Rekayasa Hijau: Jurnal Teknologi Ramah Lingkungan Volume 4 | Nomor 3
ISSN [e]: 2579-4264 | DOI: https://doi.org/10.26760/jrh.v4i3.157-166 November 2020

Identifikasi Jenis *Font* Menggunakan Metode *Genetic Modified K-Nearest Neighbor*

Yusup Miftahuddin[1], Sofia Umaroh[2], Agistya Anugrah Dwiutama[3]
[1,3]Informatika, Institut Teknologi Nasional Bandung
[2]Sistem Informasi, Institut Teknologi Nasional Bandung
Email: yusufm@itenas.ac.id[3], sofia.umaroh@itenas.ac.id[2], agistyaanugrah@gmail.com[3]

Received 24 July 2020 | Revised 26 August 2020 | Accepted 31 August 2020

ABSTRAK

Font adalah kumpulan karakter lengkap yang memiliki ukuran dan gaya. Saat melihat desain atau aplikasi, desainer grafis dan pengembang front-end terinspirasi untuk menggunakan font yang sama. Namun font telah menjadi gambar atau aplikasi dan sulit untuk mengetahui jenis font yang digunakan. Genetic Modified K-Nearest Neighbor (GMKNN) merupakan metode gabungan dari Modified K-Nearest Neighbor (MKNN) dan Genetic Algorithm (GA) untuk menentukan k Optimal. Dalam penelitian ini, sebuah sistem akan dirancang untuk mengenali jenis font sans-serif menggunakan metode Genetic Modified K-Nearest Neighbor (GMKNN) untuk mengukur akurasi dan waktu komputasi. Kinerja sistem dalam proses mengidentifikasi jenis font mendapat nilai presisi rata-rata 74,6%. Nilai akurasi adalah 72,2% dan nilai recall 72%. Dari hasil presisi dan recall yang diperoleh nilai f-measure sebesar 72,2% dan rata-rata waktu komputasi untuk satu karakter diperoleh adalah 945,04190395673 detik

***Kata kunci**: Pengolahan Citra Digital, Identifikasi Font, Pengenalan Pola dan GMKNN*

ABSTRACT

Fonts is a complete collection of characters that have size and style. When looking at a design or an application, graphic designers and front-end developers are inspired to use the same font. Unfortunately, the font has become an image or an application so it is difficult to identify the font types. Genetic Modified K-Nearest Neighbor (GMKNN) is a hybrid method of Modified K-Nearest Neighbor (MKNN) and Genetic Algorithm (GA) to determine optimal k, it also reduces the complexity of MKNN and improves the classification accuracy. In this research, a system is designed to recognize font sans-serif types using GMKNN method to measure accuracy and time computation. The performance of the system in the process of identifying font types gets an average precision value of 74.6%. The recall and accuracy values are 72% and 72,2%, respectively. Based on the results of precision and recall obtained, the system gets an f-measured value of 72.2% and time obtained for one character is 945,04190395673 seconds.

***Keywords**: Image Proccesing, Font Identify, Pattern Recognition, and GMKNN*

1. PENDAHULUAN

Font adalah sebuah kumpulan karakter lengkap yang mempunyai ukuran dan gaya. Sedangkan dalam komputer, *font* merupakan *file* data elektronik yang mengandung sebuah kumpulan dari elemen penulisan (*glyph*), karakter-karakter dan simbol-simbol [1]. Sebagian besar jenis *font* yang digunakan dalam pencetakan dan *user interface* milik dua gaya *font* yaitu *serif* dan *sans-serif* [2]. *Font* ini dipilih karena kedua jenis *font* banyak digunakan pada *website* dan merupakan jenis *font* standar dari banyak *developer website* yang paling populer.

Ketika melihat sebuah desain hasil karya atau aplikasi orang lain, para *graphic designer* dan *frontend developer* terinspirasi untuk membuat hal yang sama, atau menggunakan *font* yang sama. Namun *font* tersebut sudah menjadi gambar atau sebuah aplikasi yang sulit untuk diketahui jenis dari *font* tersebut.

