Deteksi Malware Android Menggunakan Pengklasifikasi Pembelajaran Mesin Paralel

1st Mukhamad Rafi Galih Saputro
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
masterzrafly@student.telkomuniversity.ac.id

2nd Setyorini
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
setyorini@telkomuniversity.ac.id

3rd Siti Amatullah Karimah
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
karimahsiti@telkomuniversity.ac.id

***Abstrak* — Perkembangan android semakin pesat, sehingga mendorong pertumbuhan malware android. Data malware android memiliki dimensi tinggi, dibutuhkan algoritme untuk melakukan deteksi. Support Vector Machine (SVM) adalah algoritme pembelajaran mesin yang cocok untuk data malware android. Namun SVM memiliki keterbatasan dari segi waktu komputasi untuk data dalam jumlah besar, karena membutuhkan solusi dari masalah pengoptimalan Quadratic Programming (QP). Penelitian ini mengusulkan Parallel Support Vector Machine (PSVM) dengan metode dekomposisi Sequential Minimal Optimization (SMO) untuk melakukan deteksi atau klasifikasi malware android menggunakan dataset DREBIN. algoritme SMO yang dijamin memecahkan QP, dengan menggunakan teknik dekomposisi yaitu mendistribusikan tugas ke beberapa prosesor untuk dieksekusi secara paralel. Evaluasi berdasarkan kinerja perbandingan model Parallel SVM-SMO 4 dekomposisi dan Non-Parallel SVM-SMO dengan analisis fitur menggunakan Correlation Coefficient. Pada pengujian metrik performa dan akurasi fitur paling optimal 14 fitur dengan rata-rata akurasi 78%. Pada pengujian kinerja model, fitur paling optimal 27 fitur dengan rata-rata percepatan 9.58 dan efisiensi 2.39 pada kernel linier.**

***Kata Kunci*— DREBIN, SVM, PSVM, SMO, Dekomposisi, Koefisien Korelasi.**

I. PENDAHULUAN

A. Latar Belakang

Di era Revolusi Industri 4.0, teknologi digital dan internet seperti kecerdasan buatan semakin berkembang. terutama pada sistem keamanan perangkat [1]. Android telah menjadi sistem operasi perangkat seluler paling populer dalam beberapa tahun terakhir, dengan pangsa pasar global sekitar 71,96% (November 2022) [2]. Dengan demikian, jumlah pengguna smartphone terus meningkat dan pertumbuhan pengguna aplikasi Android juga meningkat pesat. Seiring bertambahnya jumlah pengguna aplikasi Android, jumlah dan kompleksitas data malware Android juga terus meningkat [3], menimbulkan ancaman yang signifikan terhadap keamanan perangkat seluler dan layanan yang mereka sediakan. Sehingga semakin banyak peneliti yang tertarik menggunakan pembelajaran mesin untuk melakukan deteksi malware Android [4], [5].

Banyak ilmuwan telah mencoba-coba pembelajaran mesin selama bertahun-tahun untuk mendeteksi malware android, salah satunya adalah algoritme Support Vector Machine (SVM). Algoritme SVM adalah Supervised Learning yang sangat cocok untuk mendeteksi atau mengklasifikasikan data yang memiliki label kelas [6], [7] seperti data malware android [3]. Kumpulan data malware memiliki dimensi yang tinggi, dari beberapa kelebihan algoritme SVM, SVM juga sangat cocok digunakan pada data yang berdimensi tinggi [8] SVM memiliki konsep yang lebih komprehensif dan lebih eksplisit secara matematis dibandingkan dengan metode klasifikasi lainnya [7], [9]. Namun SVM memiliki keterbatasan dari segi waktu komputasi dan penggunaan memori untuk data dalam jumlah besar, data memiliki jumlah sampel yang besar yaitu lebih dari 10.000 sampel [10], dataset malware android proyek DREBIN memiliki jumlah sampel 15.036 [3], karena banyaknya vektor pelatihan sehingga membutuhkan solusi dari masalah pengoptimalan Quadratic Programming (QP), proses untuk memecahkan masalah optimisasi matematika tertentu yang melibatkan fungsi kuadrat [4], [11]. Studi ini mengusulkan penerapan klasifikasi pembelajaran mesin paralel untuk pemrosesan data skala besar untuk menghemat waktu komputasi [4], [12].

1. Topik dan Batasannya

Penelitian ini berfokus pada kinerja paralel dalam melakukan deteksi malware android, menerapkan metode klasifikasi algoritme SVM untuk mendeteksi malware android, dan menerapkan dekomposisi SMO untuk menguji konsep SVM paralel, berikut beberapa batasan pada penelitian ini:

Dalam penelitian ini dataset malware android yang dipakai berasal dari proyek DREBIN [3] yaitu kumpulan data vektor fitur, dari 215 atribut yang diektrak dari 15.036 aplikasi.

Melakukan analisis korelasi koefisien [13]–[15] pada pemilihan fitur untuk mendapatkan nilai fitur yang berkorelasi positif terhadap kelas.

Metode optimasi SMO [11] dengan dekomposisi untuk konsep paralel [4], [16].
Parameter pengujian yaitu jumlah fitur paling optimal dan dua kernel yaitu linier dan gaussian.

1. Tujuan

Terkait latar belakang didapatkan tujuan yang akan dicapai pada penelitian ini, sebagai berikut:
Membangun model deteksi malware menggunakan Non-Parallel SVM dan Parallel SVM dengan teknik SMO.
Menguji konsep paralel dengan teknik dekomposisi SMO (2 dekomposisi dan 4 dekomposisi).
Membandingkan hasil dan analisis untuk memperoleh kinerja terbaik pada model yang diusulkan berdasarkan parameter jumlah fitur dan kernel.

