IMPLEMENTASI PENDEKATAN *GRAPHIC PROCESSING UNIT* (GPU) PADA ALGORITMA *SUPPORT VECTOR MACHINE* (SVM) DAN *LEAST SQUARES SUPPORT VECTOR MACHINE* (LS-SVM)

IMPLEMENTATION OF GRAPHIC PROCESSING UNIT (GPU) APPROACH IN SUPPORT VECTOR MACHINE (SVM) AND LEAST SQUARES SUPPORT VECTOR MACHINE (LS-SVM) ALGORITHM

Hamnis Rachmat Dhana[1], Fhira Nhita, M.T.[2], Izzatul Ummah, M.T.[3]
Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
[1]hamnis.2d@gmail.com , [2]fhiranhita@telkomuniversity.ac.id , [3]izzatulummah@telkomuniversity.ac.id

Abstrak

***Data mining* adalah metode untuk menggali informasi yang berguna dari dataset. Salah satu teknik dalam *data mining* adalah klasifikasi. Klasifikasi banyak digunakan pada identifikasi penyakit, *biomedical enggineering* dan lainnya. Salah satu algoritma yang digunakan untuk klasifikasi adalah *Support Vector Machine* (SVM). SVM merupakan algoritma untuk melakukan klasifikasi dengan memaksimalkan *hyperplane* diantara kelas. SVM memiliki banyak variasi seperti *Least Square Support Vector Machine* (LS-SVM). Proses klasifikasi memerlukan waktu yang lama karena data latih memiliki banyak *record* dan *attribute*. Untuk mengatasi masalah tersebut maka SVM dan LS-SVM diimplementasikan menggunakan pendekatan *Graphic Processing Unit* (GPU) sehingga dapat meningkatkan kecepatan proses komputasi dibandingkan dengan menggunakan *processor* biasa. Berdasarkan beberapa penelitian tentang kinerja SVM menggunakan GPU seperti pada penelitian yang dilakukan oleh Jesse Patrick Harvey maka dapat dinyatakan bahwa SVM dengan GPU lebih cepat 89×-263× dibandingkan SVM tanpa GPU. Penelitian oleh Jesse Patrick Harvey membuktikan bahwa SVM menggunakan pendekatan GPU memiliki kinerja yang lebih baik dibandingkan dengan SVM tanpa GPU. Pada tugas akhir ini akan dibahas mengenai perbandingan akurasi dan waktu komputasi dari kedua algoritma yang telah disebutkan. *Dataset* yang digunakan diambil dari *http://ntucsu.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html*.**

Kata kunci: ***Classification, record, attribute SVM, LS-SVM, GPU.***

Abstract

Data mining is a method to find useful information from dataset. One of the techniques in data mining is classification. Classification is widely used in the identification of diseases, biomedical engineering, etc. One of the algorithms used for classification is Support Vector Machine (SVM). The concept of a SVM is doing classification by maximizing hyperplane between class. SVM has many variations such as Least Squares Support Vector Machine (LS-SVM). Classification process takes a long time because the training data has many records and attributes. To overcome the problem of time the SVM and LS-SVM implemented using Graphic Processing Unit (GPU), so it can increase the speed of the computing process compared to using regular processor. Based on research about the performance of SVM using GPU as the research by Jesse Patrick Harvey stated that SVM with GPU 89×-263× faster than SVM without GPU. Research by Jesse Patrick Harvey proved that SVM using GPU approach better performance than SVM without GPU. In this final project will discuss about comparison of accuracy and computation time of the algorithms that have been mentioned. The dataset used was taken from *http://ntucsu.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html*.

Keywords: **Classification, SVM, LS-SVM, GPU.**

1. Pendahuluan

Data mining merupakan proses penggalian data dari suatu *database* sehingga mendapatkan informasi yang baru. Salah satu teknik *data mining* adalah klasifikasi. Studi kasus yang biasanya dibahas pada klasifikasi adalah klasifikasi penyakit, klasifikasi citra, klasifikasi audio, dan klasifikasi dokumen web. Namun seringkali data yang akan diolah memiliki ratusan bahkan ribuan atribut dan memiliki jumlah yang bisa mencapai jutaan sehingga apabila diproses akan memerlukan waktu komputasi yang lama.

Untuk menyelesaikan permasalahan dalam pengolahan data yang banyak, maka digunakan *Graphic Processing Unit* (GPU) untuk membantu prosesnya. Pembahasan mengenai penggunaan GPU dalam proses klasifikasi telah banyak dilakukan. Seperti pada penelitian yang dilakukan oleh Harvey, Jesse Patrick dengan judul "GPU acceleration of object classification using NVIDIA CUDA"[1] menyimpulkan bahwa proses *Support*

Vector Machine (SVM) dengan GPU, 89x sampai 263x lebih cepat dibandingkan dengan LIBSVM ataupun MATLAB's *Parallel Computing Toolbox*. Penelitian yang dilakukan oleh Jesse Patrick Harvey menggunakan *database* dari Mixed National Institute of Standards and Technology (MNIST) yang mana jumlah data nya adalah 406406.

