Naskah Publikasi

TUGAS AKHIR

PENGENALAN POLA TANDA TANGAN MENGGUNAKAN HISTOGRAM OF ORIENTED GRADIENTS DAN BACKPROPAGATION

RENDI BUDIYANTO
5130411332

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN ELEKTRO
UNIVERSITAS TEKNOLOGI YOGYAKARTA
YOGYAKARTA
2020

Naskah Publikasi

PENGENALAN POLA TANDA TANGAN MENGGUNAKAN HISTOGRAM OF ORIENTED GRADIENTS DAN BACKPROPAGATION

Disusun Oleh

Rendi Budiyanto

5130411332

Telah disetujui oleh pembimbing

Pembimbing

Dr. Enny Itje Sela, S.Si., M.Kom.

Tanggal 30 September 2020

Pengenalan Pola Tanda Tangan Menggunakan Histogram of Oriented Gradients dan Backpropagation

Rendi Budiyanto
Program Studi Informatika, Fakultas Teknologi Informasi dan Elektro
Universitas Teknologi Yogyakarta
Jl. Ringroad Utara Jombor, Sleman, Yogyakarta
E-Mail: rend.budian@gmail.com

ABSTRAK

Tanda tangan merupakan sarana untuk mengidentifikasi seseorang. Keberadaan tanda tangan dalam sebuah dokumen menandakan bahwa pihak yang menandatangani dokumen tersebut sudah mengetahui dan menyetujui isi suatu dokumen. Identifikasi tanda tangan merupakan hal yang penting dalam validasi sebuah dokumen. Mengenali suatu tanda tangan secara visual memang cukup mudah dilakukan. Namun, jika dokumen yang divalidasi berdasarkan tanda tangan pihak yang berwenang berjumlah banyak, maka hal ini akan membutuhkan waktu yang lebih banyak. Backpropagation adalah algoritma yang dapat digunakan untuk mengenali, mengklasifikasi, dan memprediksi suatu data. Proses pembelajaran algoritma backpropagation termasuk kategori metode pembelajaran terbimbing (supervised learning). Dalam penelitian ini backpropagation dikombinasikan dengan batch normalization untuk meningkatkan kinerjanya. Dataset yang digunakan diperoleh dari CEDAR atas 55 responden dengan 24 data asli dan 24 data palsu. Dataset kedua, hasi foto dari 5 responden masing-masing 16 data asli dan 16 data palsu. Preprocessing dilakukan untuk menyelaraskan ukuran menjadi 100x400 piksel. Ekstraksi fitur menggunakan Histogram of Oriented Gradients dengan bin 9, sel 20x20, dan blok 1x1 yang menghasilkan 900 vektor. Pengujian dilakukan dengan mengambil 25% dari masing-masing dataset, 660 data dari dataset pertama dan 40 untuk dataset kedua. Hasil yang didapat dari beberapa percobaan, untuk dataset pertama tertinggi sebesar 91.36 %, sedangkan dataset kedua sebesar 87.5%.

Kata kunci: *Tanda Tangan, Histogram of Oriented Gradients, Backpropagation, Batch Normalization*

1. PENDAHULUAN

Tanda tangan (*signature*) adalah sebuah tanda (*sign*) atau simbol yang legal dan merupakan gambaran asli dari pemiliknya [1]. Tanda tangan bersifat unik untuk setiap penanda tangan dan secara umum tanda tangannya relatif stabil. Namun, tanda tangan seseorang bisa berubah-ubah dipengaruhi oleh waktu, kebiasaan, umur dan keadaan mental [2]. Tanda tangan dapat difungsikan sebagai segel atau legalitas suatu dokumen. Dalam kepentingan tertentu tanda tangan menjadi rentan akan pemalsuan dan penyalahgunaan yang menyebabkan perlu dilakukannya verifikasi tanda tangan [3].

Penelitian sebelumnya tentang mengenali pola tanda tangan menggunakan metode *backpropagation* memiliki akurasi 68% [4]. Penelitian ini menggunakan matrik (10 x 10) sebagai input jaringan dan 1 *output layer*. Tanda tangan dengan kerumitan tertentu, matrik tersebut belum mampu mendeskripsikan citra, output jaringan bervariasi dengan hanya 1 *output layer* hasil yang didapat belum maksimal.
Penelitian selanjutnya metode *backpropagation* dapat memverifikasi tanda tangan dengan akurasi 98,5% dari tanda tangan yang sesuai dengan kelasnya dan untuk tanda tangan baru yang dipalsukan mampu mengenali dengan akurasi 82% [1]. Penelitian ini menggunakan *input layer* berjumlah 100 didapat dari hasil ekstraksi ciri menggunakan *feature points based on vertical grid and horizontal grid*, sedangkan *output layer* berjumlah 1. Penelitian ini bertujuan untuk mengklasifikasikan tanda tangan sesuai dengan kelasnya, sehingga tanda tangan palsu masih dikenali sebagai kelas yang bersangkutan.

Penelitian lain tentang mengenali tanda tangan dengan kombinasi metode *Histogram of Oriented Gradients* (HOG) dan *Support Vector Machine* (SVM) menghasilkan akurasi sebesar 88% [2]. Penelitian ini, HOG memproses citra tanpa proses segmentasi atau *crop* posisi tanda tangan terlebih dulu, sehingga menurunkan akurasi pengenalan.

Berdasarkan penjelasan di atas, penulis tertarik untuk membuat aplikasi yang dapat mengenali tanda tangan manusia. Pada penelitian ini, penulis menggunakan metode ekstraksi fitur *Histogram of Oriented Gradients* dan *Backpropagation* digunakan untuk klasifikasi. Aplikasi yang dibuat ini nantinya dapat dimanfaatkan untuk membantu atau mempermudah pihak lain. Sebagai contoh, untuk melakukan verifikasi presensi di suatu perguruan

tinggi. Dengan adanya sistem ini diharapkan dapat mengurangi kecurangan mahasiswa dalam melakukan presensi tersebut.

