Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202072641
Vol. 7, No. 2, April 2020, hlm. 399-406 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

KLASIFIKASI APLIKASI ANDROID MENGGUNAKAN ALGORITME K-MEANS DAN CONVOLUTIONAL NEURAL NETWORK BERDASARKAN PERMISSION

Togu Novriansyah Turnip*[1], Pratiwi Okuli Manik[2], Jhon Harry Tampubolon[3], Patota Adi Petro Siahaan[4]

[1,2,3,4]Program Studi D3 Teknik Informatika, Fakultas Informatika dan Teknik Elektro, Institut Teknologi Del
Email: [1]togu@del.ac.id, [2]pratiwi24manik@gmail.com, [3]jhonarox22@gmail.com, [4]totapetro@gmail.com
*Penulis Korespondensi

(Naskah masuk: 31 Oktober 2019, diterima untuk diterbitkan: 11 Februari 2020)

Abstrak

Convolutional Neural Network (CNN) adalah salah satu metode *multilayer perceptron* yang dapat melakukan klasifikasi aplikasi lebih dari dua kelas. Penelitian ini mengklasifikasikan aplikasi ke dalam tiga kelas, yaitu kelas aplikasi tidak berbahaya, mengandung *malware* kurang berbahaya, dan mengandung *malware* berbahaya. Dataset yang digunakan pada penelitian ini terdiri dari *dataset* Androsec dan Koodous dengan total data 37289 aplikasi. *Dataset* mengandung aplikasi *undetected* (tidak mengandung *malware*) dan *detected* (mengandung *malware*). Data *detected* perlu dikelompokkan dengan algoritme *k-means* sehingga menghasilkan kelompok aplikasi kurang berbahaya dan berbahaya berdasarkan tingkat kemiripan fitur *permission* yang dimiliki aplikasi. Kerangka kerja meliputi *dataset preprocessing, learning and classification algorithm using CNN*, dan *check APK to Model*. Tingkat akurasi terbaik yang didapat pada penelitian ini adalah 92,23% dan dapat mengklasifikasikan ke dalam kelas tidak berbahaya, kurang berbahaya, dan berbahaya.

Kata kunci: *CNN, k-means clustering, permission, aplikasi android, malware*

ANDROID APP CLASSIFICATION USING K-MEANS AND CONVOLUTIONAL NEURAL NETWORK ALGORITHMS BASED ON PERMISSION

Abstract

Convolutional Neural Network (CNN) is a multilayer perceptron method which able to classify apps more than two classes. This paper describes classification into three classes such as benign/no malware, less harmful, and harmful application. In this research, we use and construct dataset from Androsec and Koodous with total 37289 apps. Dataset consists of undetected (no malware) and detected (consists of malware). Detected files need to clustered with k-means algorithm to clasify apps into less harmful and harmful based on apps permission similarity. The framework includes dataset preprocessing, learning and classification algorithm using CNN, and check APK to Model. In this research, we get the best accuracy 92,23% and able to classify apps into three classes benign, less harmful, and harmful.

Keywords: *CNN, k-means clustering, permission, android apps, malware*

1. PENDAHULUAN

Seiring dengan perkembangan teknologi di bidang komunikasi, Android menjadi salah satu sistem operasi teknologi unggul yang dapat digunakan pengguna dengan bebas. Android sebagai *open source software* yang dapat didisribusikan secara terbuka sehingga pengguna dapat membuat dan mengembangkan aplikasi baru secara bebas. Tetapi, terkadang ada pihak yang tidak bertanggung jawab dapat membangun dan mengembangkan *malware* pada sebuah aplikasi yang dapat masuk ke sistem Android dengan mudah (Novrianda, et al., 2014).

Klasifikasi aplikasi Android adalah salah satu teknik mengidentifikasi adanya *malware* pada aplikasi Android. Klasifikasi digunakan untuk mengklasifikasikan aplikasi sesuai dengan kelas aplikasi. Algoritme klasifikasi *machine learning* yang lebih dalam dapat memiliki *output* lebih dari dua kelas.

Penelitian terkait *malware detection* yakni Android *Malware Detection* and *Classification* (Abdi, 2015), algoritme *Support Vector Machine (SVM)* optimal mengklasifikasikan dua kelas, tetapi tidak optimal untuk mengklasifikasikan ke dalam lebih dari dua kelas. Metode yang dapat disarankan yaitu menggunakan *multilayer perceptron* yang

diklasifikasikan ke dalam tiga kelas yaitu aplikasi tanpa *malware*, mengandung *malware* kurang berbahaya dan mengandung *malware* berbahaya.