Penelitian kali ini juga mengacu pada penelitian yang telah dilakukan oleh Kemal Polat dan Salih Güneş yang berjudul "*Breast cancer diagnosis using least squares support vector machine*"[2]. LS-SVM merupakan variasi dari SVM.

Yang akan dilakukan adalah proses klasifikasi menggunakan SVM dan LS-SVM dengan pendekatan GPU. Untuk algoritma LS-SVM juga akan dikombinasikan dengan *Parallel Computing Toolbox* sehingga waktu komputasi yang didapat akan lebih optimal. Pada penelitian ini juga akan dilakukan perbandingan performansi agar dapat diketahui akurasi dan waktu komputasi dari kedua algoritma yang telah disebutkan.

Rumusan masalah penelitian ini adalah :

a. Bagaimana mengimplementasikan algoritma *Support Vector Machine* (SVM) dan *Least Square Support Vector Machine* (LS-SVM) dengan menggunakan pendekatan *Graphic Processing Unit* (GPU) pada data yang mempunyai banyak *record* dan atau *attribute*?
b. Bagaimana mengimplementasikan algoritma *Least Square Support Vector Machine* (LS-SVM) menggunakan pendekatan GPU yang dikombinasikan MATLAB *Parallel Computing Toolbox*?
c. Bagaimana analisis akurasi dan waktu komputasi dari SVM dan LS-SVM menggunakan pendekatan GPU dengan SVM dan LS-SVM tanpa menggunakan pendekatan GPU?
d. Bagaimana analisis mengenai evaluasi performansi dari LS-SVM yang menggunakan pendekatan GPU yang dikombinasikan dengan MATLAB *Parallel Computing Toolbox* agar dapat diketahui *speedup*, *performance improvement*, dan *efficiency*?
e. Bagaimana pengaruh dari parameter γ yang digunakan pada SVM dan LS-SVM terhadap nilai akurasi?

Tujuan dari penelitian ini adalah :

a. Mengimplementasikan algoritma *Support Vector Machine* (SVM) dan *Least Square Support Vector Machine* (LS-SVM) dengan menggunakan pendekatan *Graphic Processing Unit* (GPU) pada data yang mempunyai banyak *record* dan atau *attribute*.
b. Mengimplementasikan algoritma *Least Square Support Vector Machine* (LS-SVM) menggunakan MATLAB *Parallel Computing Toolbox*.
c. Mengetahui akurasi dan waktu komputasi dari SVM dan LS-SVM menggunakan pendekatan GPU dengan SVM dan LS-SVM tanpa menggunakan pendekatan GPU.
d. Dapat melakukan analisis mengenai evaluasi performansi dari LS-SVM yang menggunakan pendekatan GPU yang dikombinasikan dengan MATLAB *Parallel Computing Toolbox* agar dapat diketahui *speedup*, *performance improvement*, dan *efficiency*.
e. Mampu menjelaskan pengaruh dari parameter γ yang digunakan pada SVM dan LS-SVM terhadap nilai akurasi.

Batasan masalah penelitian ini adalah :

a. Data yang digunakan merupakan *binary class*, yaitu data yang hanya memiliki dua kelas.
b. Tidak membahas lebih jauh mengenai *preprocessing* data.

Penelitian ini diharapkan dapat memberikan tambahan pembelajaran mengenai klasifikasi data menggunakan algoritma *Support Vector Machine* (SVM) dan *Least Squares Support Vector Machine* (LS-SVM) yang diimplementasikan menggunakan pendekatan *Graphic Processing Unit* (GPU).

2. Landasan Teori

2.1 *Support Vector Machine* (SVM)

Support Vector Machines hanya dapat membedakan dua *class*[1]. *Label class* biasanya adalah +1 dan -1. SVM berusaha menemukan *hyperplane* dengan *maximum margin* diantara *dataset*[5].

Selama proses klasifikasi, algoritma SVM akan menentukan pada sisi mana *hyperplanes* akan diletakkan. Data yang digunakan sebagai inputan dinotasikan (x_i,y_i) (i=1,2,...,N), dimana x_i merupakan vektor dan N adalah banyaknya data. Label dari *class* dinotasikan $y_i \in \{-1,+1\}$. Asumsi kedua *class* dapat terpisah secara sempurna oleh *hyperplane*, yang didefinisikan[3,4] :

$$w \cdot x + b = 0 \quad (1)$$

x_i termasuk *class* -1 bila memenuhi pertidaksamaan

$$w \cdot x + b \leq -1 \quad (2)$$

x_i termasuk *class* +1 bila

$$w \cdot x + b \geq +1 \quad (3)$$

Dimana w dan b merupakan parameter dari model.

Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara *hyperplane* dan titik terdekatnya[3]. Hal ini dapat dirumuskan sebagai *Quadratic Programming* (QP) *problem*, yaitu mencari titik minimal persamaan (4), dengan memperhatikan *constraint* pada persamaan (5)[3].

$$\min_{w} = \frac{1}{2}\|\vec{w}\|^2 \quad (4)$$

$$y_i(w \cdot x_i + b) \geq 1,\ i = 1,2,...,N \quad (5)$$

Problem ini dapat diselesaikan dengan *Lagrange Multiplier*.

$$L_P = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\vec{x}_i \cdot \vec{w} + b) - 1) \quad (6)$$

α_i adalah pengali *lagrange*, yang bernilai nol atau positif ($\alpha_i \geq 0$). Nilai optimal dari persamaan (6) dapat dihitung dengan meminimalkan L_P terhadap w dan b, dan memaksimalkan L terhadap α_i. Dengan memperhatikan sifat bahwa pada titik optimal *gradient* L=0, persamaan (6) dapat dimodifikasi sebagai maksimalisasi *problem* yang hanya mengandung α_i, sebagaimana persamaan (7) dibawah[3].

Maksimalkan :

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \quad (7)$$

Dengan kendala :

$$\vec{a} = \sum_{i=1}^{N} \alpha_i y_i = 0 \quad (8)$$

Dari hasil perhitungan ini diperoleh α_i yang kebanyakan bernilai positif. α_i yang bernilai positif inilah yang disebut *support vector*[3]. *Support vector* adalah data yang berkontribusi dalam membentuk model yang akan digunakan. Setelah parameter model *w* dan *b* ditemukan maka gunakan fungsi sederhana (9) untuk melakukan klasifikasi[6].

$$f(z) = sign\ (w \cdot z + b) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \vec{x}_i \cdot \vec{z} + b) \quad (9)$$

Jika $f(z) = 1$, maka termasuk *class* positif dan selain itu masuk ke dalam *class* negatif[6].

Kasus yang sering terjadi pada klasifikasi adalah masalah *non linear* sehingga untuk mengatasinya digunakan fungsi kernel. Fungsi kernel digunakan untuk memetakan data ke ruang vektor yang berdimensi lebih tinggi, sehingga kedua *class* dapat dipisahkan secara *linear* oleh sebuah *hyperplane* [1].

Kernel yang umum digunakan :

Linear: $K_{Linear}(x_i, x_j) = x_i^T \cdot x_j$ (10)

Polynomial : $K_{Polunomial}(x_i, x_j) = (x_i^T \cdot x_j + 1)^n$ (11)

Radial Basis Function (RBF) : $K_{RBF}(x_i, x_j) = \frac{-|x_i - x_j|^2}{2\sigma^2}$ (12)

Maka untuk mencari pengali *lagrange* pada masalah *non linear* persamaan (7) dilakukan modifikasi.

Minimalkan :

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \quad (13)$$

Dengan kendala :

$$\vec{a} = \sum_{i=1}^{N} \alpha_i y_i = 0 \quad (14)$$

Fungsi klasifikasinya menjadi

$$f(z) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \vec{x}_i \cdot \vec{z} + b) \quad (15)$$

2.2 *Least Squares Support Vector Machine* (LS-SVM)

Least Square Support Vector Machine (LS-SVM) merupakan variasi dari algoritma *Support Vector Machine* (SVM). Proses untuk menyelesaikan *problem* pada LS-SVM sama seperti dengan SVM, hanya saja pembatas dari fungsi *Lagrange* berupa persamaan tidak seperti SVM yang pembatasnya berupa pertidaksamaan. Persamaan dari LS-SVM adalah sebagai berikut :

$$\min_{w} = \frac{1}{2}\|\vec{w}\|^2 \quad (16)$$

$$y_i(w \cdot x_i + b) = 1,\ i = 1,2,...,N \quad (17)$$

Pembatas yang berupa persamaan mengakibatkan LS-SVM mendapatkan solusi dari pemecahan persamaan linear.

2.3 *Graphic Processing Unit* (GPU)

Pada dasarnya, *Graphic Processing Unit* (GPU) banyak digunakan untuk proses *rendering* dan *gaming*

consoles, namun pada beberapa tahun terakhir GPU juga digunakan untuk melakukan komputasi paralel[1].

2.4 CUBLAS

CUBLAS *library* merupakan implementasi dari *Basic Linear Algebra Subprograms* (BLAS) yang berjalan diatas *Computing Unified Device Architecture* (CUDA). CUBLAS memungkinkan *programmer* untuk mengakses sumber daya pada GPU sehingga dapat melakukan proses komputasi pada GPU.

2.5 *Parallel Computing Toolbox*

Parallel Computing Toolbox merupakan metode yang dikembangkan pada program MATLAB dan sangat efisien dalam melakukan *parallel task processing* atau biasa disebut *parfor*. Meskipun metode *parallel task processing* merupakan metode yang lebih mengarah kepada *speed-ing Central Processing Unit* (CPU) daripada *Graphic Processing Unit* (GPU), tetapi apabila dikombinasikan degan GPU maka waktu komputasi yang didapat akan lebih optimal.