II. KAJIAN TEORI

A. Studi Terkait

1. Tinjauan Pustaka

Prasojoe et al [17] melakukan perbandingan antara FullSMO, SubsetSMO, dan JointSMO untuk mengklasifikasi dataset. Mereka mengusulkan menggunakan dekomposisi dataset untuk membagi data ke sejumlah subsets untuk melakukan klasifikasi secara paralel, penelitian ini juga menerapkan reduksi dimensi untuk mengurangi dimensi data yang tinggi. Pengujian skenario diterapkan pada dua kernel dan beberapa parameter. Hasil akurasi membuktikan JointSMO meningkatkan akurasi hingga rata rata mencapai 75% dan SpeedUp 5.7. Dari 4 dataset yang dipakai, kernel linier mendapat akurasi tertinggi yaitu mencapai 96%.

Deepa et al [15] studi ini melakukan survei terhadap metode klasifikasi malware android berdasarkan pemilihan seleksi fitur, berfokus pada pencarian fitur terbaik untuk analisis malware android. Model dibangun dengan sejumlah fitur 100, 200, 300, hingga 1000, menggunakan beberapa metode klasifikasi salah satunya adalah algoritme SMO. Hasil menunjukan algoritme SMO mengalami dua kali peningkatan akurasi beberapa mencapai 81% pada sejumlah fitur 600 hingga 1000 dengan metode seleksi fitur berdasarkan korelasi.

2. Sistem Operasi Android

Sistem operasi (OS) adalah perangkat lunak sistem yang mengelola sumber daya perangkat keras dan perangkat lunak komputer dan menyediakan layanan umum untuk program komputer [9], [18].

3. Malware Android

Mobile malware adalah perangkat lunak berbahaya yang menargetkan ponsel nirkabel atau PDA, menyebabkan sistem crash dan kehilangan atau kebocoran informasi sensitif [9], [19].

4. Machine Learning

a. Support Vector Machine (SVM)

Pada penelitian ini SVM digunakan sebagai metode klasifikasi dalam pemodelan. Proses pelatihan SVM lambat terutama untuk data dalam jumlah besar [4]. Salah satu alasan menjadi lambat adalah karena membutuhkan solusi dari masalah pengoptimalan Quadratic Programming (QP) yang sangat besar [4], [20]. SMO adalah salah satu pendekatan untuk menangani kumpulan data dalam jumlah besar [11], [16], [20].

b. Permrosesan Secara Paralel

Pemrosesan paralel berarti bahwa algoritme digunakan dalam banyak prosesor [4], [5], [21]. Paralel dapat meningkatkan kinerja SVM dalam hal waktu dan memori. Studi ini menerapkan metode dekomposisi untuk konsep komputasi paralel.

c. Dekomposisi

Implementasi dekomposisi paralel sangat populer untuk masalah SVM skala besar karena hanya menggunakan sebagian sampel dalam himpunan kerja di setiap iterasi dan mengabaikan yang lainnya [4], [20]. Dalam penelitian ini dekomposisi akan dilakukan dengan membagi dataset menjadi 4 subsets, karena menyesuaikan resource yang tersedia hanya terdapat 4 cores.

d. Sequential Minimal Optimization (SMO)

Algoritme SMO (Sequential Minimal Optimization) dikembangkan oleh John Platt [11], kemudian diperbaiki oleh Keerthi et al [22]. Platt menerapkan pemilihan dekomposisi ekstrim karena himpunan kerja hanya himpunan dari dua titik yang minimum dengan ketentuan persamaannya sebagai berikut:

$$\sum_{i=1}^{n} \alpha_i y_i \qquad (1)$$

Di sini, untuk memperjelas, α_i adalah lagrange multiplier dan y adalah nama kelas. Ini memungkinkan solusi analitis dari submasalah. Meskipun diperlukan lebih banyak iterasi, setiap iterasi memiliki sedikit operasi saat setiap *subsets* dieksekusi. Oleh karena itu, algoritme menunjukkan percepatan keseluruhan urutan besarnya [11]. John Platt mengklaim bahwa algoritme SMO dapat menawarkan peningkatan efisiensi yang menghemat waktu dan sumber daya yang digunakan relatif kecil, dan SMO sekarang menjadi salah satu algoritme SVM tercepat yang tersedia [11].

Mendefinisikan **kumpulan indeks** I berikut yang menunjukkan model **data pelatihan**:

1) $I_0 = \{i: y_i = 1, 0 < \alpha_1 < C\} \cup \{i: y_i = -1, 0 < \alpha_1 < C\}$
2) $I_1 = \{i: y_i = 1, \alpha_1 = 0\} (Positive\ Non - Support\ Vectors)$
3) $I_2 = \{i: y_i = -1, \alpha_1 = C\} (Bound\ Negative\ Support\ Vectors)$
4) $I_3 = \{i: y_i = 1, \alpha_1 = C\} (Bound\ Positive\ Support\ Vectors)$
5) $I_4 = \{i: y_i = -1, \alpha_1 = C\} (Negative\ Non - Support\ Vectors)$

Disini C sebagai parameter *tuning*, penelitian ini juga mendefinisikan bias [17] b_{up} dan b_{low} dengan indeks yang terkait, sebagai berikut:

$$b_{up} = \min\{f_i: i \in I_0 \cup I_1 \cup I_2\}$$
$$I_{up} = \arg\min_i f_i$$
$$b_{low} = \max\{f_i: i \in I_0 \cup I_3 \cup I_4\}$$

$$I_{low} = \arg\max_i f_i$$

Untuk memastikan bagaimana kondisi sudah optimal melalui vektor:

$$f_i \sum_{j=1}^{i} \alpha_j y_j K(X_j, X_i) - y_i \qquad (2)$$

Pada persamaan diatas K merupakan fungsi dari kernel untuk X_i merupakan data latihnya. Algoritme SMO mengoptimalkan dua α_i yang terkait dengan nilai bias b_{up} dan b_{low}, sebagai berikut:

$$\alpha_2^{new} = \alpha_2^{old} - y_2(f_1^{old} - f_2^{old})/n \qquad (3)$$

$$\alpha_1^{new} = \alpha_1^{old} - s(\alpha_2^{old} - \alpha_2^{new})/n \qquad (4)$$