2. LANDASAN TEORI

2.1. *Preprocessing*

Preprocessing merupakan tahap pengolahan citra yang dilakukan sebelum pengolahan lebih lanjut pada citra. *Prepocessing* bertujuan untuk meningkatkan kemungkinan keberhasilan pada tahap berikutnya. Beberapa operasi yang dilakukan pada tahap *prepocessing* yaitu:

a. *Grayscale*

Grayscale atau skala keabuan merupakan citra dengan efek keabu-abuan. *Grayscale* dapat diperoleh dengan mengalikan intensitas setiap kanal citra berwarna atau *Red Green Blue* (RGB) dengan nilai tertentu, kemudian dijumlahkan. Formula dapat dilihat pada persamaan (1) [5].

$$g = 0.2989 * R + 0.5870 * G + 0.1140 * B \quad \textbf{(1)}$$

Keterangan:
R, G, B = kernel warna
g = hasil *grayscaling*

b. *Threshold*

Threshold atau ambang batas merupakan tahapan untuk merubah nilai intensitas citra (dalam skala keabuan) menjadi dua nilai (0 untuk nilai hitam dan 1 untuk nilai putih). Sebelum dilakukan *threshold* nilai intensitas di normalisasi terlebih dulu dengan membagi seluruh nilai dengan nilai terbesar yang dilakukan 2 kali. *Threshold* diperoleh menggunakan persamaan (2) [6].

$$g(i,j) = \begin{Bmatrix} 0, & jika\ f(i,j) < batas \\ 1, & jika\ f(i,j) > batas \end{Bmatrix} \quad \textbf{(2)}$$

Keterangan:
$batas$ = nilai ambang batas *threshold*
g = hasil *thresholding*

c. *Crop*

Crop adalah proses pemotongan citra dalam hal ini menghilangkan tepi citra pada nilai intensitas 1 atau putih. Tahap awal mencari nilai nol dari nilai intensitas citra masukkan (dalam skala *threshold*), dapat dilihat pada persamaan (3). Hasil nilai nol adalah sebuah *array* yang berisi indek (i,j). Kemudian, dicari batas atas (a), bawah (b), kiri (k), dan kanan (ka) dengan fungsi $min()$ dan $max()$ pada persamaan (4) dan persamaan (5). Tahap terakhir memotong citra skala keabuan dengan batas-batas (*boundary*) tersebut menggunakan fungsi pada persamaan (6).

$$n = \{(i,j), \quad jika\ f(i,j) = 0\} \quad \textbf{(3)}$$

$$a, k = min(\mathrm{n}) \quad \textbf{(4)}$$

$$b, ka = max(\mathrm{n}) \quad \textbf{(5)}$$

$$g = I[a{:}\,b, k{:}\,ka] \quad \textbf{(6)}$$

Keterangan:
n = *array* indek nilai intensitas nol
g = hasil *cropping*

d. *Resize*

Resize adalah proses merubah ukuran tinggi dan lebar dari suatu citra (skala keabuan). Diperoleh dengan langkah berikut.

1. Baca citra yang akan di-*resize*.
2. Tentukan ukuran citra $Tbaru$ dan $Lbaru$.
3. Buat array dengan ukuran baru.
4. Perulangan bersarang $for\ (i = 1\ to\ Lbaru)$ dan $for\ (j = 1\ to\ Tbaru)$.
5. Menghitung koresponden indek (i,j), menggunakan persamaan (7) dan persamaan (8).

$$ki = int(Llama\ *\ i\ /\ Lbaru) \quad \textbf{(7)}$$

$$kj = int(Tlama\ *\ j\ /\ Tbaru) \quad \textbf{(8)}$$

6. Mengisi *array* (y) dengan nilai intensitas (I) pada indek ki dan kj menggunakan persamaan (9).

$$y(i,j) = I(ki, kj) \quad \textbf{(9)}$$

Keterangan:
ki = koresponden indek i
kj = koresponden indek j
I = citra awal
y = citra baru

2.2. *Histogram of Oriented Gradients*

Ekstraksi ciri *Histogram of Oriented Gradients* (HOG) dilakukan dengan menghitung orientasi gradien di suatu daerah yang dilokalisasi pada citra. Penerapan HOG ini dapat dicapai dengan membagi citra ke dalam daerah-daerah kecil yang disebut sel, dan untuk setiap sel disusun histogram menurut orientasi arah, yang disebut bin [3]. Kombinasi histogram ini kemudian dinormalisasikan ke dalam sebuah vektor yang menyatakan deskriptor HOG yang mewakili sebuah citra atau objek.

Tahapan HOG dijelaskan pada [7], yaitu tahap awal HOG adalah *color normalization* atau merubah citra ke *grayscale*. Kemudian menghitung nilai gradien terhadap sumbu x (Gx) dan y (Gy) dari hasil operasi nilai intensitas citra (I) indek sesudahnya dikurang indek sebelumnya, dapat dilihat pada persamaan (10) dan persamaan (11).

$$Gx(x,y) = I(x+1,y) - I(x-1,y) \quad \textbf{(10)}$$

$$Gy(x,y) = I(x,y+1) - I(x,y-1) \quad \textbf{(11)}$$

Keterangan:
Gx = citra gradien sumbu x

Gy = citra gradien sumbu y
I = citra awal

Menentukan besaran gradien atau magnitudo (φ) dan orientasi (θ) dengan menggunakan persamaan (12) dan persamaan (13).

$$\varphi = \sqrt{Gx^2 + Gy^2} \quad \textbf{(12)}$$

$$\theta = arctan\,(Gy, Gx) \quad \textbf{(13)}$$

Keterangan:
Φ = besaran gradien *(gradient magnitude)*
Θ = orientasi gradien (*gradient orientation*)

Menentukan orientasi bin dengan cara membagi 180° (*unsigned*) dengan jumlah bin. Setiap bin histogram (misal 9 bin) memiliki rentang 20 (hasil 180 dibagi dengan 9). Setiap sel akan memiliki vektor dengan banyak bin. Bin indek pertama akan diisi dengan banyaknya nilai magnitudo yang terlebih dulu dilihat orientasi arah yang berkoresponden dengan nilai bin histogram. Cara yang sama dilakukan untuk mengisi indek bin histogram berikutnya.

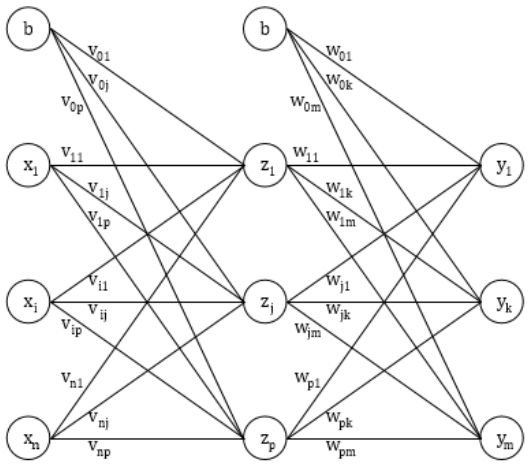
Tahap terakhir yaitu menormalisasi vektor dari masing-masing sel, kemudian menjadikan kedalam satu buah vektor yang disebut vektor HOG. Normalisasi dapat cari dengan menggunakan persamaan (14).

$$b = \frac{v}{\sqrt{||v|| + \epsilon}} \quad \textbf{(14)}$$

Keterangan:
b = vektor ternormalisasi
v = vektor belum ternormalisasi
ϵ = konstanta kecil (10^{-n}, $\mathrm{n} > 0$),
untuk menghindari pembagian dengan nol.