Metode yang menggunakan *Multilayer Perceptron* adalah *Convolutional Neural Network* (CNN) merupakan salah satu algoritme klasifikasi yang telah banyak dibahas pada penelitian Andro_MD: *Android Malware Detection based on Convolutional Neural Network* (Xie, et al., 2018) dengan tingkat akurasi yang lebih tinggi dibandingkan dengan tiga algoritme lainnya.

Algoritme CNN belajar memprediksi *output* berdasarkan data historis dengan memberikan label seperti pendekatan *supervised* pada umumnya (Brownlee, 2016). Data yang digunakan dalam penelitian berdasarkan *public* repositori yakni Androsec dan Koodous dengan data yang dihasilkan berbentuk file yang berbeda.

Teknik yang digunakan dalam proses *labelling* ini adalah dengan mengelompokkan data dalam satu *cluster* berdasarkan tingkat kemiripan yang maksimum yakni dengan metode *clustering* K-Means. Data berdasarkan repositori dibagi dalam dua bagian yakni *detected* dan *undetected* file. *Detected* file terdiri atas data yang tidak terdeteksi *malware* dan *undetected* file terdiri atas data yang terdeteksi *malware* berdasarkan sumber data.

Penelitian ini bertujuan untuk mengukur tingkat akurasi klasifikasi dalam tiga kelas tidak berbahaya, mengandung *malware* kurang berbahaya, dan mengandung *malware* berbahaya menggunakan algoritme *Convolutional Neural Network* dan *clustering K-Means* untuk membandingkan aplikasi mengandung *malware* kurang berbahaya dan mengandung *malware* berbahaya.

2. METODE PENELITIAN

2.1 Rancangan Penelitian

Jenis Penelitian yang digunakan pada penelitian ini yaitu penelitian terapan atau *applied research*. Karena penelitian ini berupa penerapan algoritma *K-means* dan *Convolutional Neural Network* dalam proses klasifikasi *malware* aplikasi Android dan evaluasi model yang dihasilkan pada eksperimen serta melakukan pengujian terhadap model.

2.2 Clustering K-Means

Untuk menentukan kemiripan data dapat digunakan pengukuran dengan *distance measure* (Anggara, et al., 2016). Metode *K-Means* merupakan metode *clustering* yang paling sederhana dan umum karena mempunyai kemampuan mengelompokkan data dalam jumlah yang cukup besar dengan waktu komputasi yang cepat dan efisien. Langkah-langkah pada K-Means:

1. Siapkan data *Detected* file

Tabel 1. Data APK untuk Clustering

Nama apk	Jumlah Permission			
	Normal	Signature	Dangerous	Miscellaneous
WebApp	4	2	3	3
Android Pay	7	1	5	0
XXX video	5	2	6	1
Browser Update	4	2	10	1
EClips	4	1	6	1
U-Mobile	4	1	3	1
Pikachu New	2	1	4	0
Zertifikat	3	0	3	0
VXPlayer	4	0	4	0
Medicine MRP	4	0	4	0

2. Menentukan *centroid* (K-Poin), nilai K= 2. Data *detected* pada Tabel 1 akan dikategorikan menjadi dua kelompok yaitu kategori mengandung *malware* kurang berbahaya dan mengandung *malware* berbahaya
3. Memilih nilai *centroid* secara random
4. Menghitung jarak setiap data ke masing-masing *centroid*. Dengan menggunakan *Euclidean Distance* seperti persamaan (1)

$$d_{ij} = \sqrt{\sum_{k=1}^{p}\{\mathcal{X}_{ik} - \mathcal{X}_{jk}\}^2} \quad (1)$$

Keterangan :
d_{ij} = *distance*
p = dimensi data
$\mathcal{X}_{ik}$ = posisi titik 1
$\mathcal{X}_{jk}$ = posisi titik 2

5. Menentukan *cluster* setiap item berdasarkan jarak
6. Menentukan *centroid* baru dengan mengkalkulasikan koordinat berdasarkan *centroid* baru yang sama pada iterasi pertama seperti persamaan (2)

$$C_k = \left(\frac{1}{n_k}\right)\sum d_1 \quad (2)$$

Keterangan :
C_k = *Centroid* baru
n_k = jumlah data dalam *cluster* k
d_1 = data dalam *cluster* k

7. Melakukan pengelompokan data kembali sehingga dihasilkan pengelompokan data matriks akhir sama dengan sebelumnya G2=G1.
8. Menentukan *cluster* setiap item berdasarkan jarak dengan *cluster* baru
9. Menentukan hasil akhir *cluster*.
$C_1 = (\text{WebApp}, \text{U} - \text{Mobile})$

$$C_2 = \begin{pmatrix} \text{Android Pay,} \\ \text{XXX video,} \\ \text{Browser Update,} \\ \text{EClips,} \\ \text{Pikachu New,} \\ \text{Zertifikat, VXPlayer} \\ \text{Medicine MRP} \end{pmatrix}$$

Berdasarkan jumlah *dangerous permission* dan banyaknya *permission* yang dimiliki aplikasi, *cluster* C_1 merupakan aplikasi kurang berbahaya dan *cluster* C_2 merupakan aplikasi berbahaya.