Pada penelitian sebelumnya dilakukan modifikasi metode *K-Nearest Neighbor* (KNN) untuk menentukan *k* optimal pada tahap pengklasifikasian dengan cara menambahkan algoritma *Genetic Algorithm* (GA) dengan hasil menunjukkan bahwa metode yang digunakan tidak hanya mengurangi kompleksitas *K-Nearest Neighbor* (KNN), tetapi juga meningkatkan akurasi klasifikasi [3]. Pada penelitian lainnya dilakukan modifikasi metode *Modified K-Nearest Neighbor* (MKNN) untuk menentukan *k* Optimal pada tahap pengklasifikasian dengan cara menambahkan algoritma *Genetic Algorithm* (GA), keunggulan dari metode GMKNN tidak perlu mencoba nilai *K* satu-persatu tetapi nilai *k* di dapatkan secara otomatis [4].

Penelitian lainnya dalam melakukan identifikasi jenis *font*, menggunakan metode *Support Vector Machine* (SVM) dan *K-Nearest Neighbor* (KNN) dengan menghasilkan tingkat akurasi 80% untuk *Support Vector Machine* (SVM) dan 75,5% untuk *K-Nearest Neighbor* (KNN) [5]. Berdasarkan penelitian-penelitian tersebut, parameter yang tidak diuji adalah parameter waktu komputasi. Pada penelitian ini, metode *Genetic Modified K-Nearest Neighbor* (GMKNN) diterapkan dalam mengidentifikasi jenis *font sans-serif* dalam bahasa latin yang berbentuk citra yang bertujuan untuk mengukur akurasi dan waktu komputasi. Pengukuran waktu komputasi diuji dengan beberapa skema yaitu panjang karakter dan jumlah data latih.

2. METODOLOGI

2.1 *Greyscaling*

Citra *grayscale* adalah citra yang memiliki dua warna yaitu hitam sebagai warna minimal (0) dan warna putih (255), sehingga berwarna keabu-abu [6]. Proses *grayscaling* mengubah citra berwarna menjadi keabuan dengan cara mengurangi dimensi yang dimiliki oleh citra, dengan melakukan pemetaan citra tiga kanal warna yaitu RGB menjadi hanya satu kanal warna yaitu warna keabuan [7]. Terdapat berbagai cara dalam melakukan proses grayscaling, di antaranya adalah menggunakan Persamaan 1.

$$Grayscale = 0.21 * \mathrm{R} + 0.72 * \mathrm{G} + 0.0 * \mathrm{B} \quad (1)$$

Dimana, nilai R (Red) adalah nilai piksel berwarna merah, G (Green) adalah nilai piksel berwarna hijau, dan B (Blue) adalah nilai pikel berwarna biru.

2.1 *Low Pass Filtering*

Low Pass Filter adalah suatu filter yang digunakan untuk menyeleksi nilai piksel dari citra. Filter ini mempunyai sifat menghilangkan yang berfrekuensi tinggi dan meloloskan yang berfrekuensi rendah [8]. Pada penelitian ini kernel yang dipakai merupakan kernel dari *low pass filter* pada Persamaan 2.

$$K = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \quad (2)$$

2.2 *Sharpening (High Pass Filter)*

Sharpening atau *High Pass Filter* digunakan untuk memperjelas detail dari citra yang menghasilkan citra menjadi lebih tajam[9]. Pada penelitian ini kernel yang dipakai merupakan kernel dari *sharpening* pada Persamaan 3.

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (3)$$

2.3 *Otsu thresholding*

Otsu thresholding merupakan metode sederhana dalam segmentasi, sehingga dapat lebih mudah dalam melakukan pembagian wilayah-wilayah yang homogen berdasarkan kriteria keserupaan untuk mengenali objek [10]. *Otsu thresholding* mengubah citra *grayscale* yang memiliki nilai piksel 0-255 menjadi citra biner yang hanya memiliki 2 nilai piksel yaitu 0 (hitam) dan 255 (putih) dengan menentukan suatu ambang (*Threshold*). Nilai *Threshold* dari metode ini didapatkan dengan menggunakan Persamaan 4.

$$p(i) = \frac{n_i}{N}, p(i) \geq 0, \sum_1^{256} p(i) = 1 \quad (4)$$

Setelah didapatkan nilai *Threshold* maka kemudian citra dapat di kelompokan menjadi dua kelas yaitu : 0(hitam) dan 255(putih) dengan menggunakan Persamaan 5.