Tabel 1 Konsep MATLAB *Parallel Computing Toolbox*

<table>
<tr><td colspan="5">*for* i = 1:100
perintah
…
End</td></tr>
<tr><td>matlabpool

parfor i = 1:100
perintah
…
end</td><td colspan="4">Jika *worker* berjumlah 4, maka pembagian perintah *for* akan dieksekusi secara paralel seperti yang digambarkan pada tabel.</td></tr>
<tr><td></td><td>*for* i = 1:25
perintah
…
end</td><td>*for* i = 26:50
perintah
…
End</td><td>*for* i = 51:75
perintah
…
end</td><td>*for* i = 76:100
perintah
…
end</td></tr>
</table>

2.6 Evaluasi Hasil Klasifikasi

Evaluasi dari hasil klasifikasi menggunakan *classification accuracy*[7]. *Classification accuracy* adalah ketepatan klasifikasi yang diperoleh.

$$\textit{Classification accuracy}\ (\%) = \frac{\square + \square}{\square + FP + FN + \square} \quad (16)$$

Classification accuracy dapat ditentukan menggunakan nilai yang terdapat dalam *confusion matrix*. *Confusion matrix* merupakan klasifikasi tentang aktual dan prediksi yang dilakukan dengan sistem klasifikasi[7].

Tabel 2 *Confusion Matrix*

Actual	*Predicted*	
	Positive = class 0	*Negative = class* 1
Positive = class 0	*True Positive* (TP)	*False Negative* (FN)
Negative = class 1	*False Positive* (FP)	*True Negative* (TN)

2.7 Evaluasi Performansi Model

Pada penelitian ini, evaluasi dari performansi model menggunakan *speedup*, *performance improvement*, dan *efficiency*[8].

Speedup mengukur seberapa cepat waktu komputasi algoritma paralel dibandingkan dengan algoritma sekuensial.

$$\textit{Speedup} = \frac{Wa\ \square\square\square\ \square\square\square\square\square\square\square\ \square(\square)}{W\square\square\square\square\square W\square\square\square\square\square\square(\square} \quad (17)$$

Performance improvement menggambarkan peningkatan performa yang menyatakan bahwa algoritma paralel lebih baik dari algoritma sekuensial.

$$\textit{Performance improvement} = \frac{Wa\ \square\square\square\ \square\square\square\square\square\square\square\ \square(\square) - Wa\ \square\square\square Wa\ \square\ \square\square\square(}{\square} \quad (18)$$

$$W\square\square\square\square\square W\square\square\square\square\square\square(\square$$

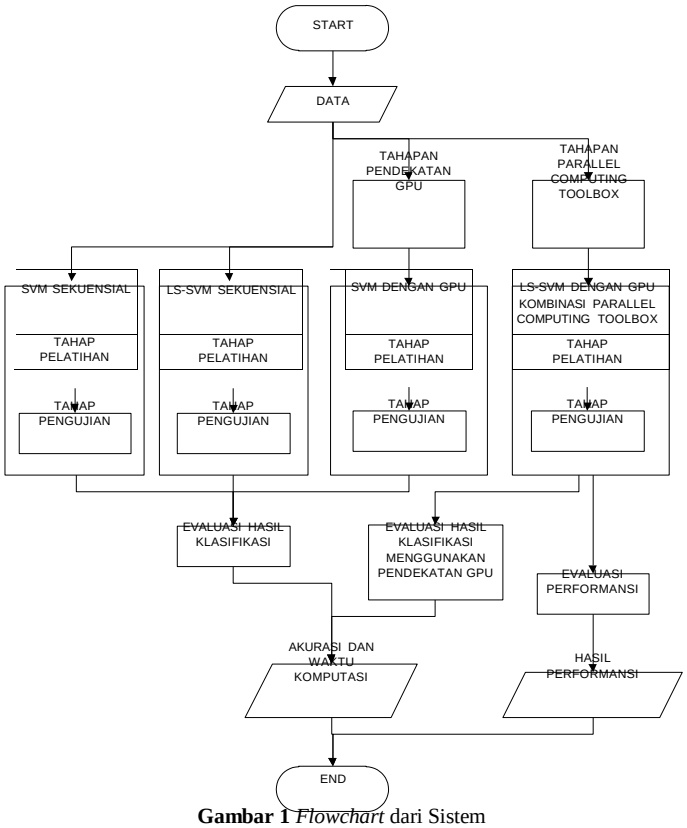
Efficiency digunakan untuk mengestimasi seberapa baik proses pemanfaatan *processors* dalam menyelesaikan masalah dibandingkan dengan usaha yang digunakan dalam berkomunikasi dan sinkronisasi.

$$\textit{Efficiency} = \frac{Wa\ \square\square\square\ \square\square\square\square\square\square\square\ \square(\square)}{\square\square\square Wa\ W\square\square\square\square\times W\square\square\square\square W\square\square\square\square\square\square(\square} \quad (19)$$

3. Perancangan Sistem

3.1 Deskripsi Sistem

Sistem klasifikasi penyakit menggunakan algoritma *Support Vector Machine* (SVM) dan *Least Square Support Vector Machine* (LS-SVM) yang proses kerjanya diselesaikan menggunakan pendekatan *Graphic Processing Unit* (GPU).