Tabel 2. Data Hasil Labelling

Nama Aplikasi	Label	*Permission (android.permission :a.p)*
Browser Update	Kurang Berbahaya	a.p..ACCESS_NETWORK_STATE a.p.INTERNET a.p.RECEIVE_BOOT_COMPLETED …
U-Mobile	Kurang Berbahaya	a.p.WRITE_EXTERNAL_STORAGE a.p.CHANGE_CONFIGURATION a.p.SYSTEM_ALERT_WINDOW …
Android Pay	Berbahaya	a.p.WRITE_EXTERNAL_STORAGE a.p.ACCESS_WIFI_STATE a.p.CHANGE_NETWORK_STATE ….
XXX video	Berbahaya	a.p.ACCESS_NETWORK_STATE a.p.INTERNET ….
Document	Tidak Berbahaya	a.p.MANAGE_DOCUMENTS
Call Management	Tidak Berbahaya	a.p.BIND_INCALL_SERVICE a.p.BLUETOOTH a.p.BLUETOOTH_ADMIN …

Berdarkan Tabel 2 data hasil *labelling*, data *undetected* diberi label tidak berbahaya, data *detected* hasil *clustering* telah diberi label kurang berbahaya dan berbahaya berdasarkan hasil *clustering*

2.3 Convolutional Neural Network (CNN)

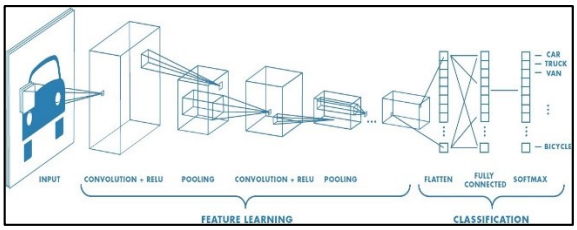
Convolutional Neural Network (CNN) merupakan pengembangan dari *Multilayer Perceptron* dalam mengolah data dua dimensi (Suartika E. P, et al., 2016). Pada konsep ini dilakukan penerimaan *input* data dari dua dimensi dan melakukan *propagation* pada *hidden layer* sehingga menghasilkan *output*.

Sebuah CNN memiliki puluhan hingga ratusan *layer* untuk mendeteksi data yang diberikan (Mathworks, 2019). Metode CNN dapat digunakan mengubah data *layer* per *layer* untuk dapat mengklasifikasikan data dimana *output* dari setiap *layer* menjadi *input* bagi *layer* selanjutnya, sehingga pada bagian ini proses iterasi dilakukan. Arsitektur CNN yang akan digunakan adalah CNN Lenet 5 yang meliputi *convolutional*, *max-pooling,* dan *fully-connected layer.*

Arsitektur yang terdapat pada modul CNN 2C 1D yang digunakan pada penelitian ini terdiri atas (Gilbert, 2016):

1. *Input layer of* NxN *pixels*
2. *Convolutional layer*
3. *Max-pooling layer.*
4. *Convolutional layer*
5. *Max-pooling layer*
6. *Fully-connected layer*
7. *Output layer*

Setiap *input* gambar akan melewati *convolution layer, pooling, fully connected* dan menerapkan fungsi *softmax* untuk mengklasifikasikan objek. Arsitektur CNN dapat dilihat pada Gambar 1.



Gambar 1. Arsitektur CNN Lenet

Penjelasan proses yang terjadi berdasarkan Gambar 1 sebagai berikut.

1. *Convolution Layer*

Convolution layer adalah *layer* pertama yang dimasuki oleh data *input*. Layer ini terdiri dari *neuron* yang tersusun membentuk sebuah *filter* dengan ukuran panjang dan tinggi *(pixel)*.

