p-issn : 2338-4697
e-issn : 2579-3322

Pemanfaatan Sequential Convolutional Neural Networks pada Deteksi Malware Android

Winarnie
Teknologi Rekayasa Perangkat Lunak Politeknik Sawungggalih Aji
Jl. Wismoaji no. 8 Kutoarjo, Purworejo
Email: winzahwa@gmail.com

Abstrak

Sistem operasi *mobile* yang paling banyak digunakan sekarang ini adalah Android . Kesuksesan android ini juga berdampak pada system keamanan data yang terancan dengan timbulnya penyebaran malware pada *platform* Android. Ada beberapa aplikasi yang menggunakan android pada google play store yang terkena *malware*. *Malware* ini dapat secara sembunyi-sembunyi membuat ponsel korbannya berlangganan dan membayar konten premium tanpa sepengetahuan korban. Penelitian deteksi malware Android ini sangat penting untuk menjaga keamanan dan privasi pengguna. karena proses identifikasi malware yang semakin rumit, maka diperlukan pendekatan *deep learning* untuk klasifikasi *malware*. Penelitian ini menggunakan metode *sequential* models untuk mendeteksi malware android. Hasil pengujian didapat hasil akurasi : 99.07%, presisi : 99.06%, recall : 98.32%, dan f1 score : 98.69%.

Keywords: *Sequential* CNN, deteksi *malware*, *deep learning*, android

1. Pendahuluan

Android merupakan sistem operasi mobile yang paling banyak digunakan pada saat ini.. Hal ini yang menjadikannya sistem operasi seluler yang paling banyak digunakan bahkan melampaui sistem operasi Windows. Dampak negatif dari berkembangnya operasi android adalah masalah malware. Sistem Android adalah target utama malware seluler karena sistem operasi Android memungkinkan pengguna untuk menginstal aplikasi yang diunduh dari pasar pihak ketiga. Bahkan dengan berbagai mekanisme proteksi keamanan seperti play protection, masih terdapat berbagai laporan keberadaan malware pada Google Play Store. Malware ini dapat secara sembunyisembunyi membuat ponsel korbannya berlangganan membayar layanan premium tanpa sepengetahuan korban. Berdasarkan berbagai kejadian tersebut, masih perlu penelitian lebih lanjut tentang untuk identifikasi malware Android. Pendekatan yang berbeda telah disarankan dalam berbagai penelitian sebelumnya dengan maksud untuk menemukan malware. Pada beberapa tahun terakhir, Machine Learning (ML) mulai digunakan untuk mengembangkan sistem cerdas dengan cara melatih mesin untuk membuat keputusan. Dengan menggunakan dataset sebagai input, ML dapat mengidentifikasi data baru yang memiliki kemiripan pola dengan dataset inputan. Terdapat berbagai algoritma klasifikasi ML dapat digunakan untuk membangun model ML, dan masing-masing algoritma memiliki kelebihan tergantung pada dataset yang digunakan.

Beberapa penelitian yang telah dilakukan diantaranya, penelitian yang mengamati tren yang berkembang dari berbagai arsitektur jaringan saraf diadopsi untuk representasi dan karakterisasi malware Android yang sesuai dengan aslinya untuk menangkap pola semantik intrinsik malware Android untuk meningkatkan kinerja tugas deteksi atau klasifikasi [1]. Penelitian yang menggunakan dataset dunia nyata yang disebut CICAndMal2017. Untuk ekstraksi fitur, menggunakan alat CICFlowMeter dan menggunakan lalu lintas jaringan. untuk mengidentifikasi malware Android. Selain itu, mengevaluasi dataset dengan dua model pembelajaran mendalam: CNN, CNN-LSTM. Terakhir, membandingkan hasil dengan algoritma lain dalam literatur. Hasil eksperimen menunjukkan bahwa CNN-LSTM memiliki ketepatan tertinggi pada malware klasifikasi biner [2].

Percobaan selanjutnya mengusulkan cara untuk mendeteksinya dengan jaringan syaraf tiruan konvolusi. Dalam pekerjaan pra-pemrosesan, file paket Android dibedah menjadi kode-kode operasi Dalvik [3]. Selanjutnya Dengan menggunakan model canggih BERT, menunjukkan bahwa mungkin untuk mencapai

p-issn : 2338-4697
e-issn : 2579-3322

kinerja deteksi malware yang diinginkan dengan dataset yang sangat tidak seimbang. Kami menemukan bahwa model berbasis BERT kami mencapai skor F1 sebesar 0,919 dengan hanya 0,5% dari contoh yang malware, yang secara signifikan mengungguli keadaan saat ini [4].