Gambar 1 *Flowchart* dari Sistem

Dalam sistem yang dibangun, akan dilakukan analisis mengenai perbandingan akurasi dan waktu komputasi. Untuk akurasi akan dibandingkan SVM dengan LS-SVM. Sedangkan untuk waktu komputasi akan dibandingkan SVM sekuensial dengan SVM paralel dan LS-SVM sekuensial dengan LS-SVM paralel.

3.2 Tahapan Pelatihan

Tahapan pelatihan merupakan proses yang dilakukan untuk mencari nilai dari parameter yang mengisi fungsi pemisah (*hyperplane*), seperti *w*, (pengali *lagrange*), *b* (bias). Langkah-langkahnya sebagai berikut :

a. Tentukan vektor input (x_i) dan target (y_i)
b. Cari nilai (pengali *lagrange*) dengan persamaan :

$$\text{□} = \sum^{N} \; l_{\text{□}} - \frac{1}{\_} \sum \; l\, l \; \text{□□□} \cdot \text{□} \qquad (7)$$

$$\text{□}=1 \qquad 2 \quad \text{□□ □□ □□ □□ □ □}$$

$$\text{□} = \sum$$

c. Hitung nilai parameter *w* dengan persamaan :

$$\sum_{\text{□}=1}^{N} l_{\text{□}}\text{□□} = 0 \qquad (8)$$

d. Hitung nilai *b* (bias) dengan persamaan :

$$b = \text{□}_{\text{□}} - \text{□} \cdot \text{□}_{\text{□}} = 0 \qquad (20)$$

3.3 Tahapan Pengujian

Setelah tahapan pelatihan akan diperoleh parameter model *w* dan *b* (bias), kemudian akan dilakukan tahap pengujian. Langkah-langkahnya sebagai berikut :

a. Masukkan parameter yang telah dicari seperti *w* dan *b* ke dalam *hyperplane*.
b. Masukkan vektor dari data uji ke dalam *hyperplane* dan lakukan perhiitungan nilai *hyperplane* dengan persamaan :

$$f(\text{z}) = \text{□□□}\left(\sum_{\text{□}=1}^{N} l_{\text{□}}\text{□□□} \cdot \text{□} + \text{□□}\right) \qquad (15)$$

c. Tentukan *class* dari vektor data uji berdasarkan nilai *hyperplane*

3.4 Tahapan Pendekatan GPU pada Algoritma SVM dan LS-SVM

Pada algoritma SVM menggunakan fungsi perkalian matriks-vektor dalam CUBLAS pada −|□− □^2 dari fungsi kernel gaussian (12) yang dapat diubah menjadi bentuk matriks-vektor (2x×y−x×x−y×y). Jadi untuk menghitung perkalian x dan y akan digunakan fungsi pada CUBLAS.

Sedangkan pada algoritma LS-SVM menggunakan fungsi sum dalam CUBLAS pada saat perhitungan akurasi.

3.5 Tahapan Parallel Computing Toolbox pada Algoritma LS-SVM

Tahapan dalam mengimplementasikan MATLAB *Parallel Computing Toolbox* pada *Least Square Support Vector Machine* (LS-SVM) adalah dengan menentukan jumlah *worker* yang digunakan untuk mengerjakan perintah, kemudian mengubah algoritma pada bagian *for-loop* menggunakan *parfor* pada fungsi kernel gaussian.

4. Pembahasan

4.1 Skenario Pengujian

Dataset yang digunakan untuk percobaan diambil dari *http://ntucsu.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html*[9]. *Dataset* yang digunakan telah dilakukan *preprocessing* menjadi ke dalam bentuk *LIBSVM format*.

Tabel 3 *Dataset*

No	*Dataset*	Jumlah atribut	Jumlah *record* data latih	Jumlah *record* data tes	Kelas
1	a9a	14	15000	15000	2
2	w6a	300	14000	14000	2

Skenario pengujian pada penelitian ini adalah sebagai berikut:

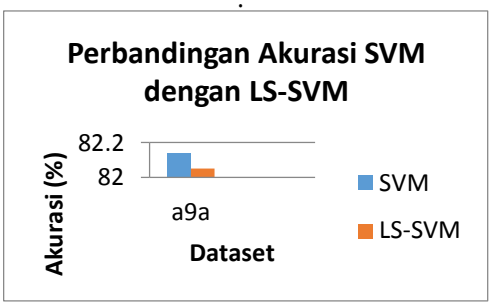
a. Nilai parameter regulasi (C) : 10 ; 100 ; 1000 ; 10000. Parameter regulasi (C) berfungsi untuk menentukan *trade off* antara margin maksimum dan kesalahan klasifikasi minimum[11].
b. Nilai parameter *bandwith* atau *gamma* (γ) : 0.5 ; 1 ; 2 ; 3 ; 4. Parameter *bandwith* atau *gamma* (γ) berfungsi untuk menentukan kedekatan antar 2 titik[12].
c. Jumlah *worker* yang digunakan untuk proses MATLAB *parallel computing toolbox* : 2 ; 3 ; 4 ; 5 ; 6.
d. Untuk percobaan LS-SVM, data yang digunakan adalah dataset a9a.