2.3. *Backpropagation*

Backpropagation adalah algoritma umum dari jaringan syaraf tiruan (JST) yang dapat digunakan untuk mengenali, mengklasifikasi, dan memprediksi suatu data. Proses pembelajaran algoritma *backpropagation* termasuk kategori metode pembelajaran terbimbing (*supervised learning*). Algoritma ini dapat menghasilkan keseimbangan antara kemampuaan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan memberikan respon yang benar terhadap pola masukan serupa namun tidak sama dengan pola yang dipakai selama pelatihan. *Backpropagation* terdiri atas dua langkah, yaitu perambatan maju (*feed forward*) dan perambatan mundur (*back forward*). Terdapat tiga lapisan pengolah, yaitu lapisan masukan (*input layers*), lapisan tersembunyi (*hidden layers*), dan lapisan keluaran (*output layers*) yang terhubung secara penuh [8]. Arsitektur jaringan dapat dilihat pada Gambar 1.



Gambar 1 Arsitektur *Backpropagation*

Keterangan:
x = lapisan masukkan
z = lapisan tersembunyi
y = lapisan keluaran
b = bias, selalu bernilai 1
v = bobot dari x ke z
w = bobot dari z ke y

Langkah-langkah perhitungan *backpropagation* [9].

Langkah 0
Inisialisasi bobot-bobot (tetapkan nilai acak kecil).

Langkah 1
Bila syarat berhenti adalah salah, kerjakan langkah 2 sampai 9.

Langkah 2
Untuk setiap pasangan pelatihan, kerjakan langkah 3 sampai 8.

Langkah 3
Tiap unit masukan (x_i, $\mathrm{i} = 1, \ldots, \mathrm{n}$) menerima isyarat masukan x_i dan diteruskan ke unit-unit tersembunyi.

Langkah 4
Setiap unit tersembunyi (z_j, $\mathrm{j} = 1, \ldots, \mathrm{p}$) jumlahkan bobot sinyal masukannya menggunakan persamaan (15).

$$z\_in_{jk} = v_{0j} + \sum_{i=1}^{n} x_i\, v_{ij} \quad \textbf{(15)}$$

v_{0j} = bias pada unit j, fungsi aktivasi untuk menghitung sinyal keluaran dengan menerapkan persamaan (16).

$$z_j = f\left(z\_in_j\right) \quad \textbf{(16)}$$

Kemudian kirimkan sinyal ini keseluruh unit pada lapisan diatasnya (unit-unit keluaran).

Langkah 5
Tiap unit keluaran $(y_k, \mathrm{k} = 1, \dots, \mathrm{m})$ jumlahkan bobot sinyal masukannya menggunakan persamaan (17).

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j\, w_{jk} \qquad \textbf{(17)}$$

w_{0k} = bias pada unit keluaran k, fungsi aktivasi untuk menghitung sinyal keluaran dengan menerapkan persamaan (18).

$$y_j = f(y\_in_k) \qquad \textbf{(18)}$$

Langkah 6
Tiap unit keluaran $(y_k, \mathrm{k} = 1, \dots, \mathrm{m})$ menerima pola target berkaitan dengan masukan pola pelatihan. Hitung galat atau kesalahan informasinya menggunakan persamaan (19).

$$\delta_k = (t_k - y_k) f'(y\_in_k) \qquad \textbf{(19)}$$

Hitung koreksi bobotnya (digunakan untuk memperbaharui w_{jk} nantinya) dengan menerapkan persamaan (20).

$$\Delta w_{jk} = \alpha \delta_k\, z_j \qquad \textbf{(20)}$$

Hitung koreksi biasnya (digunakan untuk memperbaharui w_{0k} nantinya dikirimkan δ_k ke unit-unit pada lapisan dibawahnya) dengan menerapkan persamaan (21).

$$\Delta w_{0k} = \alpha \delta_k \qquad \textbf{(21)}$$

Langkah 7
Setiap unit lapisan tersembunyi $(z_j, \mathrm{j} = 1, \dots, \mathrm{p})$ jumlahkan hasil perubahan masukannya (dari unit-unit dilapisan atasnya) menggunakan persamaan (22).

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k\, w_{jk} \qquad \textbf{(22)}$$

Hitung galat informasinya dengan menerapkan persamaan (23).

$$\delta_j = \delta\_in_j f'(x\_in_j) \qquad \textbf{(23)}$$

Hitung koreksi bobotnya (digunakan untuk memperbaharui v_{ij} nantinya) dengan menerapkan persamaan (24).

$$\Delta v_{ij} = \alpha \delta_j\, x_i \qquad \textbf{(24)}$$

Hitung koreksi biasnya (digunakan untuk memperbaharui v_{0j} nantinya) dengan menerapkan persamaan (25).

$$\Delta v_{0j} = \alpha \delta_j \qquad \textbf{(25)}$$

Langkah 8
Tiap unit keluaran $(y_k, \mathrm{k} = 1, \dots, \mathrm{m})$ perbaharui bobot-bobot dan biasnya $(\mathrm{j} = 0, 1, \dots, \mathrm{p})$ menggunakan persamaan (26).

$$w_{jk}\ (baru) = w_{jk}\ (lama) + \Delta w_{jk} \qquad \textbf{(26)}$$

Tiap unit tersembunyi $(z_j, \mathrm{j} = 1, \dots, \mathrm{p})$ memperbaharui bobot dan prasikapnya $(\mathrm{i} = 0, 1, \dots, \mathrm{n})$ menggunakan persamaan (27).

$$v_{ij}\ (baru) = v_{ij}\ (lama) + \Delta v_{ij} \qquad \textbf{(27)}$$

Langkah 9
Uji syarat berhenti.