Berikutnya menggabungkan fitur analisis statis dan dinamis dari aplikasi malware dan aplikasi bukan malware. Fitur dinamis diambil dari panggilan API pada aplikasi sedangkan fitur statis didapatkan melalui permission, system call dan intent [5]. Model Athiwaratkun untuk memasukkan dua parameter input yang paling penting dari setiap panggilan API system parameter input. kemudian menunjukkan bahwa model yang diusulkan mendominasi model yang diusulkan sebelumnya dalam hal operasi penerima kurva karakteristik operasi penerima (ROC) [6].

Penelitian untuk analisis dinamis aplikasi Android: MLDroid menggunakan teknik analisis dinamis untuk mempelajari perilaku aplikasi Android secara langsung. Tujuannya adalah untuk menganalisis dan memahami bagaimana aplikasi tersebut beroperasi, termasuk aktivitas yang mencurigakan atau berpotensi berbahaya yang dapat mengindikasikan keberadaan malware [7]. Penelitaian yang mengevaluasi menggunakan metrik performa: Penelitian ini menggunakan metrik performa seperti akurasi, recall, f- score, dan presisi untuk mengevaluasi kinerja DeepAMD. Tujuannya adalah untuk memperoleh pemahaman yang komprehensif tentang sejauh mana DeepAMD dapat secara efektif mendeteksi dan mengidentifikasi malware Android [8]. Penelitian berikutnya melakukan evaluasi dari berbagai aspek untuk mengukur keefektifan dan efisiensi sistem deteksi malware yang diusulkan. Evaluasi dilakukan terhadap akurasi deteksi berbagai jenis jaringan saraf rekuren (RNN), performa ekstraksi fitur pada berbagai perangkat seluler, dan akurasi deteksi serta waktu prediksi dengan berbagai panjang urutan. Tujuannya adalah untuk menunjukkan kemampuan sistem dalam mendeteksi malware dengan akurasi tinggi dan mengevaluasi kinerja dalam kondisi berbeda [9]

Mengidentifikasi tantangan dan hambatan dalam penerapan deep learning untuk deteksi malware Android. Penelitian ini juga bertujuan untuk mengidentifikasi tantangan dan hambatan yang dihadapi oleh peneliti dan praktisi dalam menggunakan deep learning untuk deteksi malware Android. Hal ini mencakup aspek seperti pemilihan arsitektur deep learning yang tepat, ekstraksi dan pemrosesan fitur, evaluasi kinerja, dan ketersediaan data yang memadai. [10]. Membuat profil Hidden Markov Model (HMM) berdasarkan pola encoded: Penelitian ini bertujuan untuk membuat profil Hidden Markov Model (HMM) berdasarkan pola encoded dari kelas/metode API yang mencurigakan dalam dataset malware Android [11].

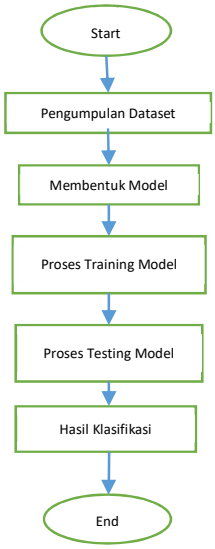
Dari studi literature yang telah dilakukan maka pada percobaain ini Algoritma ML yang digunakan dalam penelitian ini adalah Sequential Convolutional Neural yang menggunakan library keras dari python. Percobaan ini diawali dengan mengatur aksitektur jaringan saraf yang memiliki 3 lapisan yaitu :

1. Hidden layer 1 yang terdiri dari 32 neuron, aktivasi ReLU
2. Hidden layer 1I yang terdiri dari 32 neuron, aktivasi ReLU
3. Output layer, terdiri dari aktivasi Sigmoid.

Dari ketiga arsitektur tersebut diimplemtasikan pada library keras dengan menggunakan model sequential untuk menjelaskan lapisan – lapisan yang digunakan. Percobaan ini dilakukan dengan efektif dan sederhana tetapi memiliki hasil performa model yang baik untuk mendeteksi *malware* android.

2. Metode Penelitian

Pada penelitian menerapkan metode *Sequential models* dan langkah-langkah yang digunakan seperti pada gambar 1.