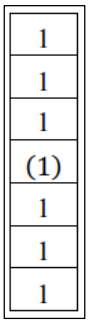
$$f = \begin{cases} 255, apabila\ f(x,y) > T \\ 0, apabila\ f(x,y) \leq T \end{cases} \quad (5)$$

2.4 Segmentasi *Mathematical Morphology*

Tahapan segmentasi digunakan untuk memisahkan setiap karakter/huruf yang ada pada citra. Untuk menemukan lokasi baris, kata, dan karakter/huruf dari citra memanfaatkan morfologi matematika menggunakan operasi dilasi. Dua proses utama tersebut menggunakan matriks *structuring elements* vertikal dan horizontal [11]. Matriks *structuring elements* horizontal berfungsi untuk menebalkan objek terhadap sumbu x, sedangkan matriks *structuring elements* vertikal berfungsi menebalkan objek terhadap sumbu y. Setelah dilakukan operasi dilasi menggunakan kedua buah matriks maka dilakukan segmentasi dengan menemukan batas vertikal dan horizontal dari citra biner. Sehingga nilai piksel-piksel tersebut merepresentasikan suatu objek.

1	1	1	(1)	1	1	1

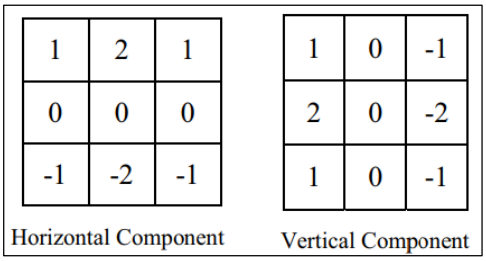
Gambar 1. Tahapan *Matriks Structuring Elements Horizontal*



Gambar 2. Tahapan *Matriks Structuring Elements Vertikal*

2.5 *Directional Feature Extraction*

Ekstrasi Ciri *Directional Feature Extraction* merupakan metode yang menganalisis ciri arah dari objek [12]. Ciri yang digunakan oleh metode ini adalah nilai gradien dari setiap piksel pada objek. Gradien adalah nilai kuantitas vektor yang merupakan komponen arah dan dihitung dengan melibatkan kedua arah baik arah horizontal dan arah vertikal [13]. Untuk menentukan arah horizontal dan vertikal dari citra dengan menggunakan kernel operator *sobel* pada Gambar 3



Gambar 3. Tahapan *Matriks Strel Vertikal*

Persamaan untuk mendapatkan nilai arah horizontal ditunjukkan pada Persamaan 6 dan vertikal ditunjukkan pada Persamaan 7.

$$Gx(a,b) = I(a-1,b-1) + 2 * I(a-1,b) + I(a-1\ b+1) - I(a+1,b-1) - 2 * I(a+1,b) - I(a+1,b+1) \quad (6)$$

$$Gy(a,b) = I(a-1,b-1) + 2 * I(a,b-1) + I(a+1,b-1) - I(a-1,b+1) - 2 * I(a,b+1) - I(a+1,b+1) \quad (7)$$

Untuk mendapatkan menilai arah gradien piksel (g) dengan menghitung nilai inverse tangent dari hasil pembagian nilai gradien vertikal dengan nilai gradien horizontal menggunakan Persamaan 8.