2.4. *Batch Normalization*

Batch Normalization (BN) adalah teknik untuk meningkatkan kecepatan, kinerja, dan stabilitas jaringan saraf tiruan. BN memungkinkan penggunaan *learning rate* yang lebih tinggi dan tidak perlu khawatir tentang inisialisasi [10]. BN bekerja dengan menormalisasi input dengan terlebih dulu mencari nilai *mean* pada persamaan (28) dan *varian* pada persamaan (29). Formula normalisasi BN dapat dilihat pada persamaan (31).

$$\mu B = \frac{1}{m} \sum_{i=1}^{m} x_i \qquad \textbf{(28)}$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu B)^2 \qquad \textbf{(29)}$$

$$\hat{x}_i = \frac{x_i - \mu B}{\sqrt{\sigma_B^2 + \epsilon}} \qquad \textbf{(30)}$$

$$y_i = \gamma \hat{x}_i + \beta \qquad \textbf{(31)}$$

Keterangan:
μB = *mean* pelatihan
σ_B^2 = *varian* pelatihan
$\mathrm{x_i}$ = sinyal masukkan ke *hidden*
$\hat{\mathrm{x}}_\mathrm{i}$ = masukkan ke *hidden* ternormalisasi
$\mathrm{y_i}$ = hasil *batch normalization*
γ = gama, nilai awal 1
β = beta, nilai awal 0
ϵ = konstanta kecil $(10^{-\mathrm{n}},\ \mathrm{n} > 0)$,
untuk menghindari pembagian dengan nol.

Turunan *batch normalization* digunakan untuk mencari $\frac{\partial \ell}{\partial \mathrm{x_i}}$ pada proses *back forward* juga $\frac{\partial \ell}{\partial \gamma}$ dan $\frac{\partial \ell}{\partial \beta}$ untuk memperbaharui bobot gama (γ) dan beta (β).

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} . \gamma \qquad \textbf{(32)}$$

$$\frac{\partial \ell}{\partial \sigma_B^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} . (x_i - \mu B) . \frac{-1}{2} . (\sigma_B^2 + \epsilon)^{-3/2} \qquad \textbf{(33)}$$

$$\frac{\partial \ell}{\partial \mu B} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} . \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_B^2} . \frac{\sum_{i=1}^{m} -2(x_i - \mu B)}{m} \qquad \textbf{(34)}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} . \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_B^2} . \frac{2(x_i - \mu B)}{m} . \frac{\partial \ell}{\partial \mu B} . \frac{1}{m} \qquad \textbf{(35)}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i \quad \textbf{(36)}$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \quad \textbf{(37)}$$

Keterangan:
$\frac{\partial \ell}{\partial \hat{x}_i}$ = turunan masukkan ke *hidden* ternormalisasi
$\frac{\partial \ell}{\partial y_i}$ = turunan y
$\frac{\partial \ell}{\partial \sigma_B^2}$ = turunan *varian*
$\frac{\partial \ell}{\partial \mu B}$ = turunan *mean*
$\frac{\partial \ell}{\partial x_i}$ = turunan sinyal masukan ke *hidden*

Bobot *mean*, *varian*, gama, dan beta ketika proses pelatihan diperbaharui dengan menggunakan persamaan (38) untuk bobot *mean*, persamaan (39) untuk bobot *varian*, persamaan (40) untuk bobot gama, dan persamaan (41) untuk bobot beta.

$$\mu = \mu + \alpha\, \mu B \quad \textbf{(38)}$$

$$\sigma^2 = \sigma^2 + \alpha\, \sigma_B^2 \quad \textbf{(39)}$$

$$\gamma = \gamma + \alpha\, \frac{\partial \ell}{\partial \gamma} \quad \textbf{(40)}$$

$$\beta = \beta + \alpha\, \frac{\partial \ell}{\partial \beta} \quad \textbf{(41)}$$

Keterangan:
μ = *mean*
σ^2 = *varian*
γ = gama
β = beta
α = konstanta belajar *(0 < α < 1)*

2.5. *Activation Function* dan *Loss Function*

Activation Function atau fungsi aktivasi merupakan fungsi yang mentransformasikan input menjadi output tertentu, sedangkan *Loss Function* atau fungsi kesalahan adalah fungsi untuk menghitung besaran kesalahan output terhadap target. Terdapat dua fungsi aktivasi dan satu fungsi kesalahan yang digunakan sebagai berikut [11].

a. Rectified Linear Unit

Aktivasi *Rectified Linear Unit* (ReLU) yaitu merubah nilai menjadi nol (0) apabila nilai kurang dari nol. ReLU digunakan setelah proses *batch normalization*, formulanya seperti pada persamaan (42).

$$y_i = max(0, y_i) \quad \textbf{(42)}$$

b. Softmax

Softmax adalah fungsi eksponensial yang dinormalisasi ke dalam distribusi probabilitas dengan rentang (0 sampai dengan 1). *Softmax* digunakan pada *output layers*, formulanya dapat dilihat pada persamaan (43) dan turunannya pada persamaan (44).

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad \textbf{(43)}$$

$$\frac{\partial \ell}{\partial y_i} = t_i - y_i \quad \textbf{(44)}$$

Keterangan:
y_i = keluaran *softmax*
x_i = data masukkan
e^{x_i} = eksponensial x_i
e^{x_j} = jumlah eksponensial x
$\frac{\partial \ell}{\partial y_i}$ = turunan y_i
t_i = target

c. Cross Entropy

Cross entropy digunakan untuk mencari kesalahan prediksi terhadap target. Formula dapat dilihat pada persamaan (45).

$$CE = -\sum_{i=1}^{n} t_i\, log(y_i) \quad \textbf{(45)}$$

Keterangan:
CE = *cross entropy*
t_i = target
y_i = prediksi

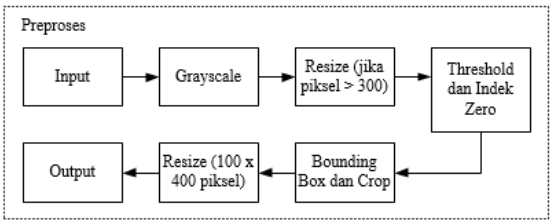
3. METODOLOGI PENELITIAN

3.1. Objek Penelitian

Objek yang digunakan dalam penilitian ini adalah citra tanda tangan. Aspek penelitian yang diteliti adalah mengenali tanda tangan menggunakan metode *backpropagation*. Tanda tangan yang dikenali terdiri dari dua kategori yaitu tanda tangan asli (positif) dan tanda tangan palsu (negatif).

3.2. Tahapan Proses Penelitian

Terdapat tiga tahap utama yaitu *preprocessing*, proses pelatihan dan proses pengujian. *preprocessing* dapat dlihat pada Gambar 2.