Setelah melakukan *convolution layer*, digunakan *activation function* yang berfungsi untuk menentukan *neuron* untuk aktif atau tidak. Karena menggunakan konsep *multilayer perceptron*, *activation function* yang digunakan adalah jenis *non-linear*. Salah satu *activation* yang dapat digunakan adalah ReLU. ReLU pada dasarnya melakukan *threshold* dari 0 (nol) hingga *infinity.*

2. *Pooling Layer*

Pooling layer terdiri dari sebuah *filter* yang memiliki ukuran dan *stride* tertentu yang akan bergeser pada seluruh area *feature map*. *Pooling* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Tujuan penggunaan *pooling* adalah mengurangi dimensi dari *feature map* (*downsampling*), sehingga mempercepat komputasi dan mengatasi *overfitting*.

3. *Fully Connected Layer* (FC-Layer)

Hasil *feature map* dari *pooling layer* masih berbentuk *multidimensional array*, sehingga perlu mengubah *feature map* menjadi sebuah *vector* dengan melakukan *flatten* agar dapat digunakan sebagai *input* pada *fully connected* layer. *FC layer* terdiri atas *multilayer perceptron* untuk proses klasifikasi data

yang telah dipelajari pada tahap *convolution* dan *pooling*. FC-*layer* memiliki *hidden layer, activation function, output layer,* dan *loss function.*

2.4 Deep Learning Framework (Library)

Penggunaan *library* ini untuk membangun suatu jaringan saraf atau *neural network*. Di dalam Tensorflow, *package* yang digunakan dalam *neural network* yaitu Keras (Keras, 2019). Keras adalah *library neural network* yang bersifat *high level*, dibuat dengan bahasa Python dan dapat dijalankan di atas *library* TensorFlow, CNTK, atau Theano sebagai *backend*-nya.

Pada dasarnya, TensorFlow sudah cukup untuk digunakan, tetapi kebutuhan peneliti untuk melakukan riset sering mencoba arsitektur berbeda, mencari *optimizer* yang paling cepat dan bagus, *tweaking hyperparameter*, dll. Untuk mengatasi hal tersebut, peneliti perlu menggunakan Keras karena peneliti dapat melakukan riset dengan relatif lebih cepat dari pada hanya menggunakan TensorFlow.

Hyperparameter yang dapat ditentukan diantaranya *epoch*, *learning rate*, *batch_size*, dll. Nilai ideal suatu *hyperparameter* belum dapat dipecahkan, karena tergantung pada data yang dibutuhkan diteliti. Salah satu *hyperparameter* yang umum digunakan pada *machine learning* adalah *epoch*. *Epoch* biasa digunakan saat melakukan *training* dan *testing dataset*. *Epoch* digunakan untuk pembelajaran jaringan saraf tiruan, salah satu metode *training* yaitu propagasi balik *(backpropagation). Epoch* (Tensorflow, 2019) secara umum didefinisikan sebagai proses satu kali melewati keseluruhan *dataset*.

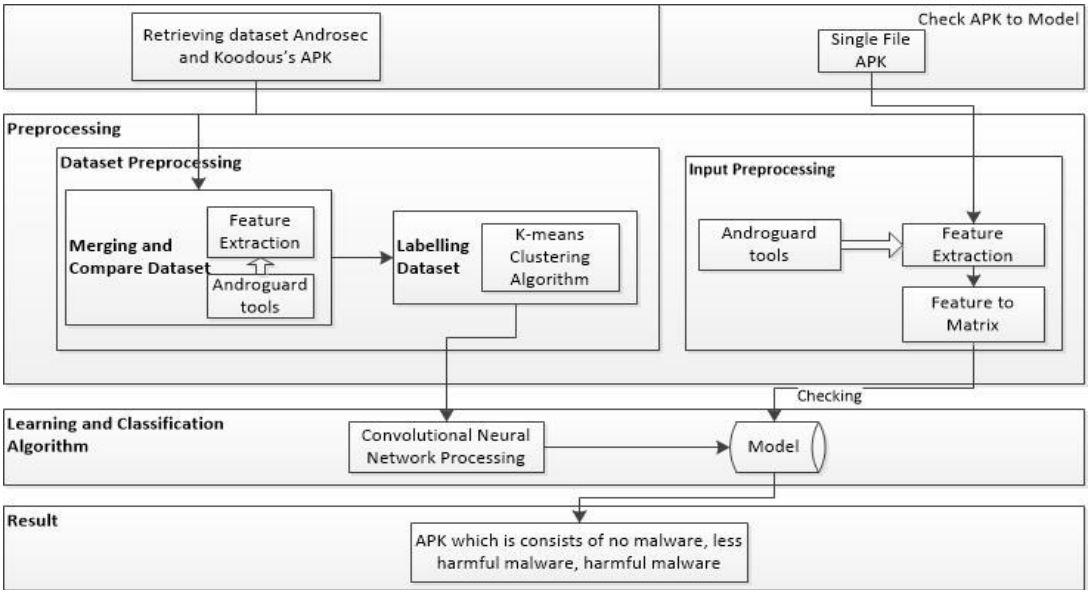
Epoch (Rahmadya, 2017) adalah proses *training* pada *Neural Network* untuk melakukan satu kali iterasi dengan melakukan rambatan balik. Misalnya, satu iterasi melibatkan a-b-c-d, maka satu *epoch* akan melibatkan a-b-c-d-c-b-a. Dua iterasi: a-b-c-d-a-b-c-d, sedangkan untuk dua epoch: a-b-c-d-c-b-a-b-c-d-c-b-a.