Gambar 1. Langkah Metode Penelitian

p-issn : 2338-4697
e-issn : 2579-3322

Proses awal dengan mengumpulkan dataset, kemudian membentuk model, selanjutnya melakukan training dan testing sehingga menghasilkan deteksi android *malware*.

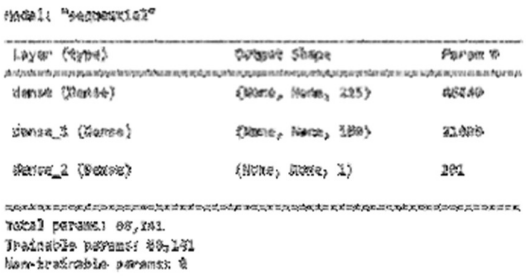
A. Pengumpulan Data

Deteksi malware dengan menggunakan atribut yang diekstraksi dari aplikasi Android sebagai fitur. Penelitian ini menggunakan dataset Android Malware Dataset for Machine Learning dari kaggle yang terdiri dari vektor fitur dari 215 atribut yang diekstraksi dari 15.036 aplikasi (5.560 aplikasi malware dari proyek Drebin dan 9.476 aplikasi jinak). Dataset telah digunakan untuk mengembangkan dan mengevaluasi pendekatan fusi pengklasifikasi bertingkat untuk deteksi malware Android, diterbitkan dalam IEEE Transactions on Cybernetics paper 'DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. File pendukung berisi deskripsi vektor/atribut fitur yang diperoleh melalui analisis kode statis aplikasi Android.. Perancangan dan pemodelan system menggunakan bahasa pemrograman python.

B. Membentuk Model

Sebelum dilakukan pemodelan, data dibagi menjadi data training, validasi dan data test menggunakan sample data secara acak. Data training sebanyak 80%, data validasi sebanyak 20% dan data test sebanyak 15%. Kemudian dilakukan proses preprosessing data dengan menggunakan tool keras pada bahasa pemrograman python.

Kami memilih lapisan yang terhubung dengan rapat sebagai model prediktif. Jenis saraf jaringan adalah tumpukan lapisan padat, di mana neuron dari setiap lapisan terhubung ke setiap neuron lain di lapisan berikutnya. Melatih jaringan yang terhubung dengan padat adalah proses yang sangat iteratif. Ini termasuk keputusan tentang jumlah dan ukuran lapisan, pilihan fungsi aktivasi dalam lapisan, fungsi kerugian, dan jumlah zaman. Tujuannya adalah untuk menemukan parameter dan hiper parameter ini yang mengoptimalkan fungsi kerugian pada sampel pelatihan dan pengujian. Untuk mencapai tujuan ini, kami merancang jaringan multilayer pertama dan kemudian mengevaluasinya secara empiris kinerja menggunakan set (5031 Malware, 10000 Goodware) . Kami memulai eksperimen kami dengan arsitektur : (164,)x(12, Relu)x(8, Relu)x (1, sigmoid). Lapisan tersembunyi pertama terdiri dari 12 neuron dengan Relu berfungsi sebagai fungsi aktivasi, lapisan kedua memiliki 256 neuron dan menggunakan fungsi aktivasi yang sama. Lapisan terakhir adalah lapisan keluaran dan menggunakan sigmoid berfungsi sebagai fungsi aktivasi. Algoritma yang dipilih untuk optimasi adalah Adam, alasan di balik pilihan ini adalah kesederhanaan dan efisiensinya dalam hal perhitungan. Bobot diinisialisasi ke angka yang dihasilkan secara acak antara 0 dan 0,05. Parameter yang diuji pada *Convolution Neural Network* terdiri dari jenis konvolusi, dan jumlah lapisan dense [12]. Struktur parameter dari Sequential Convolutional Neural dapat dilihat pada gambar 2.



Gambar 2. Struktur Parameter Sequential Convolutional Neural

C. Proses Training Model

Tahap pelatihan model menggunakan data sebanyak 12024 data atau 80% dari total dataset. Training model merupakan proses bekerjanya machine learning sehingga algoritma yang kita disain bisa mengingat pola tiap klas pada data yang kita latih. Model Sequential Convolutional Neural menggunakan tool augmentasi keras yang tersedia pada bahasa pemrograman python..