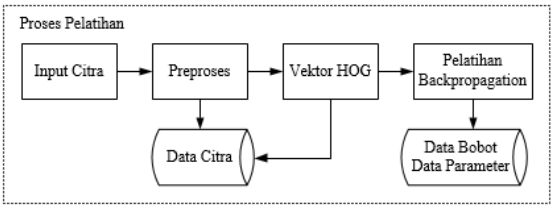


Gambar 2 Tahap *Preprocessing*

Keterangan:
a. Input *preprocessing* berupa citra berformat png dan jpg dengan ukuran bervariasi.
b. Mengubah input ke skala *grayscale* (keabuan).
c. Citra di-*resize* atau di ubah ukurannya menjadi tidak lebih dari 300 pada lebar maupun panjang citra.

d. Dilakukan *thresholding*, sehingga didapat citra dengan intensitas bernilai 0 (hitam) dan 1 (putih). Indek *zero* artinya menyimpan indek (i, j) dari citra dengan intensitas 0.
e. *Bounding box* digunakan untuk mencari batas terluar dari intensitas citra yang bernilai 0 atau nilai minimal dan maksimal dari *array* indek *zero*. Keluaran *bounding box* adalah indek batas kiri, batas atas, batas kanan, dan batas bawah.
f. *Crop* berfungsi untuk menghilangkan piksel putih di luar batas piksel hitam, artinya citra akan lebih fokus. *Cropping* dilakukan pada data *grayscale*.
g. *Resize* yang kedua berfungsi untuk menyelaraskan ukuran menjadi (100 x 400) piksel.
h. Output adalah citra yang telah melalui *preprocessing*.

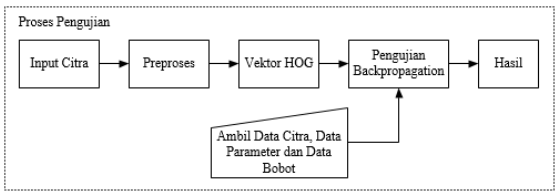
Tahap pelatihan dapat dilihat pada Gambar 3.



Gambar 3 Tahap Pelatihan

Keterangan:
a. Input *preprocessing* berupa citra berformat png dan jpg dengan ukuran bervariasi.
b. Preprocessing menghasilkan citra *grayscale* berukuran (100 x 400) piksel. Citra disimpan pada *database*.
c. Vektor HOG yang dihasilkan berupa matrik satu dimensi, juga disimpan pada database.
d. Pelatihan dengan paramater yang telah ditentukan sebelumnya akan menghasilkan bobot-bobot yang dapat digunakan untuk proses pengenalan.

Tahap pengujian dapat dilihat pada Gambar 4.



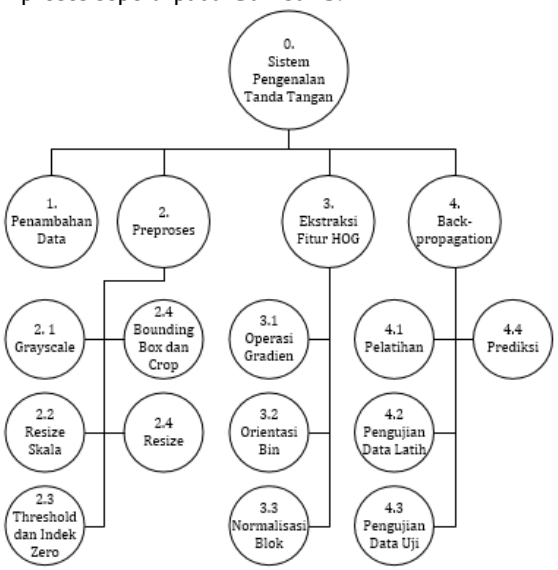
Gambar 4 Tahap Pengujian

Keterangan:
a. Input *preprocessing* berupa citra berformat png dan jpg dengan ukuran bervariasi.
b. *Preprocessing* menghasilkan citra *grayscale* berukuran (100 x 400) piksel.
c. Vektor HOG yang dihasilkan berupa matrik satu dimensi.
d. Data yang diuji dalam proses pengujian adalah data pelatihan dan data pengujian diambil dari *database*, juga data baru diluar *database*.

3.3. Perancangan Sistem

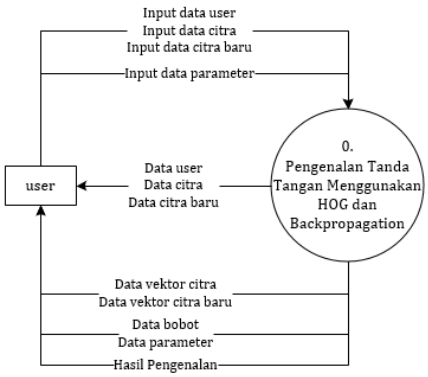
Perancangan sistem meliputi Diagram Jenjang, *Data Flow Diagram* (DFD), dan *Entity Relationship Diagram* (ERD).

a. Diagram Jenjang
Menjelaskan gambaran umum sistem yang dibangun. Sistem ini mempunyai proses penambahan data, *preprocessing* berfungsi menyelaraskan citra, ekstraksi fitur HOG untuk menghitung vektor citra dan proses *backpropagation*. Beberapa proses memiliki sub proses seperti pada Gambar 5.



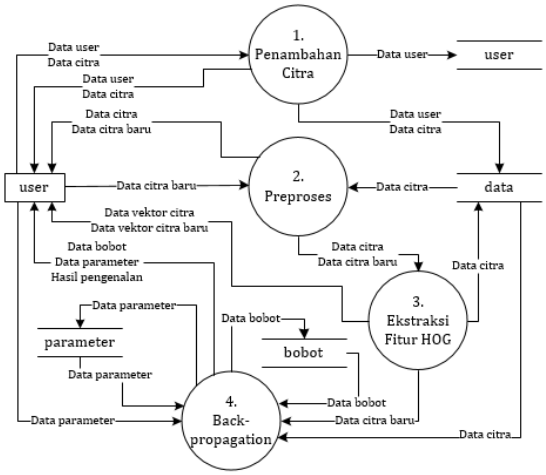
Gambar 5 Diagram Jenjang

b. *Data Flow Diagram*
1. Diagram Kontek
Diagram kontek (C*ontext diagram*) atau disebut juga DFD level 0, Diagram kontek ini menguraikan tindakan yang dapat dilakukan pengguna terhadap sistem dan apa saja yang didapat pengguna dari sistem tersebut. Diagram kontek dapat dilihat pada