Saat menggunakan *validation split* (Keras, 2019), data dibagi ke dalam dua bagian untuk setiap *epoch* yaitu *training data* dan *validation data*. Proses *training* dan *validation* akan menghasilkan sebuah model yang berisikan nilai *weight* dan *bias neural network* yang disimpan dalam *file* .h5 hasil pembelajaran data. Proses ini akan melatih model pada *training data* dan memvalidasi model pada *validation data* dengan mengecek nilai *loss* dan *accuracy*. Evaluasi kinerja model penting dilakukan untuk mendapatkan dan memahami kualitas model yang dirancang, memperbaiki model, dan memastikan apakah model sudah tepat untuk melakukan prediksi atau klasifikasi (Oprea, 2014). Model yang dihasilkan akan digunakan untuk klasifikasi data.

Pada klasifikasi yang memiliki lebih dari dua kelas, indikator atau ukuran yang digunakan adalah *accuracy*. *Accuracy* adalah peluang bahwa data yang di label diberikan dengan benar (Elgamal, 2016). Rumus untuk menghitung *accuracy* dapat dilihat pada persamaan (3)

$$accuracy = \frac{N(correct\ classifications)}{N(all\ classifications)} \quad (3)$$

Keterangan:
N (correct classifications) = jumlah data yang diprediksi dan diklasifikasi secara tepat
N (all classifications) = jumlah semua data yang diklasifikasi



Gambar 2. Proses / Kerangka Kerja Sistem

3. PERANCANGAN SISTEM

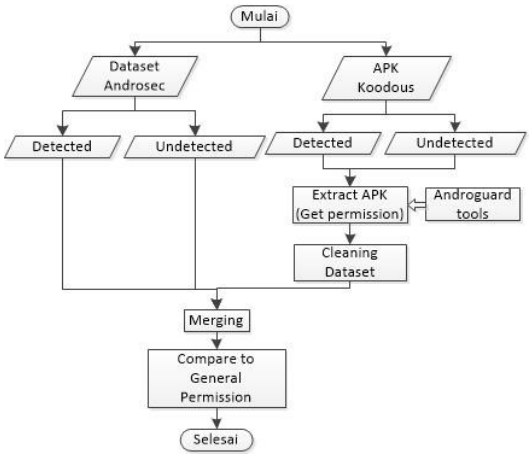
Secara umum, tahapan yang dilakukan pada penelitian ini dapat dilihat pada Gambar 2. Berikut merupakan proses-proses yang dilakukan selama melakukan penelitian:

a. Dataset Preprocessing

Preprocessing dilakukan dengan beberapa tahapan:

1. *Merging and Compare Dataset*

Pada proses ini dilakukan ekstraksi fitur menggunakan Androguard *tools* pada *dataset* Koodous untuk mendapatkan fitur *permission*, sedangkan Androsec sudah berisikan data fitur *permission* tanpa harus mengekstraknya terlebih dahulu.