Setelah proses training selanjutnya dengan melakukan epoch sebanyak 5 kali. Proses epoch dapat dilihat pada gambar 3 dengan menggunkan tool yang tersedia pada python.

```
history = model.fit(train_x,
                    train_y,
                    validation_data = (test_x,test_y),
                    epochs = ep)
```

Gambar 3. Proses Training dengan Epoch

p-issn : 2338-4697
e-issn : 2579-3322

D. Proses Validasi Model

Proses evaluasi model menggunakan data testing sebanyak 383 data 15% dari data keseluruhan. Proses evaluasi dapat dilihat pada gambar 6 yang memanfaatkan tool dari python

```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.plot([str(i) for i in range(1,ep+1)],history.history['accuracy'],label="Train Accuracy")
plt.plot([str(i) for i in range(1,ep+1)],history.history['val_accuracy'],label="Validation Accuracy")
plt.legend()
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Epoch vs Train Loss")

plt.subplot(1,2,2)
plt.plot([str(i) for i in range(1,ep+1)],history.history['loss'],label="Train Loss")
plt.plot([str(i) for i in range(1,ep+1)],history.history['val_loss'],label="Validation Loss")
plt.legend()
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Epoch vs Validation loss")
```

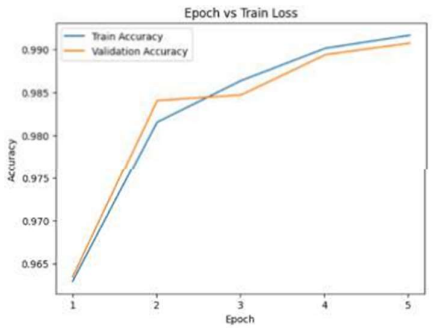
Gambar 4. Proses Validasii Model

3. HASIL DAN PEMBAHASAN

A. Data Training

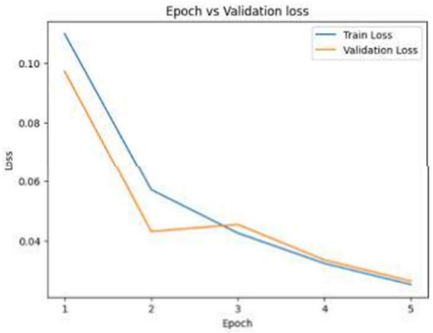
Setelah data terkumpul, kami menggunakan alat bantu Google Colab untuk memproses data. Pada tahap prapemrosesan data, kami mendapatkan 5 missing value pada kumpulan data yang dikumpulkan. Nilai yang hilang ini terutama disebabkan oleh data yang rusak saat mengumpulkan data dalam lingkungan berbasis kerangka kerja. Hasil tersebut menyisakan 15031 aplikasi untuk diproses ke tahap selanjutnya. Total ini terdiri dari 10.000 aplikasi jinak dan 5.031 malware. Sebelum memasuki tahap membagi data, kami membagi data 80% dan 20% untuk training dan validasi

Perbandingan nilai akurasi epoch dapat dilihat pada gambar 5.



Gambar 5. Grafik akurasi Epoch

Dari grafik gambar 5 dari hasil 5 kali epoch didapat nilai akurasi traing sebesar 0.9923 dan nilai akurasi validasi sebesar 0.9097. Perbandingan nilai loss epoch dapat dilihat pada gambar grafik 6. Hasil dari train loss sebesar 0.0248 dan validasi loss sebesar 0.0279.



Gambar 6. Grafik Loss Epoch

B. Hasil Perhitungan

Hasil dari perhitungan akurasi dari percobaan pada penetlitian ini menggunakan 2 metode yaitu confosius matriks dan nilai eproch sebanyak 5 kali. Hasil yang didapat dari confosius matriks dapat dilihat pada gambar 7 dan hasil epoch sebanyak 5 kali dapat dilihat pada tabe1 1 dan tabel 2.

```
from sklearn.metrics import classification_report

print(classification_report(y_pred,test_y))
```

```
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99      1927
         1.0       0.99      0.98      0.99      1980

    accuracy                           0.99      3907
   macro avg       0.99      0.99      0.99      3907
weighted avg       0.99      0.99      0.99      3907
```

Gambar 7. Confosius Matriks

Tabel 1. Hasil Epoch Training Accuraracy dan Loss

Epoch	Accuracy	Loss
1	0.9643	0.1066
2	0.9815	0.0576
3	0.9870	0.0423
4	0.9903	0.0325
5	0.9923	0.0248

p-issn : 2338-4697
e-issn : 2579-3322

Tabel 2. Validacy Accurasy dan Loss

Epoch	Accuracy	Loss
1	0.9618	0.1039
2	0.9854	0.0410
3	0.9857	0.0430
4	0.9864	0.0347
5	0.9897	0.0279

Hasil dari confosius matriks dapat dilihat nilai akurasi yang diperoleh nilai sebesar 99%, recall sebesar 98% . Dari hasil epoch sebanyak 5 kali diperoleh hasil akurasi terbesar diperoleh 99,23% , nilai loss 2,48%, val loss 2,79%, dan nilai val akurasi sebesar 98.97%.