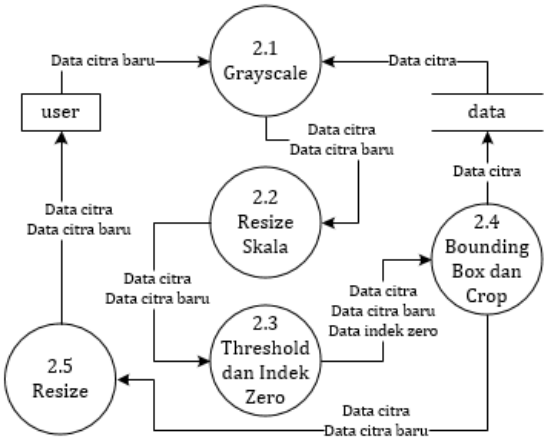
Gambar 6 Diagram Kontek

2. *Data Flow Diagram* Level 1
   Menguraikan secara lebih rinci pada diagram level 0 atau diagram konteks. Pada level ini terdiri atas beberapa proses yaitu pendataan, *preprocessing*, ekstraksi fitur *Histogram of Oriented Gradients*, dan *backpropagation*. Kemudian digunakan empat tabel penyimpanan yaitu user, data, parameter, dan bobot. Diagram level 1 dapat dilihat pada Gambar 7.



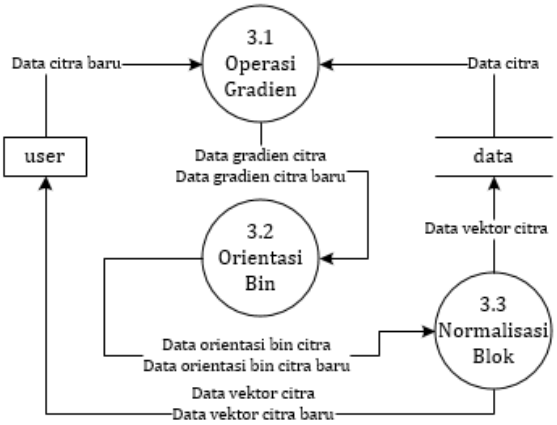
Gambar 7 *Data Flow Diagram* Level 1

3. *Data Flow Diagram* Level 2 Proses 2
   Pada level ini terdiri atas beberapa proses yaitu *grayscale*, *resize* skala, *threshold* dan indek *zero*, *bounding box* dan *crop*, dan *reisze*. Tabel penyimpanan yang terlibat dalam proses ini adalah tabel data. Diagram Level 2 Proses 2 dapat dilihat pada Gambar 8.



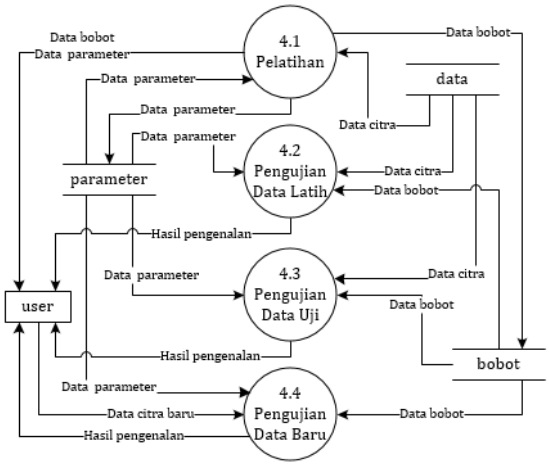
Gambar 8 *Data Flow Diagram* Level 2 Proses 2

4. *Data Flow Diagram* Level 2 Proses 3
   Pada level ini terdiri atas beberapa proses yaitu operasi gradien, orientasi bin, dan normalisasi blok. Tabel penyimpanan yang terlibat dalam proses ini adalah tabel data. Diagram Level 2 Proses 2 dapat dilihat pada Gambar 9.



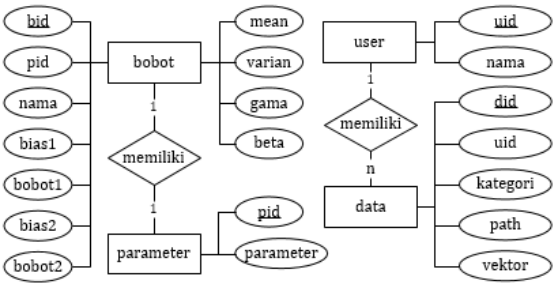
Gambar 9 *Data Flow Diagram* Level 2 Proses 3

5. *Data Flow Diagram Level* 2 Proses 4
   Pada level ini terdiri atas beberapa proses yaitu pelatihan, pengujian data latih, pengujian datauji, dan pengujian data baru. Tabel penyimpanan yang terlibat dalam proses ini adalah tabel data, parameter, dan bobot. Diagram Level 2 Proses 4 dapat dilihat pada Gambar 10.



Gambar 10 Data Flow Diagram Level 2 Proses 4

c. *Entity Relationship Diagram*
   Entity Relationship Diagram (ERD) menunjukkan hubungan antar entitas. Perancangan ERD dapat dilihat pada.

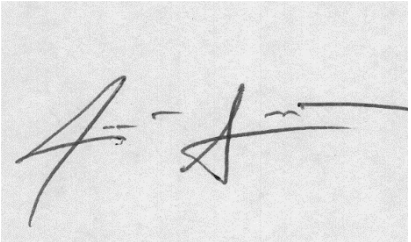


Gambar 11 *Entity Relational Diagram*

4. HASIL DAN PEMBAHASAN

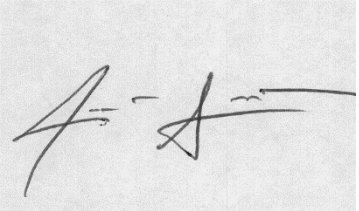
Data pengujian sistem yang pertama menggunakan data citra tanda tangan 55 responden dari situs yang telah disebutkan, dengan ketentuan diambil 18 untuk data pelatihan dan enam untuk data pelatihan di setiap kelas positif (asli) dan negatif (palsu). Pengujian kedua menggunakan data citra tanda tangan dari hasil foto lima responden, masing-masing 12 untuk data pelatihan dan empat untuk data pengujian. Proses pengujian menggunakan jaringan syaraf *backpropagation* dengan jumlah unit pada *input layer* sebanyak 900, *hidden layer* sebanyak 512 unit, dan *output layer* sebanyak n unit (110 untuk data pertama dan 10 untuk data kedua). Data yang akan dijadikan sebagai input layer adalah nilai vektor HOG citra dengan ukuran diselaraskan (100 x 400) piksel, dengan ketentuan jumlah bin 9, sel (20 x 20) piksel, blok (1 x 1) sel dan parameter sudut yang digunakan adalah 180 derajat.