$$g = \tan - 1[Gy(a,b)/Gx(a,b)] \quad (8)$$

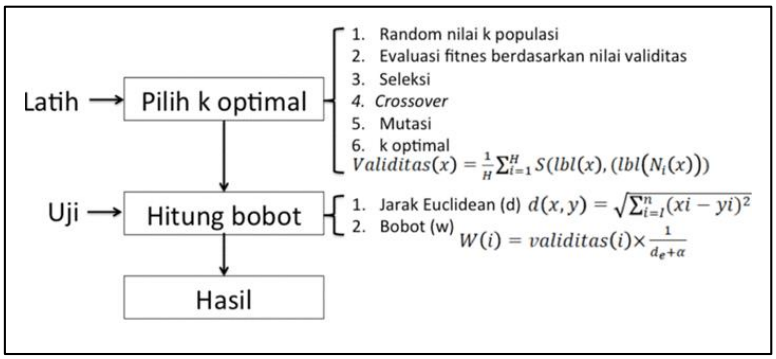
Kemudian dicari nilai direksi, nilai g yang didapatkan sebelumnya akan bernilai dari 0 sampai 2π yang kemudian dicari nilai arahnya berdasarkan Tabel 1.

Tabel 1. Nilai Diereksi

Gradient Values (g)	Direction	Gradient Values (g)	Direction
G = -1	0	π <g <= 7 π/6	7
0 <= g <= π/6	1	7 π/6 < g <= 4 π/3	8
π/6 < g <= π/3	2	4 π/3 <g <= 3 π/2	9
π/2 < g <= 2 π/3	3	3 π/2 <g <= 5 π/3	10
π/2 < g <= 2 π/3	4	5 π/3 < g <= 11 π/6	11
2 π/3 < g <= 5 π/6	5	11 π/6 < g<= 2 π	12
5 π/6 < g <= π	6	--	--

2.6 *Genetic Modified K-Nearest Neighbor* (GMKNN)

Algoritma *Modified K-Nearest Neighbor* (MKNN) masih belum mampu mengatasi permasalahan algoritma *K-Nearest Neighbor* (KNN) dalam menentukan nilai *k* yang masih bias. Oleh karena itu dilakukan optimasi menggunakan algoritma *Genetic Algorithm* dalam menentukan nilai *k* yang bertujuan mengatasi kelemahan tersebut dan nilai *k* didapatkan berdasarkan nilai kromosom, Algoritma tersebut dinamakan *Genetic Modified K-Nearest Neighbor* (GMKNN) yang digunakan sebagai model pada saat klasifikasi [4]. Pada algoritma *Genetic Modified K-Nearest Neighbor* terbagi menjadi dua tahapan yaitu tahapan *Genetic Algorithm* dan *Modified K-Nearest Neighbor.*



Gambar 1. Tahapan *Genetic Modified K-Nearest Neighbor*

2.6 Dataset

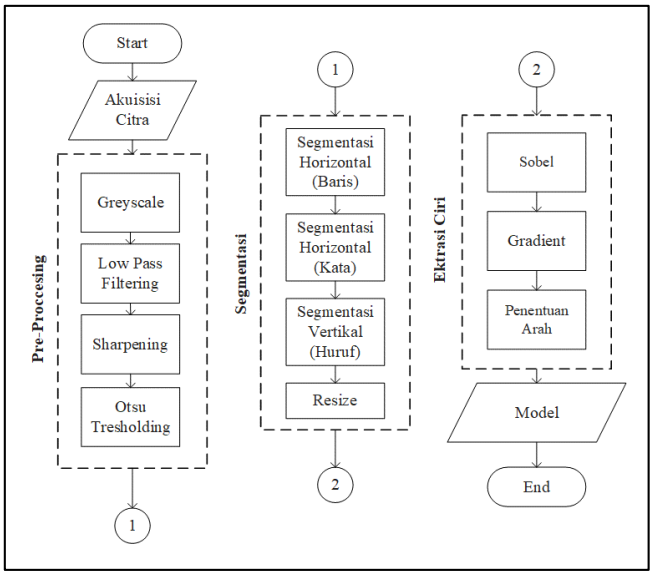
Dataset yang digunakan merupakan keluarga *font sans-serif. Sans-serif* merupakan jenis *font* tanpa ekor atau hiasan. Dataset yang digunakan terdiri dari lima jenis *font* yaitu *lato, montserrat, raleway, tahoma dan roboto* dengan ukuran citra 30x30 piksel menggunakan seluruh huruf/karakter dengan *style Reguler*, *Bold*, dan *Italic* dengan jumlah total dataset sebanyak 4680 citra. *Dataset* ini digunakan karena tidak memiliki ekor/hiasan sehingga hanya bentuk huruf yang membedakannya.