Pada persamaan diatas dimana $n = 2k(X_1, X_2) - k(X_1, X_1) - k(X_2, X_2)$. Setelah α_1 dan α_2 berhasil dioptimalkan, selanjutnya f_i jika terdapat *error* dalam data pelatihan, melakukan persamaan berikut:

$$f_i^{new} = f_i^{old} + (\alpha_1^{new} - \alpha_i^{old}) y_1 k(X_1, X_i) + (\alpha_2^{new} - \alpha_2^{old}) y_2 (X_2, X_i) \qquad (5)$$

5. Confusion Matrix

Confusion matrix digunakan untuk mengevaluasi kinerja klasifikasi [23], [24]. *Confusion matrix* yang digunakan sama dengan tes metrik yang digunakan pada studi klasifikasi sebelumnya. Metrik kinerja berikut dipertimbangkan dalam evaluasi model:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (6)$$

$$Precision = \frac{TP}{TP + FP} \qquad (7)$$

$$Recall = \frac{TP}{TP + TN} \qquad (8)$$

$$F1 - score = \frac{2PrecisionRecall}{Precision + Recall} \qquad (9)$$

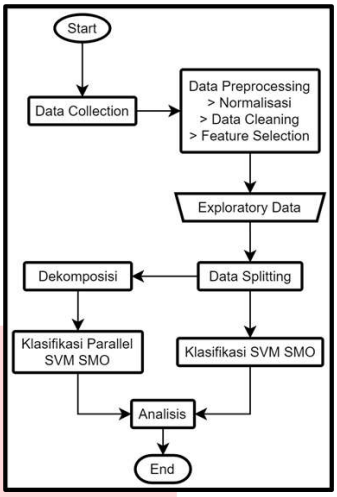
Accuracy: nilai aktual seberapa akurat model dalam melakukan klasifikasi data. Precision: nilai kasus yang terprediksi positif benar pada kelas positif [24], Recall: nilai data positif yang diklasifikasikan dengan benar oleh model [24], F1-Score: nilai rata-rata harmonis dari presisi dan recall [25].

Dimana True Positive (TP) adalah jumlah prediksi yang benar dari klasifikasi malware dan False Negative (FN) adalah jumlah instance malware yang salah diklasifikasikan dalam himpunan [3].

Dimana False Positive (FP) adalah jumlah prediksi yang salah dari klasifikasi aplikasi jinak sedangkan True Negative (TN) adalah jumlah prediksi yang benar dari kasus aplikasi jinak [3].

III. METODE

A. Sistem yang Dibangun



GAMBAR 1.
Desain sistem

` Gambar 1 adalah desain sistem yang dibangun untuk model klasifikasi malware android menggunakan dataset android malware proyek DREBIN [3], dengan teknik SVM-SMO dan dekomposisi SMO untuk konsep klasifikasi SVM paralel.

1. Analisis Kebutuhan Sistem

Sebelum melakukan skenario pengujian dibutuhkan beberapa parameter terkait, parameter pengujian pada penelitian ini adalah jumlah fitur yang diambil saat skenario, *resource* dan spesifikasi perangkat yang tertera pada Tabel 1 dan Tabel 2.

TABEL 1.
Spesifikasi Perangkat Keras

Hardware	Detail
Model	Lenovo G40-70
Prosesor	Intel(R) Core(TM) i3-4030M @1.90GHz
Kartu Grafis	Intel® HD
Memori	2GB DDR3 PC3-12800S + 4GB DDR3 PC3-10800
Penyimpanan	HDD 1TB Seagate FireCuda

TABEL 2.
Spesifikasi Perangkat Lunak

Software
OS Windows 10 Enterprise
Python 3.10
Anaconda 3
Jupyter Notebook
Google Chrome

2. Skenario Pengujian

Sebelum masuk tahapan skenario untuk keterangan kernel terdapat 2 kernel yaitu linier dan gaussian, sedangkan untuk jumlah fitur yang akan diuji (27, 14, 7, 3, 59, dan 215). Berikut skenarionya:

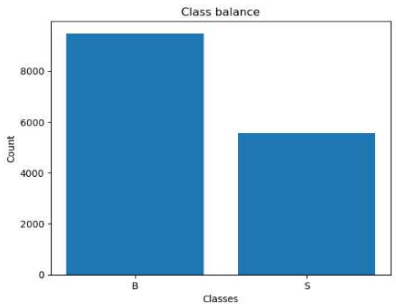
Menguji kinerja perbandingan metrik performa dan akurasi dari model parallel dan non-parallel berdasarkan kernel dan jumlah fitur sebagai parameter, menghasilkan metrik performa dan akurasi, hasil klasifikasi di rata-ratakan untuk analisis data variansi. Pengujian ini juga menentukan apakah kinerja model paralel SMO dapat meningkatkan akurasi model non-paralel SMO.

Menguji kinerja perbandingan waktu eksekusi, SpeedUp dan Efficiency dari model parallel dan non-parallel berdasarkan kernel dan jumlah fitur sebagai parameter. Untuk membandingkan waktu komputasi pada model serta menentukan kernel dan jumlah fitur yang paling optimal. Pengujian ini juga menentukan apakah kinerja model.

Menguji konsep paralel dekomposisi dengan perbandingan jumlah dekomposisi SMO, SMO-2 (2 dekomposisi), dan SMO-4 (4 dekomposisi) kernel sebagai parameter. Untuk membandingkan waktu komputasi pada dekomposisi serta menentukan kernel yang paling optimal. Hasil perbandingan untuk menentukan kinerja dekomposisi.

3. Data Collection

Dataset yang digunakan pada penelitian adalah dataset malware android bersumber dari situs web Kaggle [3]. Dataset berupa table memiliki jumlah kolom sebanyak 215 kolom atribut dan 1 kolom kelas, serta memiliki jumlah baris data 15036 diantaranya terdapat 5.560 aplikasi malware dengan label S (*Suspicious*) dan 9.476 aplikasi jinak dengan label B (*Benign*), kumpulan data berupa vektor fitur. Visualisasi data dapat dilihat pada Gambar 2.