4. KESIMPULAN

Dari hasil percobaan dari penelitian deteksi malware android dengan Sequential Convolutional Neural menggunakan dataset sebanyak 15031 maka dapat disimpulkan sebagai berikut :

1. Dataset berhasil dideteksi dengan nilai akurasi sebesar 99.23%, melalui proses epoch sebanyak 5 kali dan nilai loss sebesar 2.48%,.
2. Untuk nilai akurasi data validasi didapat nilai sakurasi ebesar 99% presisi sebesar 99%, recall sebesar 98%, dan f1 score sebesar 99%.
3. Metode Sequential Convolutional Neural cukup sederhana dalam prosesnya tetapi memiliki performa hasil yang sangat baik dalam mendeteksi malware android.

DAFTAR PUSTAKA

[1] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, “A Survey of Android Malware Detection with Deep Neural Models,” *ACM Comput. Surv.*, vol. 53, no. 6, 2021, doi: 10.1145/3417978.

[2] M. Gohari, S. Hashemi, and L. Abdi, “Android Malware Detection and Classification Based on Network Traffic Using Deep Learning,” *2021 7th Int. Conf. Web Res. ICWR 2021*, pp. 71–77, 2021, doi: 10.1109/ICWR51868.2021.9443025.

[3] X. Sun, J. Peng, H. Kang, and Y. Shen, “Android Malware Detection using Sequential Convolutional Neural Networks,” *J. Phys. Conf. Ser.*, vol. 1168, no. 6, 2019, doi: 10.1088/1742-6596/1168/6/062010.

[4] R. Oak, M. Du, D. Yan, H. Takawale, and I. Amit, “Malware detection on highly imbalanced data through sequence modeling,” *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 37–48, 2019, doi: 10.1145/3338501.3357374.

[5] 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.

[6] R. Agrawal, J. W. Stokes, M. Marinescu, and K. Selvaraj, “Neural Sequential Malware Detection with Parameters,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2018-April, pp. 2656–2660, 2018, doi: 10.1109/ICASSP.2018.8461583.

[7] A. Mahindru and A. L. Sangal, *MLDroid—framework for Android malware detection using machine learning techniques*, vol. 33, no. 10. Springer London, 2021. doi: 10.1007/s00521-020-05309-4.

[8] S. K. Sasidharan and C. Thomas, “ProDroid — An Android malware detection framework based on profile hidden Markov model,” *Pervasive Mob. Comput.*, vol. 72, p. 101336, 2021, doi: 10.1016/j.pmcj.2021.101336.

[9] R. Feng, J. Q. Lim, S. Chen, S. W. Lin, and Y. Liu, “SeqMobile: An Efficient Sequence-Based Malware Detection System Using RNN on Mobile Devices,” *Proc. IEEE Int. Conf. Eng. Complex Comput. Syst. ICECCS*, vol. 2020-October, pp. 63–72, 2020, doi: 10.1109/ICECCS51672.2020.00015.

[10] R. Feng, S. Chen, X. Xie, G. Meng, S. W. Lin, and Y. Liu, “A Performance-Sensitive Malware Detection System Using Deep Learning on Mobile Devices,” *IEEE Trans. Inf. Forensics Secur.*, vol. 16, no. XX, pp. 1563–1578, 2021, doi: 10.1109/TIFS.2020.3025436.

[11] S. I. Imtiaz, S. ur Rehman, A. R. Javed, Z. Jalil, X. Liu, and W. S. Alnumay, “DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network,” *Futur. Gener. Comput. Syst.*, vol. 115, pp. 844–

p-issn : 2338-4697
e-issn : 2579-3322

856, 2021, doi: 10.1016/j.future.2020.10.008.

[12] Y. Nan, J. Ju, Q. Hua, H. Zhang, and B. Wang, "A-MobileNet: An approach of facial expression recognition," *Alexandria Eng. J.*, vol. 61, no. 6, pp. 4435–4444, 2022, doi: 10.1016/j.aej.2021.09.066.