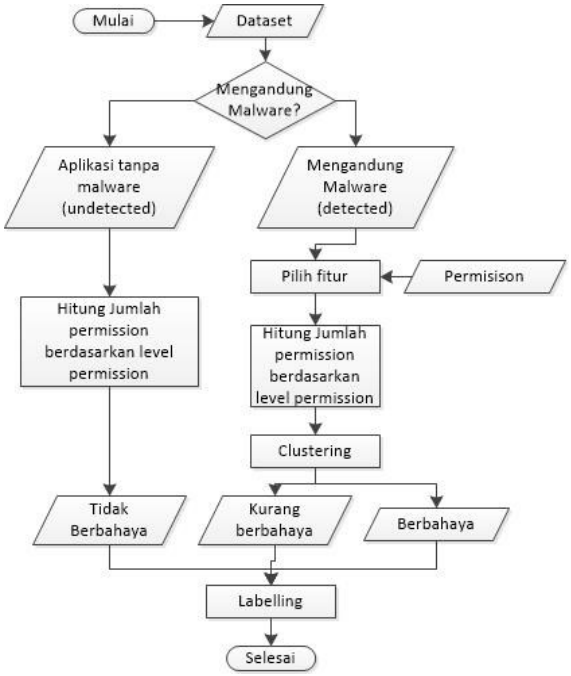


Gambar 3. Merging and Cleaning Dataset

Dataset selanjutnya di-*cleaning* untuk membersihkan *dataset* dari fitur kosong dan redundan. Kemudian, kedua *dataset* akan digabungkan menjadi satu *dataset,* seperti pada Gambar 3. Tahap selanjutnya, *permission dataset* akan dibandingkan terhadap *general permission* yang dimiliki oleh Android (Developers, 2019).

2. *Labelling Dataset*

Pada proses ini dilakukan pemberian label pada *dataset*, alur pada Gambar 4. Untuk data *undetected* diberikan label tidak berbahaya, sedangkan data *detected* perlu dibagi ke dalam dua bagian untuk membedakan aplikasi yang mengandung *malware* kurang berbahaya dan berbahaya. Untuk dapat membedakan dan menggolongkan data *undetected*, digunakan algoritme *k-means clustering. Clustering* dilakukan untuk mengidentifikasi aplikasi yang memiliki tingkat kemiripan maksimum/minimum.



Gambar 4. Labelling Dataset

Pengelompokan didasarkan pada jumlah *permission* pada *general permission* dan dibandingkan dengan level *permission* (Retenaga, 2015) normal, *signature, dangerous*, dan *miscellaneous*. Hasil *clustering detected* akan memberikan label mengandung *malware* kurang berbahaya dan berbahaya.

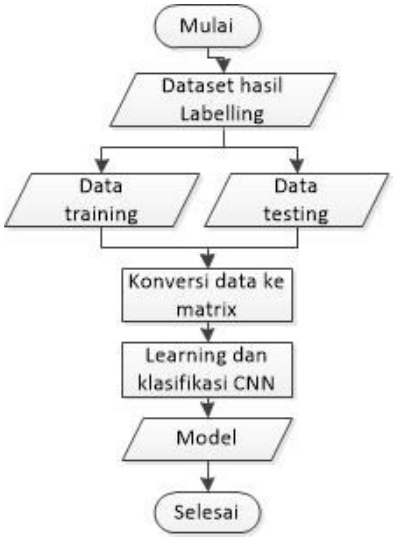
b. Learning and Classification Algorithm

Hasil *preprocessing* (*labelling*) dataset dibagi ke dalam dua bagian, yaitu 80% data *training* dan 20% data *testing* Gambar 5. *Dataset* dikonversi terlebih dahulu ke dalam bentuk matriks supaya dapat dijadikan sebagai *input* CNN. Berdasarkan jumlah *general permission* sebanyak 160, matriks yang dapat dibentuk berukuran 13x13. Pada tahap ini, dataset akan melewati fase *convolution layer, max-pooling*, dan *fully connected* layer.

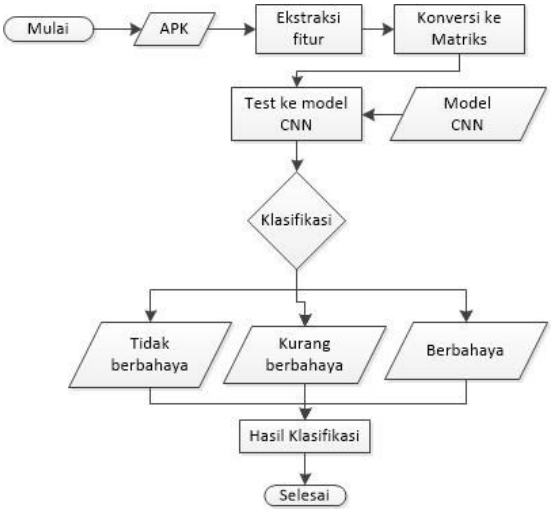
Learning dan *testing* bertujuan untuk mempelajari pola dari *dataset* sehingga menghasilkan satu model yang digunakan untuk klasifikasi aplikasi. Klasifikasi akan menghasilkan tiga kelas tidak berbahaya, kurang berbahaya, dan berbahaya. *Hyperparameter* yang digunakan dalam penelitian adalah jumlah *epoch*. Penggunaan *epoch* memungkinkan perulangan *learning* dan klasifikasi sesuai nilai maksimum perulangan yang ditentukan.

c. Test APK to Model

Pengecekan APK dilakukan untuk deteksi dan klasifikasi. Tahap ini, Gambar 6 memerlukan *single file* APK untuk menjalankan proses *input preprocessing* dengan melakukan ekstraksi aplikasi dan mengonversi fitur aplikasi ke dalam matriks.