GAMBAR 2.
Data balance

4. Data Preprocessing

Pada fase preprocessing ada beberapa tahapan yang perlu perhatikan untuk mengubah atau encode data sehingga dapat dengan mudah diurai oleh mesin, sebagai berikut:

a. Normalisasi

LabelEncoder digunakan untuk melakukan transformasi atau normalisasi data dengan mengubah kelas label yang berisi nilai kategori yaitu B (Benign) dan S (Suspicious) dengan mengubahnya menjadi nilai integer 0 dan 1.

b. Data Cleaning

Pada dataset terdapat beberapa karakter acak seperti "?" dan "S" maka perlu mengaturnya menjadi NULL supaya terdeksi sebagai missing value setelah itu dapat menghapusnya dengan dropna(). Sehingga hasil akhir dari normalisasi dan clean mengubah nilai kategori B dan S menjadi 0 dan 1, dan untuk jumlah data sampel yang tadinya berjumlah 15036 setelah proses data cleaning tersisa 15031 data sampel.

c. Feature Selection

Dataset DREBIN [3] mempunyai jumlah fitur yang banyak yaitu 215 fitur, studi akan dilakukan seleksi fitur untuk analisis jumlah fitur terbaik pada pemodelan. Pada tahapan seleksi fitur menggunakan Correlation Coefficient Based, hubungan antara dua variabel berkisar dari nilai korelasi -1 untuk hubungan terbalik sempurna dan 1 korelasi positif sempurna, sedangkan untuk nilai korelasi yang mendekati 0 hampir tidak memiliki hubungan atau korelasinya sangat lemah [13]. Pada studi ini memilih fitur yang berkorelasi positif terhadap kelas. Teknik seleksi fitur ini juga diterapkan dibeberapa penelitian [13]–[15].

5. Exploratory Data

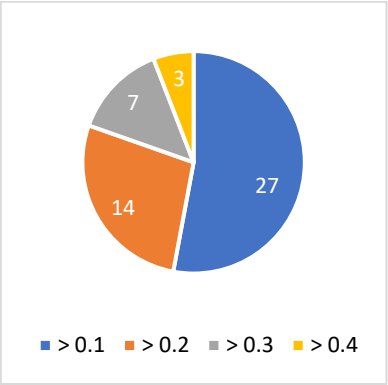
Dari hasil seleksi fitur, fitur yang berkorelasi positif terhadap kelas berjumlah sebanyak 59 fitur atau atribut serta terdapat 4 kategori fitur yaitu Manifest Permission, API call signature, Intent, Command signature [3], [9]. Dari 59 fitur yang berkorelasi positif, READ_CONTACTS adalah korelasi terlemah positif hanya bernilai 0.000831 korelasi dan SEND_SMS adalah korelasi terkuat positif bernilai 0.546075.

Selanjutnya mengidentifikasi fitur paling signifikan [26]. Berdasarkan hasil seleksi fitur menggunakan metode korelasi koefisien, dari analisis data didapatkan jumlah fitur paling signifikan dari dataset malware android DREBIN [3] sebagai berikut:

TABEL 3.
Top 27 feature set DREBIN

No	Fitur	Korelasi	Kategori
1	SEND_SMS	0.546075	Manifest Permission
2	android.telephony.SmsManager	0.435190	API call signature
3	READ_PHONE_STATE	0.409344	Manifest Permission
4	RECEIVE_SMS	0.388328	Manifest Permission
5	READ_SMS	0.370336	Manifest Permission
6	android.intent.action.BOOT_COMPLETED	0.314303	Intent
7	TelephonyManager.getLine1Number	0.305944	API call signature
8	WRITE_SMS	0.267501	Manifest Permission
9	WRITE_HISTORY_BOOKMARKS	0.242250	Manifest Permission
10	TelephonyManager.getSubscriberId	0.241551	API call signature
11	android.telephony.gsm.SmsManager	0.241038	API call signature
12	INSTALL_PACKAGES	0.235660	Manifest Permission
13	READ_HISTORY_BOOKMARKS	0.231044	Manifest Permission
14	INTERNET	0.204219	Manifest Permission
15	ACCESS_LOCATION_EXTRA_COMMANDS	0.197165	Manifest Permission
16	WRITE_APN_SETTINGS	0.193827	Manifest Permission
17	abortBroadcast	0.191727	API call signature
18	createSubprocess	0.185971	API call signature
19	TelephonyManager.getDeviceId	0.179910	API call signature
20	RECEIVE_BOOT_COMPLETED	0.159870	Manifest Permission
21	RESTART_PACKAGES	0.157646	Manifest Permission
22	chmod	0.146465	Commands signature
23	TelephonyManager.getSimSerialNumber	0.135075	API call signature
24	PackageInstaller	0.113861	API call signature
25	remount	0.112943	Commands signature
26	sendDataMessage	0.110062	API call signature
27	chown	0.107540	Commands signature

Pada tabel 3 dari analisis diperoleh 27 fitur paling signifikan yaitu diatas korelasi 0.1, hasil yang diperoleh akan digunakan sebagai parameter untuk meguji perbandingan klasifikasi pada model. Berikut jumlah fitur yang digunakan sebagai parameter uji:



GAMBAR 3.
Top feature dataset DREBIN

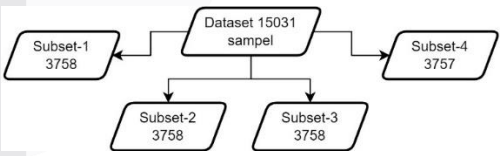
Pada gambar 3 jumlah fitur paling signifikan diatas korelasi 0.1 berjumlah 27 fitur, diatas korelasi 0.2 berjumlah 14 fitur, diatas korelasi 0.3 berjumlah 7 fitur, dan diatas korelasi 0.4 berjumlah 3 fitur.