Tabel 1. Sample Dataset Font

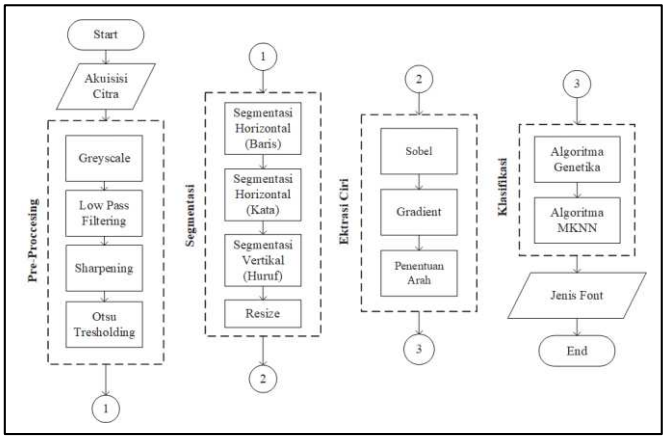
No	Contoh Gambar	Font	Jumlah Dataset
1	a	Lato	936
2	a	Montserrat	936
3	a	Raleway	936
4	a	Roboto	936
5	a	Tahoma	936
		Total	4680

2.7 Diagram Alir

Dalam penelitian ini terdapat 2 tahapan untuk mengidentifikasi jenis *font* yaitu tahapan training dan tahapan klasifikasi. Tahapan traning merupakan tahapan untuk membuat model yakni proses pada training dimulai dari *inputan* yang merupakan sebuah citra yang di dalamnya terdapat huruf/karakter dalam bahasa latin.



Gambar 2. Diagram Alir *Training*



Gambar 3. Diagram Alir *Testing*

Citra tersebut akan melalui proses *pre-processing*, segmentasi, dan ekstrasi ciri. Isi dari model merupakan hasil ekstrasi ciri yang akan digunakan pada tahapan klasifikasi alur proses *training* yang dapat dilihat pada Gambar 2. Sedangkan pada tahapan klasifikasi digunakan untuk menguji model yang telah di buat pada tahapan training. Proses klasifikasi dapat dilihat pada Gambar 3. *Input* dari sistem adalah sebuah citra yang di dalamnya terdapat huruf/karakter dalam bahasa latin. Citra tersebut akan melalui proses *pre-processing*, segmentasi, ekstrasi ciri, dan klasifikasi kemudian sistem akan mengidentifikasi jenis *font*.

3. HASIL DAN PEMBAHASAN

Pada algoritma *Genetic Modified K-Nearest Neighbor* tahapan pertama yaitu tahapan *Genetic Algorithm algorithm* merupakan tahapan untuk menentukan nilai *k* optimal. Pada tahapan awal di mana menentukan nilai kromosom secara acak yang nantinya di jadikan nilai *k* kemudian dihitung nilai *fitness* setelah mendapatkan nilai *fitness* melakukan *crossover* dan dilakukan mutasi untuk menghasilkan kromosom baru kemudian hasil nilai kromosom awal, *crossover* dan mutasi di urutkan berdasarkan nilai *fitness* terbesar setelah itu nilai *fitness* terbesar akan di gunakan sebagai *parent* untuk generasi selanjutnya dan akan berhenti sesuai generasi yang di tentukan dan menghasilkan nilai *k* optimal. Nilai *k* optimal di gunakan pada metode *Modified K-Nearest Neighbor* di mana langkah pertama yaitu menghitung jarak data uji pada data training kemudian dihitung nilai bobot pada setiap kelas yang akan menentukan hasil prediksi. Pada penelitian pengujian dilakukan dengan cara menguji tiap citra uji untuk mengukur kinerja sistem dan waktu komputasi. Kinerja sistem dilakukan dengan mengukur *accuracy pa*da Persamaan (8), *precision* pada Persamaan (9), *recall* Persamaan (10) dan untuk mengukur *F Measure* menggunakan Persamaan (11) [14]. Sedangkan waktu komputasi merupakan waktu yang dibutuhkan dalam memproses citra uji. Untuk mengukur kinerja sistem menggunakan persamaan berikut :