4.2 Analisis Akurasi

Berikut rata-rata akurasi terbaik pada algoritma *Support Vector Machine* (SVM) dan *Least Squares Support Vector Machine* (LS-SVM).

Tabel 4 Perbandingan Akurasi SVM dengan LS-SVM

Dataset	Akurasi SVM Sekuensial (%)	Akurasi LS-SVM Sekuensial (%)	Akurasi SVM Paralel (%)	Akurasi LS-SVM Paralel (%)
a9a	82.14	82.05	82.14	82.05
w6a	95.20	-	95.20	-



Gambar 2 Perbandingan Akurasi SVM dengan LS-SVM

Pada Gambar 2 dapat dilihat bahwa algoritma SVM sedikit lebih unggul dibandingkan dengan algoritma LS-SVM. Perbedaan akurasi SVM dengan LS-SVM pada proses sekuensial maupun paralel sebesar 0.09% untuk dataset a9a. Hal ini membuktikan bahwa kedua algoritma ini sangat kompetitif
ini disebabkan fungsi parameter γ adalah menentukan kedekatan antar 2 titik. Apabila kedekatan antar titik Pada percobaan dapat disimpulkan bahwa rata-rata nilai akurasi sangat dipengaruhi nilai dari parameter γ Hal tinggi maka bisa menyebabkan algoritma mudah atau sulit dalam mencari pemisah *hyperplane*[12]. Untuk itulah diperlukan parameter masukan yang tepat agar hasil akurasi menjadi optimal.

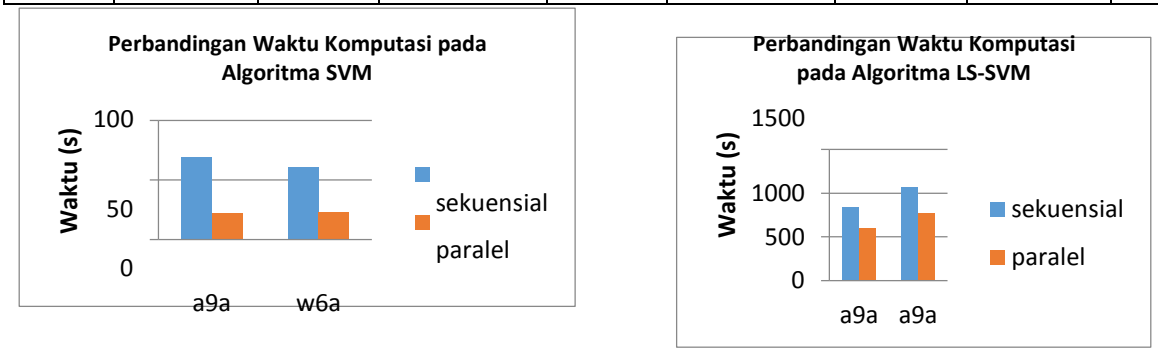
4.3 Analisis Waktu

Berikut rata-rata waktu komputasi terbaik pada algoritma *Support Vector Machine* (SVM) dan *Least Squares Support Vector Machine* (LS-SVM).

Tabel 5 Perbandingan Waktu Komputasi SVM dengan LS-SVM

Dataset	Waktu Komputasi SVM Sekuensial (s)	Ket	Waktu Komputasi SVM Paralel (s)	Ket	Waktu Komputasi LS-SVM Sekuensial (s)	Ket	Waktu Komputasi LS-SVM Paralel (s)	Ket