6. Data Splitting

Data splitting adalah mengesampingkan sebagian data sebagai set evaluasi dan menggunakan sisanya untuk penyetelan model. Penelitian ini mengambil 80% sebagai data latih dan 20% sebagai data uji. Beberapa penelitian mendapatkan akurasi yang optimal dengan perbandingan data 80:20. Menurut penelitian PIndroid pembagian 80:20 sangat efisien [14], menggunakan kernel RBF dan Polynomial dengan feature selection grid search akurasi yang dihasilkan optimal [27].

7. Dekomposisi

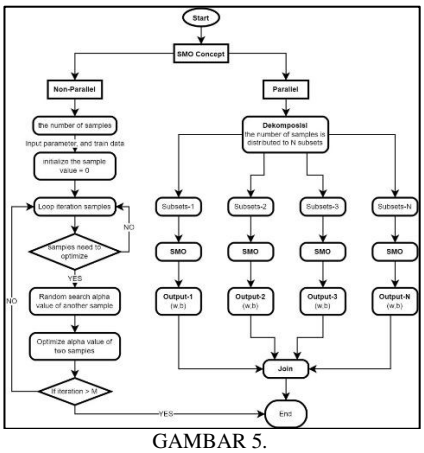
Pada dasarnya dekomposisi digunakan untuk memecah data menjadi sejumlah subset untuk memudahkan dalam proses pelatihan pada data yang berskala besar [16]. Paper survei menyatakan bahwa beberapa penelitian menyebutkan metode dekomposisi sangat diperlukan sebelum melakukan pelatihan pada algoritme Paralel SMO [4]. Pada penelitian ini data terbagi sesuai dengan sumber daya yang dipakai, karena berdasarkan jumlah thread yang tersedia hanya memiliki 4 thread, data sampel akan terbagi menjadi 4 bagian. Berikut skema pembagiannya:



GAMBAR 4.
Dekomposisi dataset

Pada gambar 4 dataset dengan jumlah sampel 15031 didekomposisikan menjadi 4 bagian, dimana per subset terbagi menjadi 3757-3758 sampel.

8. Klasifikasi SVM SMO dan Parallel SVM SMO



GAMBAR 5.
SMO SVM *Non-Parallel and Parallel Concept*

Pseudo- Code untuk algoritme SMO penelitian ini mengutipnya dari report chubak [28] :

Algoritme 1: Pseudo-Code Untuk Algoritme SMO

Input:
C: regularization parameter
tol: numerical tolerance
max_passes: max # of times to iterate over α's without changing
$(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$: training data

Output:
$\alpha \in \mathbb{R}^m$: Lagrange multipliers for solution
$b \in \mathbb{R}$: threshold for solution

Initialize $\alpha_i = 0, \forall i, b = 0$.
Initialize $passes = 0$.
while $(passes < max_passes)$
  $num_changed_alphas = 0$.
  for $i = 1, \dots m,$
    Calculate $E_i = f(x^{(i)}) - y^{(i)}$
    if $((y^{(i)}E_i < -tol \;\&\&\; \alpha_i < C) \,||\, (y^{(i)}E_i > tol \;\&\&\; \alpha_i > 0))$
      Select $j \neq i$ randomly.
      Calculate $E_j = f(x^{(j)}) - y^{(j)}$
      Save old $\alpha's: \alpha_i^{(old)} = \alpha_i, \alpha_j^{(old)} = \alpha_j$.
      Compute L and H
      if $(L == H)$
        continue to next i.
      Compute η
      if $(\eta >= 0)$
        continue to next i.
      Compute and clip new value for α_j
      if $(|\alpha_j - \alpha_j^{(old)}| < 10^{-5})$
        continue to next i.
      Determine value for α_i
      Compute b_1 and b_2
      Compute b
      $num_changed_alphas := num_changed_alphas + 1.$
    end if
  end for
  if $(num_changed_alphas == 0)$
    $passes := passes + 1$
  else
    $passes := 0$
end while

a. Parallel SMO

Konsep paralel menggunakan teknik dekomposisi [4], [16], [17], yaitu memecah data ke beberapa sub bagian untuk melakukan pelatihan pada model klasifikasi paralel SMO. Studi ini mendistribusikan data ke sejumlah *4 subsets*, berikut terkait penjelasan dan persamaannya:
Subset memiliki *output* berupa nilai w (*weight*) parsial dan b (bias) parsial, *output* ini nanti dipakai untuk parsial. Setiap parsial nilai outputnya berbeda, untuk persamaannya:

$$\underset{Partial}{\overrightarrow{w}} = \sum_{i=1}^{I} y_i \alpha_i \overrightarrow{x_i} \qquad (10)$$

Dari *output* parsial, setiap *output* w dirata-ratakan lalu digabungkan untuk mendapatkan nilai bobot atau w (*weight*) global:

$$\underset{Global}{\overrightarrow{w}} = \sum_{i=1}^{n} \underset{Partial}{\overrightarrow{w}} \qquad (11)$$

Selanjutnya *output* parsial dari nilai b, karena nilai b (bias) berbeda untuk setiap partisi. Oleh karena perlu untuk menghitung nilai rata-rata b untuk setiap parsial.
Proses tahapan ini hanya diperlukan nilai w (*weight*) global dan nilai b (bias) global untuk menampilkan hasil *output* klasifikasi SVM paralel, berikut persamaannya:

$$u = \vec{w}.\vec{x} - b \qquad (12)$$

IV. HASIL DAN EVALUASI

Ada dua tahapan pengujian yang pertama untuk menguji metrik performa menggunakan model klasifikasi yang diusulkan pengujian dilakukan pada dua kernel, pengujian kedua untuk membandingkan kinerja paralel dan non-paralel dilakukan pada dua kernel, terdapat evaluasi performa pada pengujian ini yaitu *SpeedUp* dan *Efficiency* definisi persamaannya, sebagai berikut:

$$SpeedUp = \frac{T(1)}{T(n_p)} \qquad (13)$$

$$Efficiency = \frac{SpeedUp}{n_p} \qquad (14)$$

Dimana *SpeedUp* adalah metrik yang menghitung kinerja dengan membagi dua nilai waktu eksekusi, yang dihasilkan oleh waktu eksekusi algoritme serial $T(1)$ dibagi waktu eksekusi paralel $T(n_p)$ waktu dari beberapa

jumlah *cores* [12], [21]. *Efficiency* adalah metrik yang dibangun dari *SpeedUp* dengan jumlah *cores* tersedia untuk melakukan klasifikasi paralel, Karena efisiensi menormalkan jumlah *cores* yang digunakan, ini adalah ukuran yang lebih intrinsik dari kinerja paralel implementasi daripada percepatan [12], [21].

