*, vol. 58, no. January, p. 102696, 2021, doi: 10.1016/j.jisa.2020.102696.

[2] A. N. Iman, M. T. Avon Budiyono, S.T., and M. T. Ahmad Almaarif, S.Kom., "ANALISIS MALWARE PADA SISTEM OPERASI ANDROID MENGGUNAKAN PERMISSION-BASED MALWARE ANALYSIS IN ANDROID OPERATION SYSTEM USING PERMISSION-BASED," *Angewandte Chemie International Edition, 6(11), 951–952.*, vol. 6, no. Mi, pp. 5–24, 1967.

[3] S. Michael and A. Honig, "Practical Malware Analysis," *Network Security*, vol. 2012, no. 12, p. 4, 2012, doi: 10.1016/s1353-4858(12)70109-5.

[4] M. S. Akhtar and T. Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," *Symmetry 2022, Vol. 14, Page 2304*, vol. 14, no. 11, p. 2304, Nov. 2022, doi: 10.3390/SYM14112304.

[5] P. Faruki *et al.*, "Android security: A survey of issues, malware penetration, and defenses," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 998–1022, 2015, doi: 10.1109/COMST.2014.2386139.

[6] W. F. Elsersy, A. Feizollah, and N. B. Anuar, "The rise of obfuscated Android malware and impacts on detection methods," *PeerJ Comput Sci*, vol. 8, p. e907, Mar. 2022, doi: 10.7717/PEERJ-CS.907/SUPP-2.

[7] A. Arora, S. K. Peddoju, and M. Conti, "PermPair: Android Malware Detection Using Permission Pairs," *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 8, pp. 1968–1982, 2020, doi: 10.1109/TIFS.2019.2950134.

[8] P. Agrawal and B. Trivedi, "Unstructured Data Collection from APK files for Malware Detection," *Int J Comput Appl*, vol. 176, no. 28, pp. 42–45, 2020, doi: 10.5120/ijca2020920308.

[9] F. Idrees and M. Rajarajan, "Investigating the android intents and permissions for malware detection," *International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 354–358, 2014, doi: 10.1109/WiMOB.2014.6962194.

[10] D. Hindarto, "Perbandingan Kinerja Akurasi Klasifikasi K-NN, NB dan DT pada APK Android," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 9, no. 1, pp. 486–503, 2022, doi: 10.35957/jatisi.v9i1.1542.

[11] S. Alsoghyer and I. Almomani, "On the Effectiveness of Application Permissions for Android Ransomware Detection," *Proceedings - 2020 6th Conference on Data Science and Machine Learning Applications, CDMA 2020*, no. c, pp. 94–99, 2020, doi: 10.1109/CDMA47397.2020.00022.

[12] O. Campesato, "Artificial Intelligence Machine Learning and Deep Learning." 2020.

[13] R. Aryanto, M. A. Rosid, S. Busono, P. S. Informatika, and U. M. Sidoarjo, "Jurnal Informasi dan Teknologi Penerapan Deep Learning untuk Pengenalan Tulisan Tangan Bahasa Akasara Lota," vol. 5, no. 1, pp. 258–264, 2023, doi: 10.37034/jidt.v5i1.313.

[14] K. P. Danukusumo, "Convolutional neural network untuk mendeteksi bangunan," vol. 1, no. 1, pp. 10–12, 2017.

[15] W. C. Lin and Y. R. Yeh, "Efficient Malware Classification by Binary Sequences with One-Dimensional Convolutional Neural Networks," *Mathematics*, vol. 10, no. 4, pp. 1–14, 2022, doi: 10.3390/math10040608.

[16] A. Sharma, P. Malacaria, and M. H. R. Khouzani, "Malware detection using 1-dimensional convolutional neural networks," *Proceedings - 4th IEEE European Symposium on Security and Privacy Workshops, EUROS and PW 2019*, pp. 247–256, 2019, doi: 10.1109/EuroSPW.2019.00034.

[17] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mech Syst Signal Process*, vol. 151, p. 107398, 2021, doi: 10.1016/j.ymssp.2020.107398.