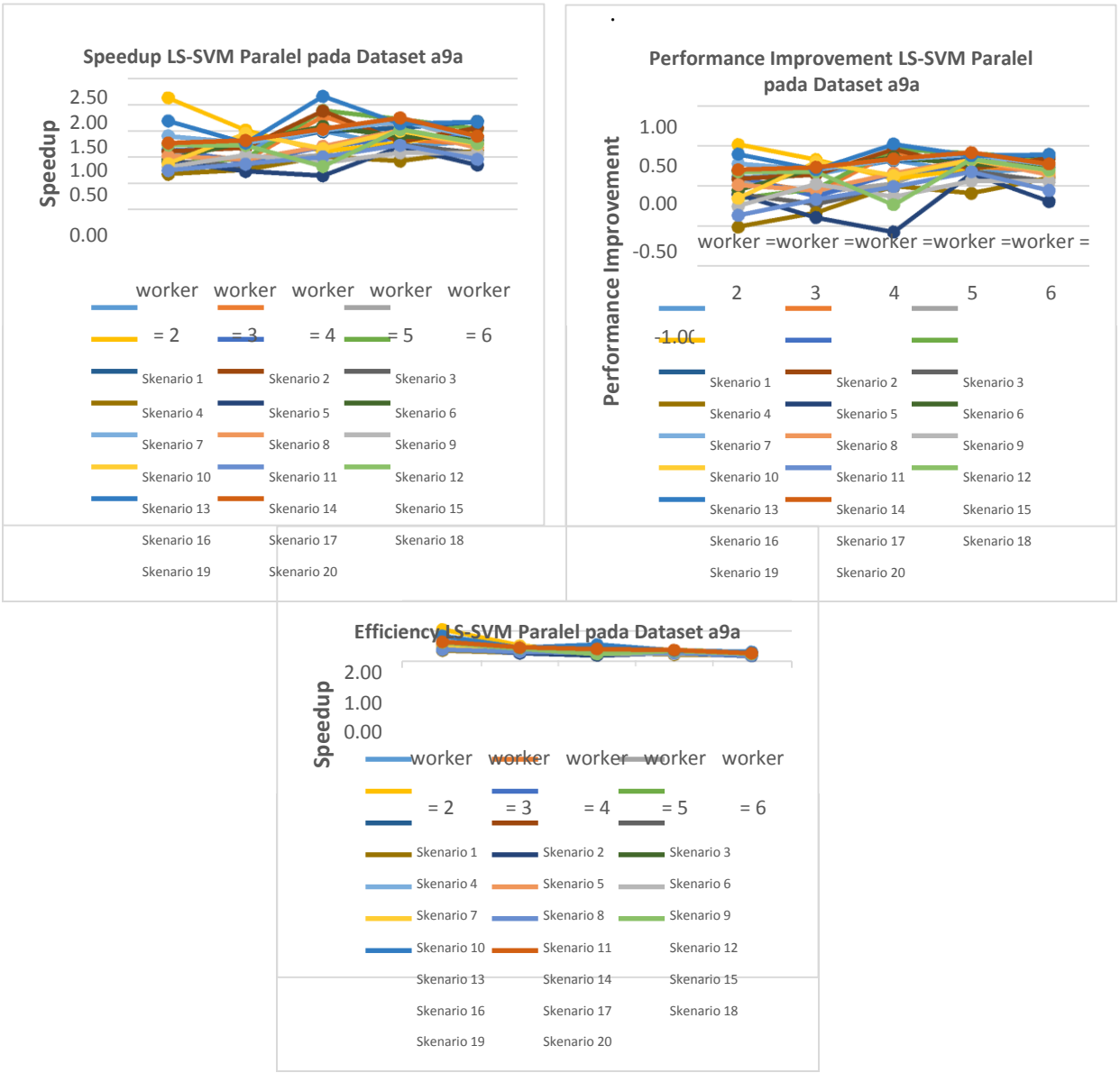
a9a	69.44	C = 10000; γ = 0.5; Akurasi = 82.14%	22.35	C = 100; γ = 0.5; Akurasi = 82.14%	843.97	C = 1000; γ = 0.5; Akurasi = 77.29%	604.66	C = 10; γ = 1; Akurasi = 77.29%; worker = 4
w6a	60.84	C = 100; γ = 0.5; Akurasi = 95.20%	22.89	C = 10; γ = 0.5; Akurasi = 95.20%	1071.79	C = 10; γ = 4; Akurasi = 82.05%	770.59	C = 10; γ = 4; Akurasi = 82.05%; worker = 6



Gambar 3 Perbandingan Waktu Komputasi

Pada gambar 3 terlihat bahwa SVM dan LS-SVM paralel memberikan waktu komputasi yang lebih cepat dibandingkan dengan SVM dan LS-SVM sekuensial.

4.4 Analisis Performansi LS-SVM Paralel



Gambar 4 Speedup, Performance improvement, Efficiency

Jumlah *worker* sama dengan 2 memiliki efisiensi paling baik.Sedangkan jumlah *worker* yang sama dengan 5 memilki rata-rata nilai *speedup* dan *performance improvement* yang lebih besar dibandingkan dengan jumlah *worker* lainnya.