Gambar 5. Proses Algoritme CNN



Gambar 6. Test APK Terhadap Model dan Klasifikasi APK

Selanjutnya, aplikasi akan di-*test* atau dibandingkan dengan model yang telah ada pada saat proses *learning* dan *testing*. Hasil akhir proses ini akan memberikan kelas mana yang dimiliki aplikasi, apakah kelas tidak berbahaya, kurang berbahaya, atau berbahaya.

4. HASIL DAN PEMBAHASAN

Adapun hasil dan pembahasan dari penelitian ini adalah:

A. Hasil pengujian jumlah *epoch*

Tahap *learning* dan *testing* menggunakan *hyperparameter* jumlah *epoch* untuk membuat dan mendapatkan model terbaik.

Dari eksperimen yang dilakukan berdasarkan Tabel 3, nilai akurasi yang diperoleh cenderung berubah dan tidak stabil terhadap jumlah *epoch* yang berbeda. Pengujian 6 (P6) dengan jumlah *epoch* 3500 mendapatkan nilai akurasi terbaik sebesar 92,23% untuk *training accuracy* dan 92,10% untuk *validation accuracy*. Pada Pengujian 6 juga didapat nilai *loss* terendah sebesar 0,2217 dan *val_loss* 0,2408. Semakin kecil nilai *loss*, berarti kehilangan data selama *training* sedikit, dan model yang dibentuk akan semakin baik.

Tabel 3. Hasil Pengujian Eksperimen Jumlah Epoch

ID	*Epoch*	*loss*	*acc*	*val_loss*	*val_acc*
P1	1000	0.4318	0.8998	0.4363	0.8963
P2	1500	0.2725	0.8773	0.2904	0.8724
P3	2000	0.4019	0.9074	0.4940	0.8908
P4	2500	0.4279	0.9047	0.4463	0.8988
P5	3000	0.3969	0.9102	0.4788	0.8946
P6	3500	0.2217	0.9223	0.2408	0.9210
P7	4000	0.4305	0.9095	0.4055	0.9041
P8	4500	0.4351	0.8965	0.4650	0.8906
P9	5000	0.5019	0.9048	0.5546	0.8986
P10	5500	0.4273	0.9029	0.4729	0.8936

Akurasi terendah dimiliki oleh Pengujian 2 dengan jumlah *epoch* 1500 dengan *acc* 0,8773 dan *val_acc* 0,8724. Pada pengujian ini terjadi penurunan akurasi terendah dari jumlah *epoch* 1000. Hal ini menunjukkan sistem belum cukup baik untuk melakukan pembelajaran terhadap data *training* sehingga sistem juga belum dapat memprediksi data *testing* dengan baik.

Melalui pengujian yang dilakukan, maka digunakan model yang memiliki akurasi terbaik dalam proses *learning* dan klasifikasi. Model yang tercipta akan digunakan untuk melakukan pengecekan dan klasifikasi APK. Dari pengujian eksperimen ini didapatkan hasil bahwa jumlah *epoch* yang tinggi tidak menentukan nilai akurasi yang lebih baik, tetapi ada jumlah *epoch* ideal untuk mendapatkan nilai akurasi terbaik.

B. Evaluasi Eksperimen

Dalam penelitian ini, fitur data statis aplikasi yang digunakan adalah *permission*. Pada saat proses *clustering*, *dataset* yang di-*cluster* hanya aplikasi yang *detected* atau mengandung *malware*. Hal ini dilakukan karena kedua kategori aplikasi *detected* dan *undetected* memiliki fitur *permission* yang sama. Dari sejumlah *dataset* yang dimiliki, ada *dataset undetected* yang memiliki *permission* yang sama dengan *detected*.

Apabila *detected* dan *undetected* di-*cluster* secara bersamaan, maka akan ada aplikasi *undetected* yang akan digolongkan ke dalam *detected* dan aplikasi *detected* digolongkan ke dalam *undetected*. Dengan adanya kondisi ini, sehingga peneliti menggunakan *clustering* hanya pada *detected* untuk membedakan aplikasi yang mengandung *malware* kurang berbahaya dan *malware* berbahaya. Hasil penggolongan ini akan dilabeli. Untuk aplikasi

undetected, peneliti melabeli langsung ke dalam tidak berbahaya.