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}\ x\ 100\% \quad (8)$$

$$Precision = \frac{TP}{TP+FP}\ x\ 100\% \quad (9)$$

$$Recall = \frac{TP}{TP+FN}\ x\ 100\% \quad (10)$$

$$F\ Measure = 2\ x\ \frac{(Precision\ x\ Recall)}{(Precision+Recall)} \quad (11)$$

Keterangan :

TP (True Positive) = Correct Result

FP (False Positive) = Unexpected Result

FN (False Negative) = Missing Result

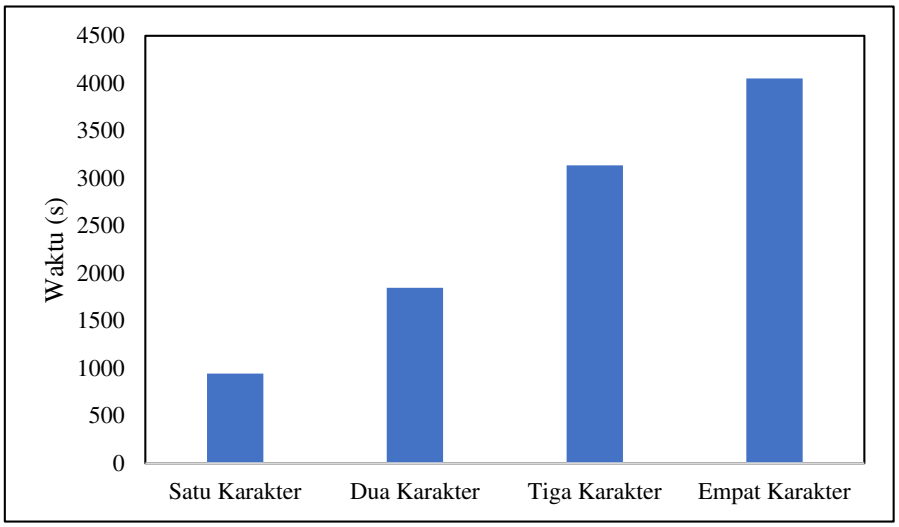
TN (True Negative) = Correct Absence of result

3.1 Pengujian Waktu terhadap Jumlah Karakter

Pengujian sistem dalam mengidentifikasi *font* cetak yang terdapat pada citra teks. Citra yang diuji dengan lima jenis *font* yang berbeda yaitu *lato, montserrat, raleway, tahoma* dan *roboto* dan gaya berbeda-beda. Pada pengujian waktu digunakan dengan satu karakter dengan, dua karakter, tiga karakter, dan empat karakter dengan jumlah masing-masing citra. Hasil pengujian waktu dapat dilihat pada Tabel 2 dan Gambar 4 merupakan hasil perbandingan dari setiap jumlah karakter.

Tabel 2. Pengujian Jumlah Karakter

No.	Font	Jumlah Data Uji	Waktu
1	1 Karakter	15	945,04190395673
2	2 Karakter	15	1847,7643669764
3	3 Karakter	15	3136,6385683695
4	4 Karakter	15	4051,6246039868
	Rata -Rata		2495,2673608224