5. Kesimpulan

Kesimpulan yang didapatkan dari hasil analisis pada penelitian ini adalah :

a. Cara mengimplementasikan algoritma *Support Vector Machine* (SVM) dengan menggunakan pendekatan *Graphic Processing Unit* (GPU) adalah dengan menggunakan fungsi perkalian matriks-vektor dalam CUBLAS pada fungsi kernel gaussian[13]. Sedangkan pada algoritma *Least Square Support Vector Machine* (LS-SVM) adalah dengan menggunakan fungsi sum dalam CUBLAS pada saat perhitungan akurasi. Cara ini terbukti dapat mempercepat proses komputasi dari algoritma SVM.
b. Cara mengimplementasikan algoritma *Least Square Support Vector Machine* (LS-SVM) dengan menggunakan MATLAB *Parallel Computing Toolbox* adalah dengan mengganti algoritma *for-loop* dengan menggunakan *parfor* pada fungsi kernel. Cara ini juga terbukti dapat mempercepat proses komputasi dari algoritma LS-SVM.
c. Algoritma *Support Vector Machine* (SVM) menghasilkan akurasi terbaik sebesar 82.14% dan *Least Squares Support Vector Machine* (LS-SVM) sebesar 82.05% pada dataset a9a. Kemudian untuk dataset w6a, SVM menghasilkan akurasi terbaik sebesar 95.20%. Sedangkan waktu komputasi pada algoritma paralel lebih unggul sekitar 37 detik - 302 detik dibandingkan dengan algoritma sekuensial.
d. Setelah dilakukan evaluasi performansi pada algoritma *Least Squares Support Vector Machine* (LS-SVM), terlihat bahwa jumlah *worker* sama dengan 2 memiliki efisiensi paling baik.Sedangkan jumlah *worker* yang sama dengan 5 memilki rata-rata nilai *speedup* dan *performance improvement* yang lebih besar dibandingkan dengan jumlah *worker* lainnya. Pada penelitian ini juga didapatkan nilai minus pada *performance improvement* yang disebabkan proses sikronisasi *memory* saat pembagian perintah pada MATLAB *parallel computing toolbox* memerlukan waktu yang terkadang lama sehingga dapat disimpulkan bahwa rendahnya tingkat kestabilan dari perangkat keras yang digunakan pada saat uji coba sangat mempengaruhi waktu komputasi.
e. Pada penelitian ini, telah dibuktikan bahwa parameter γ mempengaruhi hasil akurasi. Hal ini disebabkan fungsi parameter γ adalah menentukan kedekatan antar 2 titik. Apabila kedekatan antar titik tinggi maka bisa menyebabkan algoritma mudah atau sulit dalam mencari pemisah *hyperplane*[23].

Daftar Pustaka :

[1] Harvey, Jesse Patrick, "*GPU acceleration of object classification algorithms using NVIDIA CUDA*" (2009). Thesis. Rochester Institute of Technology.
[2] Polat, K and Güneş, S. (2007). "*Breast cancer diagnosis using least squares support vector machine*". Science Direct, vol 17, pp. 694-701, July 2007.
[3] Nugroho, A.S., Witarto, A.B, dan Handoko, D. (2003). "*Support Vector Machine, Teori dan Aplikasinya dalam Bioinformatika*". Copyright www.ilmukomputer.com
[4] Santosa, B. (2007). "*DATA MINING: Teknik Pemanfaatan Data untuk Keperluan Bisnis*". Yogyakarta : Graha Ilmu.
[5] Li, Qi, "*Fast Parallel Machine Learning Algorithms for Large Datasets Using Graphic Processing Unit*" (2011). VCU Theses and Dissertations. Paper 2625.
[6] Tan, Pang-Ning; Steinbach,Michael; Kumar ,Vipin . (2006). "*Introduction to Data Mining*". United States of America : Pearson Education, Inc.
[7] Novianti, F.A., dan Purnami, S.W. (2012). "*Analisis Diagnosis Pasien Kanker Payudara Menggunakan Regresi Logistik dan Support Vector Machine* (SVM)", Journal of Science and Art of ITS, Volume 1. Surabaya : Institut Teknologi Surabaya.
[8] Ahmad Firdaus Ahmad Fadzil, Noor Elaiza Abdul Khalid, dan Mazani Manaf. (2013), *Scaling Performance of Task-Intensive Applications via Mapreduce Parallel Processing*, Faculty of Computer and Mathematical Science, UiTM Shah Alam Selangor, Malaysia..
[9] Dataset yang digunakan untuk penelitian ini, diambil dari situs (*http://ntucsu.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html* ,diakses 24 Juli 2015)
[10] W.Suh, Jung; Youngmin, Kim. (2014). "*Accelerating MATLAB with GPUs A Primer with Examples*". Elsevier Inc.
[11] Handayani, V. P. (2013). "*Klasifikasi Penyakit Kanker Prostat menggunakan Principal Component Analysis (PCA) dan Least Squares Support Vector Machine (LS-SVM)*". Laporan Tugas Akhir. Jurusan Ilmu Komputasi Universitas Telkom.
[12] Pasaribu, H. E., Atastina, I., Shaufiah (2010). "*Klasifikasi Dokumen WEB menggunakan Version Space Support Vector Machine*" KNS&I10-059. Bandung : Universitas Telkom.
[13] Carpenter, A. (2009). "*CUSVM: A CUDA Implementation of Support Vector Classification and Regression*".