Hasil *clustering* terhadap *detected* masih diteruskan ke tahap *learning* menggunakan algoritme CNN. Dalam proses ini, setiap pola aplikasi yang telah dilabeli akan dipelajari dan diklasifikasi ke dalam tiga kelas tanpa *malware*/tidak berbahaya, mengandung *malware* kurang berbahaya, dan mengandung *malware* berbahaya. Pada tahap ini, memungkinkan algoritme mempelajari *detected* dan *undetected* dengan pola yang sama sehingga dapat mengurangi akurasi klasifikasi dan kualitas model yang dibangun yang dapat menimbulkan ketidaktepatan penggolongan jenis aplikasi.

Oleh karena itu, fitur *permission* tidak cukup digunakan sebagai data utama dalam proses penelitian ini. Diperlukan jenis data statis atau dinamis lainnya agar dapat meningkatkan kualitas model untuk dapat mengklasifikasikan aplikasi.

5. KESIMPULAN

Melalui penelitian yang telah dilakukan, didapatkan kesimpulan:

1. Algoritme *K-Means* dapat meng-*cluster* aplikasi Android ke dalam lebih dari satu kelas berdasarkan tingkat kemiripan fitur aplikasi yang dimiliki.
2. Algoritme *K-Means* dan *Convolutional Neural Network* dapat mengklasifikasikan aplikasi Android ke dalam kelas tanpa *malware*/tidak berbahaya, mengandung *malware* kurang berbahaya, dan mengandung *malware* berbahaya, dengan akurasi model terbaik mencapai 92,23%.
3. Nilai akurasi model terbaik tidak dipengaruhi oleh banyaknya jumlah *hyperparameter epoch*. Namun, ada jumlah *epoch* ideal untuk menghasilkan model terbaik. Pada penelitian ini, jumlah *epoch* ideal adalah sebanyak 3500.

UCAPAN TERIMA KASIH

Terima kasih kepada LPPM Institut Teknologi Del yang memberikan dana penyelenggaraan dalam melaksanakan penilitian ini.

DAFTAR PUSTAKA

ABDI, M., 2015. Android Malware Detection And Classification. pp. 1-9.

ANGGARA, M., SUJIANI, H. & NASUTION, H., 2016. Pemilihan Distance Measure Pada K-Means Clustering Untuk Pengelompokan Member Di Alvaro Fitness. *Jurnal Sistem dan Teknologi Informasi (JUSTIN),* Volume 1, pp. 1-6.

BROWNLEE, J., 2016. *Understand Machine Learning Algorithms.* [Online] Available at: https://machinelearningmastery.com [Diakses 29 November 2018].

DEVELOPERS, G., 2019. *Developers.* [Online] Available at: https://developer.android.com/guide/topics/permissions/overview#permission-groups [Diakses 25 March 2019].

ELGAMAL, M., 2016. Sentiment Analysis Methodology of Twitter Data with an application on Hajj season. *International Journal of Engineering Research & Science (IJOER),* 2(1), pp. 82-87.

GILBERT, D., 2016. *Convolutional Neural Networks for Malware Classification.* Barcelona: Escola Politècnica de Catalunya (UPC) - BarcelonaTech.

KERAS, 2019. *Keras Documentation.* [Online] Available at: https://keras.io/ [Diakses 13 May 2019].

MATHWORKS, 2019. *MathWorks Deep Learning.* [Online] Available at: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html [Diakses 15 May 2019].

NOVRIANDA, R., KUNANG, Y. N. & P, S. H., 2014. Analisis Forensik Malware Pada Platform Android. *Konferensi Nasional Ilmu Komputer,* p. 377.

OPREA, C., 2014. PERFORMANCE EVALUATION OF THE DATA MINING CLASSIFICATION METHODS. *Annals of the „Constantin Brâncu şi" University of Târgu Jiu, Economy Series,* pp. 249-253.

RAHMADYA, 2017. *Rahmadya - "Just for a little kindness".* [Online] Available at: https://rahmadya.com [Diakses 13 May 2019].

RETENAGA, A. M., 2015. *Android Malware Situation.* Spanyol: INCIBE.

SUARTIKA E. P, I. W., WIJAYA, A. Y. & SOELAIMAN, R., 2016. Klasifikamsi Citra Menggunakan Convolutional Neural Network(CNN). *Jurnal Teknik ITS,* Volume 5, pp. A65-A69.

TENSORFLOW, 2019. *TensorFlow.* [Online] Available at: https://www.tensorflow.org/ [Diakses 13 May 2019].

XIE, N., DI, X., WANG, X. & ZHAO, J., 2018. Andro_MD : Android Malware Detection based on Convolutional Neural Networks. *International Journal of Performability Engineering,* Volume 14, pp. 547-558.

Halaman ini sengaja dikosongkan