Gambar 4. Grafik Pengujian Waktu Pada Jumlah Karakter

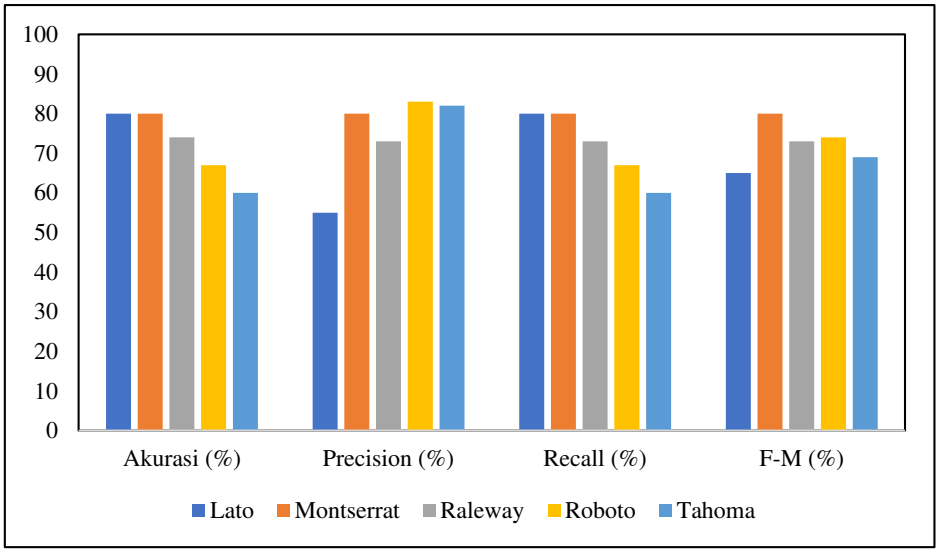
Berdasarkan hasil pengujian waktu terhadap jumlah karakter yang disajikan pada Gambar 4, semakin bertambahnya jumlah karakter maka semakin lama waktu yang dibutuhkan untuk mengidentifikasi *font*. Setelah dilakukan pengujian sebanyak 15 kali, dalam mengidentifikasi satu karakter dengan jenis *font* yang berbeda-beda, waktu komputasi yang dibutuhkan adalah 647,7898898124695 detik sampai 1234,2329063415527 detik dengan rata-rata 945,04190395673 detik.

3.2 Pengujian Kinerja Sistem

Pada pengujian kinerja sistem dengan digunakan data uji dengan jenis *font* berbeda-beda masing-masing jenis *font* menggunakan 15 citra uji dengan gaya *regular*, *bold*, dan *italic* sehingga total percobaan sebanyak 75 citra uji. Tabel 6 merupakan representasi dari masing-masing percobaan yang telah dilakukan dan pada Gambar 5 merupakan perbandingan nilai akurasi, *precision*, *recall*, dan *F-Measure* yang di representasikan dalam bentuk grafik.

Tabel 6. Pengujian Kinerja Sistem

No	Font	Akurasi (%)	Precision (%)	Recall (%)	F-M (%)
1	Lato	80	55	80	65
2	Montserrat	80	80	80	80
3	Raleway	74	73	73	73
4	Roboto	67	83	67	74
5	Tahoma	60	82	60	69
Rata-rata		72,2	74,6	72	72,2



Gambar 5. Grafik Kinerja Sistem

Berdasarkan hasil kinerja sistem pada Gambar 5, semakin bertambahnya jumlah karakter maka semakin lama waktu yang dibutuhkan untuk mengidentifikasi *font*. Setelah dilakukan pengujian sebanyak 15 kali, dalam mengidentifikasi satu karakter dengan jenis *font* yang berbeda-beda, waktu komputasi yang dibutuhkan adalah 647,7898898124695 detik sampai 1234,2329063415527 detik dengan rata-rata 945,04190395673 detik.