Preprocessing dilakukan untuk menyelaraskan citra sebelum proses *Histogram of Oriented Gradients. Preprocessing* dimulai dengan memuat gambar dengan modul Matplotlib dan mengubah citra ke skala abu-abu dengan menerapkan langkah pada Persamaan (1), sehingga didapat hasil citra *grayscale* seperti pada Gambar 12.



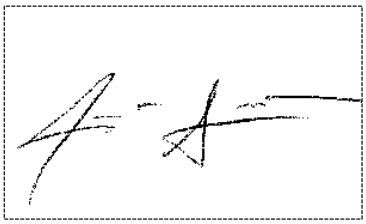
Gambar 12 *Grayscale* Ukuran (576 x 343) Piksel

Tahap berikutnya adalah mengecilkan ukuran citra apabila panjang atau lebar citra melebihi 300 piksel. Proses ini bertujuan untuk mempercepat proses pada tahap ekstraksi fitur. Hasil yang didapat ditunjukkan pada Gambar 13.



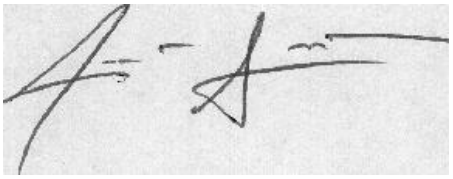
Gambar 13 Hasil *Resize* Rasio Ukuran (300 x 178) Piksel

Berikutnya adalah mengubah citra ke skala *threshold* dan menyimpan indek (i, j) ke dalam suatu *array* (indek zero). Hasil *thrreshold* yang didapat dapat dilihat pada Gambar 14.



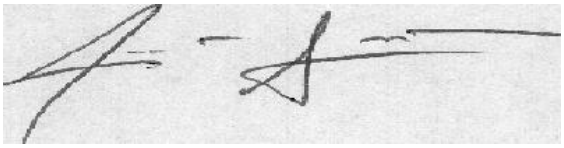
Gambar 14 Hasil *Threshold* Ukuran (300 x 178) Piksel

Hasil indek *zero* dari proses sebelumnya dilakukan proses *min* dan *max* untuk mendapat batas kiri, atas, kanan dan bawah. Batas-batas tersebut digunakan untuk memotong citra *grayscale* pada Gambar 14, sehingga didapat hasil *crop* pada Gambar 15.



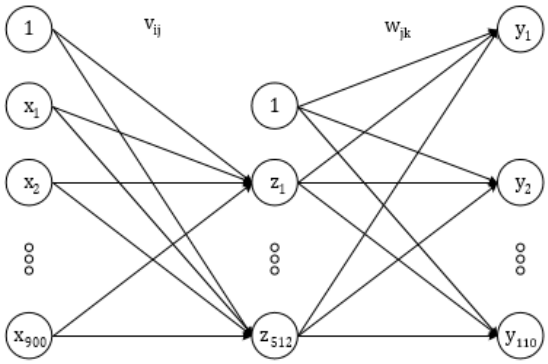
Gambar 15 Hasil *Crop* Ukuran **(**288 x 109) Piksel

Selanjutnya dilakukan penyelarasan ukuran, pada penelitian ini ukuran ditentukan (100 x 400) piksel. Hasil yang didapat ditunjukkan pada Gambar 16.



Gambar 16 Hasil *Resize* Ukuran (100 x 400) Piksel

Histogram of Oriented Gradients digunakan untuk mendeskripsikan sebuah citra melalui vektor biasa disebut vektor HOG. Untuk mencari vektor HOG dibutuhkan beberapa parameter yaitu jumlah bin, ukuran sel dan ukuran blok. Panjang vektor dengan inputan citra berukuran (100 x400) piksel dan menggunakan parameter yang telah disebutkan menghasilkan 900 vektor. Vektor tersebut digunakan sebagai inputan jaringan syaraf tiruan *backpropagation.* Berikut arsitektur jaringan untuk dataset pertama yang ditunjukkan pada Gambar 17.



Gambar 17 Arsitektur Jaringan Dataset Pertama

Berikut adalah tabel pengujian untuk dataset pertama dengan data uji sebanyak 660 buah citra yang ditunjukkan pada Tabel 1. Akurasi user maksudnya adalah akurasi sistem dalam memprediksi kelas user tersebut, walaupun salah dalam memprediksi kategorinya.

Tabel 1 Pengujian Dataset Pertama

No.	Hidden	Batch	Batas Error	Konstanta Belajar	Iterasi (waktu)	Akurasi	Akurasi User
1	512	128	0.001	0.005	870 (141.39 s)	90.76 %	92.12 %
2	1024	128	0.001	0.005	680 (220.56 s)	90.15 %	91.52 %
3	512	256	0.001	0.005	467 (131.65 s)	88.79 %	90.61%
4	1024	256	0.001	0.005	344 (187.42 s)	87.42 %	89.09 %
5	512	128	0.001	0.004	1101 (179.13 s)	89.09 %	90.91 %
6	1024	128	0.002	0.004	870 (228.25 s)	91.36%	92.58 %
7	512	256	0.002	0.004	292 (86.83 s)	89.55 %	91.06%
8	1024	256	0.002	0.003	338 (168.3 s)	88.64 %	90.0 %
9	512	128	0.002	0.003	804 (121.75 s)	89.09 %	90.91 %
10	1024	128	0.002	0.002	950 (293.26 s)	91.06 %	91.82 %

Berdasarkan Tabel 1, hasil pengujian terbaik ditunjukkan oleh tabel nomor 6, dengan akurasi sebesar 91.36 %. Memperbanyak jumlah *hidden* mempercepat iterasi, tetapi akurasi menurun. Pergantian konstanta belajar lebih kecil tidak menjamin akurasi meningkat. Jarak antara akurasi terendah dan tertinggi dalam pengujian tersebut, artinya jaringan telah bekerja secara maksimal. Untuk meningkatkan akurasi, perlu ditinjau ulang pada proses ekstraksi fiturnya. Pengujian dataset kedua terhadap data uji sebanyak 40 buah citra dapat dilihat pada Tabel 2.