1. Hasil Pengujian

Proses pengujian dilakukan 6 kali eksekusi program, dengan max_iter=3 dan C=0.1

2. Analisis Hasil Pengujian

Analisis yang dilakukan mengacu pada skenario pengujian dari model *Non-Parallel* SVM SMO dan *Parallel* SVM SMO yang diperoleh. Berikut analisis yang dilakukan:

Bagian A, hasil pengujian metrik performa dilakukan 6 kali eksekusi program, memperoleh hasil metrik performa dengan akurasi paling optimal yaitu pada fitur berkorelasi positif yang berjumlah 14 fitur. Dengan perbandingan kernel sebagai parameter.

TABEL 4.
Perbandingan Akurasi Performa *Non-Parallel* (SMO) & *Parallel* (PSMO)

Eksekusi	SMO#1	PSMO#1	SMO#2	PSMO#2	SMO#3	PSMO#3
Jumlah Fitur dan Kernel						
27 Fitur (Linier)	74.61	74.51	74.51	74.38	74.58	74.55
27 Fitur (Gaussian)	74.59	74.49	74.59	74.53	74.59	74.48
14 Fitur (Linier)	77.05	78.32	77.09	78.26	78.31	78.31
14 Fitur (Gaussian)	78.32	78.36	78.35	78.45	78.3	78.42
7 Fitur (Linier)	76.97	76.97	76.97	76.61	76.97	76.97
7 Fitur (Gaussian)	76.97	76.97	76.97	76.97	76.97	76.97
3 Fitur (Linier)	66.95	66.95	66.95	66.95	66.95	66.95
3 Fitur (Gaussian)	66.95	66.95	66.95	66.95	66.95	66.95
59 Fitur (Linier)	68.68	68.66	68.64	68.74	68.74	68.76
59 Fitur (Gaussian)	69.12	69.16	69.37	68.76	69.14	69.19
215 Fitur (Linier)	44.28	44.27	45.19	45.2	45.2	44.32
215 Fitur (Gaussian)	42.64	42.69	42.66	42.7	42.63	42.55

TABEL 5.
Perbandingan Akurasi Performa *Non-Parallel* (SMO) & *Parallel* (PSMO)

Eksekusi	SMO#4	PSMO#4	SMO#5	PSMO#5	SMO#6	PSMO#6
Jumla Fitur dan Kernel						
27 Fitur (Linier)	74.55	74.59	74.49	74.43	74.53	74.55
27 Fitur (Gaussian)	74.6	74.54	75.11	74.46	75.21	74.55
14 Fitur (Linier)	78.32	78.26	78.35	78.36	78.35	78.45
14 Fitur (Gaussian)	78.31	78.34	78.31	78.34	78.35	77.05
7 Fitur (Linier)	76.97	76.97	76.97	76.97	76.91	76.97
7 Fitur (Gaussian)	76.97	76.97	76.97	76.97	76.97	76.97
3 Fitur (Linier)	66.95	66.95	66.95	66.95	66.95	66.95
3 Fitur (Gaussian)	66.95	66.95	66.95	66.95	66.95	66.95
59 Fitur (Linier)	68.68	68.6	68.64	68.61	68.58	68.58
59 Fitur (Gaussian)	69.33	69.24	69.22	68.73	69.4	69.24
215 Fitur (Linier)	44.28	45.23	44.08	45.17	44.33	45.21
215 Fitur (Gaussian)	42.67	42.68	42.57	42.56	42.63	42.59

Dari tabel 4 dan 5 sudah melakukan perbandingan hasil akurasi performa di sejumlah fitur dan kedua kernel hasilnya bervariasi. Dari **6 kali eksekusi program**, karena pada beberapa eksperimen yang dilakukan algoritme paralel smo (PSMO) mampu **memberi peningkatan akurasi**, pada studi kasus ini menyimpulkan bahwa model paralel smo mampu meningkatkan akurasi performa. **Hasil analisis** berdasarkan **rata-rata** dari peningkatan akurasi setiap eksekusi program diperoleh:

Pada jumlah 14 fitur (linier) memperoleh rata-rata akurasi SMO 77.91% dan PSMO 78.33%

Pada jumlah 7 fitur (linier) memperoleh rata-rata akurasi SMO 76.91% dan PSMO 76.96%

Pada jumlah 59 fitur (linier) memperoleh rata-rata akurasi SMO 68.64% dan PSMO 68.66%

Pada jumlah 215 fitur (linier) memperoleh rata-rata akurasi SMO 44.56% dan PSMO 44.9%

Dari hasil akurasi terdapat fitur yang paling **optimal** berjumlah **14 fitur**, pada **215 fitur dan 59 fitur** mendapatkan rata-rata akurasi kecil karena hasil prediksi lebih dominan ke nilai fitur terbesar, sedangkan untuk fitur yang berjumlah sedikit kecil kemungkinan mempengaruhi hasil dari prediksi.