4. KESIMPULAN

Setelah di lakukan pengujian dengan panjang karakter yang berbeda-beda. Hasil menunjukkan bahwa waktu komputasi tercepat didapatkan oleh satu karakter, semakin banyak karakter pada sebuah citra teks semakin lama waktu komputasi karena sistem mengidentifikasi untuk setiap karakter. Pengujian kinerja sistem dengan jenis *font* yang berbeda-beda, hasil menunjukkan bahwa nilai rata-rata nilai akurasi 72,2%

nilai *precision* 74,6 dan nilai recall 72% dan nilai *F-Measure* 72,2%. Nilai akurasi tertinggi didapatkan oleh jenis *font lato* dan *montserrat* dengan nilai 80%. Nilai *precision* tertinggi didapatkan oleh jenis *font roboto* 83% dan terendah adalah kelas *lato* dengan nilai 55% sedangkan nilai *recall* tertinggi didapatkan oleh kelas *lato* dan *montserrat* dengan nilai 80% dan nilai *precision* terkecil adalah kelas tahoma dengan nilai 60% dan nilai *F-Measure* teretinggi adalah kelas *montserrat* dengan nilai 80% dan terendah dengan nilai 65% adalah kelas *lato* faktor-faktor yang mempengaruhi akurasi adalah *dataset* dan nilai kromosom yang digunakan sebagai nilai *k*.

DAFTAR PUSTAKA

[1] A. Haryono, "Studi Pembentukan Huruf Font Dengan Kurva Bezier," *Teknika,* pp. 69-78, 2014.

[2] A. Altaboli, "Investigating the effects of font styles on perceived visual aesthetics of website interface design.," *International Conference on Human-Computer Interaction,* pp. 549-554, 2013.

[3] N. Suguna dan K. Thanushkodi, "An Improved k-Nearest Neighbor Classification Using Genetic Algorithm," *International Journal of Computer Science,* pp. 18-21, 2010.

[4] S. Mutrofin, A. Izzah, A. Kurniawardhani dan M. Masrur, "Optimasi teknik klasifikasi modified k nearest neighbor menggunakan algoritma genetika," *Jurnal Gamma,* 2015.

[5] Bharath, V., dan Rani, N. S. "A Font style classification system for English OCR," *2017 International Conference on Intelligent Computing and Control (I2C2),* no. However for the font style/ size, 2017.

[6] S. E. Indraani, I. D. Jumaddina dan S. R. S. Sinaga, "Implementasi Edge Detection Pada Citra Grayscale dengan Metode Operator Prewitt dan Operator Sobel," *Majalah Ilmiah Inti,* 2014.

[7] G. Lanaro, Q. Nguyen dan S. Kasampalis, Learning Path Advanced Python Programming, Birmingham: Packt Publishing, 2019.

[8] G. A. Prakoso, "Penerapan Metode Low Pass Filter (LPF) untuk Mengurangi Derau pada Citra Magnetic Resonance Image (MRI)," Universitas Muhammadiyah Surakarta, Surakarta, 2017.

[9] Tayja, Y. D. Lestari dan U. Khair, "Aplikasi Perbaikan Citra Digital dalam Pengolahan Citra dengan Menggunakan Metode Smoothing Filter dan Sharpening Filter," *E-Journal UPMI,* pp. 313-318, 2018.

[10] A. T. Utami, "Implementasi Metode Otsu Tresholding Untuk Segmentasi CItra Daun," Universitas Muhammadiyah Surakarta, Sukoharjo, 2017.

[11] N. Trisnadik, A. Hidayatno dan R. R. Isnanto, "Pendeteksian Posisi Plat Nomor Kendaraan Menggunakan Metode Morfologi Matematika," *TRANSIENT vol. 2,* pp. 56-62, 2013.

[12] F. Z. Putri, B. Irawan dan U. A. Ahmad, "Perancangan Dan Implementasi Directional Feature Extraction Dan Support Vector Machines Untuk Menerjemahkan Kata Denga Pengenalan Huruf Hiragana Dalam Bahasa Jepang Ke Bahasa Indonesia Berbasis Android," *Jurnal Teknik Elektro,* 2017.

[13] A. Aggarwal, K. Singh dan K. Singh, "Use of Gradient Technique for Extracting Features from Handwritten Gurmukhi Characters and Numerals," *Procedia Computer Science 46,* pp. 1716-1723, 2015.

[14] J. Pardede dan M. G. Husada, "Comparison of Vsm, Gvsm, and Lsi In Information Retrieval For Indonesian Text," *Jurnal Teknologi 78 (5-6),* 2015.