Tabel 2 Pengujian Dataset Kedua

No.	Hidden	Batch	Batas Error	Konstanta Belajar	Iterasi (waktu)	Akurasi	Akurasi User
1	512	128	0,0001	0.005	180 (9.64 s)	87.5 %	100%
2	512	160	0,0001	0.005	199 (11.61 s)	85.0 %	100%
3	1024	128	0.0001	0.005	214 (20.5 s)	87.5 %	100%
4	1024	160	0.0001	0.005	79 (8.11 s)	85.0 %	100%

Berdasarkan Tabel 2, terlihat bahwa, sistem masih belum maksimal dalam memprediksi kategori tanda tangan. Hasil percobaan terbaik ditunjukkan oleh tabel nomor 1, dengan akurasi 87.5 %. Pada bagian akurasi user menghasilkan akurasi 100 %, ini menunjukkan jaringan mampu mengklasifikasikan data sesuai dengan kelas user tersebut. Pada Tabel 1, akurasi user belum maksimal, ini disebabkan adanya kemiripan vektor yang dihasilkan dari user satu dengan user lainnya. Solusi yang dapat dilakukkan adalah mengganti nilai parameter HOG yaitu ukuran sel yang lebih sedikit dan ukuran blok lebih besar. Cara tersebut akan menghasilkan vektor yang lebih banyak, sehingga fitur HOG lebih spesifik lagi, tetapi proses pelatihan lebih lama.

Berikut adalah perbandingan untuk dataset kedua tanpa dan dengan *batch normalization* yang diset 160. Vektor berjumlah 900, hasil dari proses HOG dari citra *grayscale* dengan ukuran 100x400 piksel. Parameter HOG ditentukan bin = 9, sel = 20x20, dan blok = 1x1. Parameter JST jumlah hidden 512 dan konstanta belajar 0.005. Perbandingan ditunjukkan pada Tabel 3.

Tabel 3 Perbandingan Tanpa dan Dengan *Batch Normalization*

No.	Batas Error	Iterasi (waktu)		Akurasi		Akurasi User	
		Tanpa BN	Dengan BN	Tanpa BN	Dengan BN	Tanpa BN	Dengan BN
1	0.005	383 (24.07 s)	21 (1.15 s)	87.5 %	97.5 %	85 %	100 %
2	0.004	432 (26.65 s)	28 (1.54 s)	80 %	92.5 %	87.5 %	100 %
3	0.003	485 (28.91 s)	41 (2.19 s)	85 %	95 %	85 %	100 %
4	0.002	663 (41.04 s)	46 (2.46 s)	77.5 %	92.5 %	87.5 %	100 %
5	0.001	1023 (61.55 s)	116 (6.07 s)	75 %	97.5 %	90 %	100 %

Berdasarkan Tabel 3 dengan menggunakan *batch normalization*, waktu pembelajaran lebih singkat dan meningkatkan akurasi.

5. PENUTUP

5.1. Kesimpulan

Kesimpulan yang dapat diambil pada penelitian yang telah dilakukan terhadap penerapan *histogram of oriented gradients* dan *backpropagation* adalah sebagai berikut.

a. Penelitian ini telah berhasil membuat sebuah sistem pengenalan pola tanda tangan melalui tahapan *preprocessing*, ekstraksi fitur menggunakan metode *Histogram of Oriented Gradients* (HOG) dan proses klasifikasi menggunakan metode Jaringan Syaraf Tiruan *Backpropagation*.
b. Sistem pengenalan ini memiliki akurasi tertinggi 91.36 % untuk dataset pertama yang didapat dari http://www.cedar.buffalo.edu/NIJ/data/signatures.rar. Dengan ketentuan pembagian data untuk data uji 25 persen pada masing-masing kelas. Kemudian pada dataset kedua yang diperoleh dari hasil foto, memiliki akurasi tertinggi 87.5 % juga dengan 25 persen data uji.
c. Penerapan *batch normalization* terbukti sangat efektif dalam proses pelatihan. Terlihat dari waktu pelatihan yang lebih singkat dan akurasi meningkat.

5.2. Saran

Berdasarkan kesimpulan yang telah diperoleh, maka terdapat beberapa saran agar kedepannya dapat diperoleh hasil yang lebih baik. Saran untuk penelitian selanjutnya adalah sebagai berikut.

a. Pengenalan pola tanda tangan selanjutnya dapat dilakukan secara online dan *real-time object detection* dengan *mobile application*, sehingga lebih mudah dalam pengunaannya.
b. Kendala *histogram of oriented gradient* adalah semakin citra terdeskripsikan dengan baik, semakin besar pula vektor yang dihasilkan. Selanjutnya dapat dikombinasikan dengan teknik reduksi dimensi.

6. DAFTAR PUSTAKA

[1] Octariadi, B. C., *Pengenalan Pola Tanda Tangan Menggunakan Metode Jaringan Syaraf Tiruan Backpropagation*, *J. Teknoinfo*, vol. 14, no. 1, hal. 15–21, (2020).

[2] Marthatiyanda, O., *Pengenalan Tanda Tangan Menggunakan Histogram Of Oriented Gradients Dan Smooth Support Vector Machine*. Universitas Komputer Indonesia, (2019).

[3] Harfiya, L. N., Widodo, A. W. dan Wihandika, R. C., *Verifikasi Citra Tanda Tangan Berdasarkan Ciri Pyramid Histogram Of Oriented Gradient (PHOG) Menggunakan Metode Klasifikasi K-Nearest Neighbor*, *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, hal. 964X, (2017).

[4] Martha, A. V., Hanafi, M. dan Burhanuddin, A., *Implementasi Jaringan Syaraf Tiruan (JST) Untuk Mengenali Pola Tanda Tangan Dengan Metode Backpropagation*, hal. 51–57, (2019).

[5] Kadir, A. dan Susanto, A., *Teori dan Aplikasi Pengolahan Citra*. Yogyakarta: Andi Publishing, 2013.

[6] Putra, D., *Pengolahan Citra Digital*. Yogyakarta: Andi, 2010.

[7] Dalal, N. dan Triggs, B., *Histograms Of Oriented Gradients For Human Detection*, in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, 2005, vol. 1, hal. 886–893.

[8] Arief, H., *Jaringan Saraf Tiruan*, *Yogyakarta Andi*, (2006).

[9] Kusumadewi, S., *Artificial Intelligence (Teknik Dan Aplikasinya)*, *Graha Ilmu. Yogyakarta*, (2003).

[10] Ioffe, S. dan Szegedy, C., *Batch Normalization: Accelerating Deep Network Training By Reducing Internal Covariate Shift*, *arXiv Prepr. arXiv1502.03167*, (2015).

[11] Goodfellow, I., Bengio, Y. dan Courville, A., *Deep Learning*. MIT Press, 2016.