Bagian B, analisis **hasil pengujian** kinerja paralel **4 dekomposisi** yang dilakukan **6 kali eksekusi program**. Setelah dilakukan analisis perbandingan, data fitur yang berjumlah **27 fitur** pada kernel linier dan gaussian memperoleh hasil waktu komputasi, percepatan dan efisiensi **paling optimal**. Selanjutnya melakukan perbandingan berdasarkan kernel, sebagai berikut:

TABEL 6.
Perbandingan Kinerja *Non-Parallel* (SMO) & *Parallel* (PSMO) Pada Kernel Linier

Eksekusi	SMO	PSMO-4		
	Runtime(s)	Runtime(s)	Speedup	Efficiency
1	165.4	15.74	10.48	2.62
2	152.79	16.18	9.44	2.36
3	158.25	16.45	9.62	2.4
4	169.9	20.16	8.42	2.1
5	162.16	16.63	9.75	2.43
6	182.27	18.59	9.8	2.45

Average		9.58	2.39

Pada tabel 6, dari seluruh eksekusi yang dilakukan menghasilkan percepatan yang signifikan dari masing-masing eksekusi serta model PSMO-4 pada kernel linier mampu mengatasi masalah waktu komputasi. Dari 6 kali eksekusi program diantaranya eksekusi 1 menghasilkan percepatan 10x, pada eksekusi 2, 3, 5, dan 6 menghasilkan percepatan 9x, dan pada eksekusi 4 percepatan 8x.

TABEL 7.
Perbandingan Kinerja *Non-Parallel* (SMO) & *Parallel* (PSMO) Pada Kernel Gaussian

Eksekusi	**SMO**	**PSMO-4**		
	Runtime(s)	**Runtime(s)**	**Speedup**	**Efficiency**
1	165.8	17.47	9.49	2.37
2	154.54	18.2	8.49	2.12
3	155.78	17.99	8.65	2.16
4	160.34	20.97	7.64	1.91
5	165.48	18.44	8.97	2.24
6	183.05	19.05	9.6	2.4
Average			8.8	2.2

Pada tabel 7, dari seluruh eksekusi yang dilakukan menghasilkan percepatan yang signifikan dari masing-masing eksekusi serta model PSMO-4 pada kernel gaussian mampu mengatasi masalah waktu komputasi. Dari 6 kali eksekusi program diantaranya pada eksekusi 1 dan 6 menghasilkan percepatan 9x, pada eksekusi 2, 3, dan 5 menghasilkan percepatan 8x, dan pada eksekusi 4 percepatan 7x.

Pada tabel 6 dan 7 perbandingan kinerja SMO dan PSMO-4 diatas, **kinerja kernel linier** menghasilkan **rata-rata** *speedup* 9.58 dan *efficiency* 2.39 **jauh lebih baik** dibandingkan kernel gaussian dengan **rata-rata** *speedup* 8.8 dan *efficiency* 2.2.

Bagian C, analisis uji konsep paralel, pada eksperimen ini berfokus menguji **konsep dekomposisi**. Terdapat dua dekomposisi dilakukan, PSMO-2 untuk **2 dekomposisi** dan PSMO-4 untuk **4 dekomposisi**, jumlah dekomposisi berdasarkan **kelipatan memori dan *resource*** yang tersedia.

TABEL 8.
Uji Konsep Paralel Dekomposisi Kernel Linier (L) dan Gaussian (G)

Eksekusi	**SMO**		**PSMO-2**						**PSMO-4**					
	Runtime(s)		**Runtime (s)**		**Speedup**		**Efficiency**		**Runtime(s)**		**Speedup**		**Efficiency**	
Kernel	**L**	**G**	**L**	**G**	**L**	**G**	**L**	**G**	**L**	**G**	**L**	**G**	**L**	**G**
27 fitur	162	162	85	80	1.9	2.02	0.47	0.5	16	18	10.3	9.22	2.57	2.3
14 fitur	84	88	16	18	5.13	4.96	1.28	1.24	11	13	7.45	6.79	1.86	1.69
7 fitur	18	19	11	14	1.56	1.4	0.39	0.35	8	10	2.06	1, 89	0.51	0.47
3 fitur	10	11	7	10	1.2	1.1	0.32	0.2	5	6	2.08	1.89	0.52	0.4
59 fitur	333	339	173	172	1.92	1.96	0.48	0.49	87	94	3.82	3.59	0.95	0.89
215 fitur	1179	1174	585	585	2.01	2	0.5	0.5	301	306	3.91	3.84	0.97	0.96
Average					2.3	2.25	0.57	0.56			4.93	4.53	1.23	1.13

Pada tabel 8, menggunakan kernel linier, hasil eksekusi PSMO-2 menghasilkan **nilai rata-rata** percepatan 2.3 dan efisiensi 0.57, sedangkan PSMO-4 menghasilkan percepatan 4.93 dan efisiensi 1.23. menggunakan kernel gaussian, hasil eksekusi PSMO-2 menghasilkan **nilai rata-rata** percepatan 2.25 dan efisiensi 0.56, sedangkan PSMO-4 menghasilkan percepatan 4.53 dan efisiensi 1.13.

V. KESIMPULAN

A. Kesimpulan

Studi penelitan ini mampu membangun suatu model klasifikasi algoritme SVM dengan menerapkan teknik SMO dan dekomposisi SMO untuk konsep paralel dalam melakukan deteksi malware android yang menggunakan dataset malware android proyek DREBIN [3]. Dataset malware android DREBIN memiliki dimensi tinggi dan jumlah data yang besar.

Studi penelitian ini berhasil menerapkan konsep paralel dengan dekomposisi SMO, PSMO-4 memperoleh hasil percepatan 4x lebih cepat dibanding SMO dan 2x lebih cepat dibanding PSMO-2. Hasil dan analisis perbandingan klasifikasi Non-Parallel SVM-SMO dan Parallel SVM-SMO menggunakan Parallel SVM-SMO mampu meningkatkan akurasi dapat dilihat pada tabel 4 dan 5.

Pada perbandingan akurasi jumlah 14 fitur (linier) memperoleh rata-rata akurasi SMO 77.91% mengalami peningkatan rata-rata akurasi pada model PSMO-4 78.33%. Pada perbandingan kinerja model SMO dan PSMO-4 memperoleh waktu komputasi, percepatan dan efisiensi dari fitur optimal yang berkorelasi positif diatas 0.1 sejumlah 27 fitur, model Parallel SVM-SMO menggunakan kernel linier dengan 4 dekomposisi mampu menghemat waktu komputasi hampir 10 kali lipat dari algoritme Non-Parallel SVM-SMO dengan nilai rata-rata percepatan 9.58 dan efisiensi 2.39, sedangkan pada kernel gaussian mendapatkan rata-rata percepatan 8.8 dan efisiensi 2.2.

Ada beberapa saran penelitian selanjutnya untuk dilakukan yaitu

Jika klasifikasi menggunakan algoritme SVM, mampu mencoba menerapkan teknik *preprocessing data* untuk menyeimbangkan variable dependen (*class*).

Menerapkan teknik optimasi SVM lain atau menggunakan alternatif algoritme *deep learning*.

REFERENSI

[1] A. Haleem, M. Javaid, R. Singh, S. Rab, and R. Suman, "Perspectives of Cybersecurity for Ameliorative Industry 4.0 Era: A Review based Framework," *Industrial Robot*, vol. ahead-of-print, Dec. 2022, doi: 10.1108/IR-10-2021-0243.

[2] "Mobile Operating System Market Share Worldwide | Statcounter Global Stats." https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202112-202212 (accessed Dec. 23, 2022).

[3] S. Y. Yerima and S. Sezer, "DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection," *IEEE Trans Cybern*, vol. 49, no. 2, pp. 453–466, Jan. 2018, doi: 10.1109/tcyb.2017.2777960.

[4] S. Tavara, "Parallel computing of support vector machines: A survey," *ACM Computing Surveys*, vol. 51, no. 6. Association for Computing Machinery, Jan. 01, 2019. doi: 10.1145/3280989.

[5] L. Zanni, S. T. It, and G. Z. It, "Parallel Software for Training Large Scale Support Vector Machines on Multiprocessor Systems * Thomas Serafini Gaetano Zanghirati," 2006. [Online]. Available: http://www.dm.unife/gpdt.

[6] M. O. Stitson, J. A. E. Weston, A. Gammerman, V. Vovk, and V. Vapnik, "Theory of Support Vector Machines," 1996.

[7] D. A. Pisner and D. M. Schnyer, "Support vector machine," in *Machine Learning: Methods and Applications to Brain Disorders*, Elsevier, 2019, pp. 101–121. doi: 10.1016/B978-0-12-815739-8.00006-7.

[8] H. Han, S. Lim, K. Suh, S. Park, S. J. Cho, and M. Park, "Enhanced android malware detection: An SVM-based machine learning approach," in *Proceedings - 2020 IEEE International Conference on Big Data and Smart Computing, BigComp 2020*, Feb. 2020, pp. 75–81. doi: 10.1109/BigComp48618.2020.00-96.

[9] A. Muzaffar, H. Ragab Hassen, M. A. Lones, and H. Zantout, "An in-depth review of machine learning based Android malware detection," *Computers and Security*, vol. 121. Elsevier Ltd, Oct. 01, 2022. doi: 10.1016/j.cose.2022.102833.

[10] J. Khalilzadeh and A. D. A. Tasci, "Large sample size, significance level, and the effect size: Solutions to perils of using big data for academic research," *Tour Manag*, vol. 62, pp. 89–96, Oct. 2017, doi: 10.1016/j.tourman.2017.03.026.

[11] J. C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," 1998.

[12] E. A. Carmona and M. D. Rice, "A MODEL OF PARALLEL PERFORMANCE."

[13] N. Shehab, M. Badawy, and H. A. Ali, "Toward feature selection in big data preprocessing based on hybrid cloud-based model," *Journal of Supercomputing*, vol. 78, no. 3, pp. 3226–3265, Feb. 2022, doi: 10.1007/s11227-021-03970-7.

[14] F. Idrees, M. Rajarajan, M. Conti, T. M. Chen, and Y. Rahulamathavan, "PIndroid: A novel Android malware detection system using ensemble learning methods," *Comput Secur*, vol. 68, pp. 36–46, Jul. 2017, doi: 10.1016/j.cose.2017.03.011.

[15] K. Deepa, G. Radhamani, and P. Vinod, "Investigation of feature selection methods for android malware analysis," in *Procedia Computer Science*, 2015, vol. 46, pp. 841–848. doi: 10.1016/j.procs.2015.02.153.

[16] P.-H. Chen, R.-E. Fan, and C.-J. Lin, "A Study on SMO-type Decomposition Methods for Support Vector Machines," 2006.

[17] R. R. Prasojoe and S. Setyorini, "SVM Parallel Concept Test with SMO Decomposition on Cancer Microarray Dataset," in *2021 9th International Conference on Information and Communication Technology (ICoICT)*, 2021, pp. 467–471. doi: 10.1109/ICoICT52021.2021.9527411.

[18] S. Y. Vali and J. M. Sudha, "Article ID: IJARET_11_11_157 Android Application In Smartphones," *International Journal of Advanced Research in Engineering and Technology (IJARET)*, vol. 11, no. 11, pp. 1678–1693, 2020, doi: 10.34218/IJARET.11.11.2020.157.

[19] Ken. Dunham, *Mobile malware attacks and defense*. Syngress/Elsevier, 2009.

[20] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2018.

[21] G. R. Andrews, *Foundations of multithreaded, parallel, and distributed programming*. Addison-Wesley, 2020.

[22] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO Algorithm for SVM Classifier Design."

[23] S. Visa, B. Ramsay, A. Ralescu, and E. van der Knaap, "Confusion Matrix-based Feature Selection."

[24] "Evaluation_From_Precision_Recall_and_F-Factor_to_R".

[25] Y. Sasaki and R. Fellow, "The truth of the F-measure," 2007.

[26] V. Kouliaridis, G. Kambourakis, and T. Peng, "Feature importance in Android malware detection", doi: 10.1109/TrustCom50675.2020.00195/20/$31.00.

[27] A. N. Ulfah, M. Khairul Anam, N. Yona, S. Munti, S. Yaakub, and M. B. Firdaus, "Sentiment Analysis of the Convict Assimilation Program on Handling Covid-19," 2022.

[28] "CS229 Simplified SMO Algorithm." [Online]. Available: http://research.microsoft.com/