PENERAPAN METODE *CONVOLUTIONAL NEURAL NETWORK* PADA PENGENALAN POLA CITRA SANDI RUMPUT

Alwan Hibatullah[1], Irfan Maliki[2]

[1,2]Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
Email: alwanhibatullah@gmail.com[1], irfan.maliki@email.unikom.ac.id[2]

ABSTRAK

Sandi rumput termasuk sandi yang unik karena memiliki pola yang menyerupai rumput. Karena bentuk polanya yang unik, dilakukan penelitian mengenai pengenalan pola citra sandi rumput. Pada penelitian sebelumnya yang membahas kajian mengenai pengenalan sandi rumput didapat hasil akurasi yang cukup bagus namun masih banyak pola yang belum dikenali. Penelitian ini menggunakan metode *Convolutional Neural Network* (CNN) untuk mengklasifikasi jenis pola sandi rumput. Tujuan dari penelitian ini adalah mengetahui tingkat akurasi dari CNN dalam mengenali pola citra sandi rumput. Penelitian ini memiliki beberapa tahapan yaitu *preprocessing* yang meliputi proses *grayscale*, *thresholding*, segmentasi, dan *resize*. Kemudian dilanjutkan dengan proses pelatihan pada CNN yang meliputi *Convolution Layer, Activation Layer, Pooling Layer, Fully Connected Layer, Hidden Layer*, dan *Output Layer* serta proses *Backpropagation*. Proses pelatihan menggunakan data sebanyak 2600 data. Adapun proses pengujian CNN meliputi *Convolution Layer, Activation Layer, Pooling Layer, Fully Connected Layer, Hidden Layer*, dan *Output Layer*. Berdasarkan hasil pengujian dengan menggunakan data sebanyak 260 data, didapat persentasi akurasi terbaik sebesar 96.92%. Hasil akurasi ini dipengaruhi oleh nilai *learning rate* pada proses pelatihan. Selain itu, tingkat kerapihan pola, jumlah *dataset* dan jumlah *layer* dalam arsitektur CNN juga berpengaruh terhadap tingkat akurasi.

Kata kunci: Sandi Rumput, Pengolahan Citra, *Convolutional Neural Network*, *Feedforward*, *Backpropagation*

1. PENDAHULUAN

Sandi rumput termasuk sandi yang unik karena memiliki bentuk pola yang menyerupai rumput. Karena bentuk polanya yang unik inilah, dilakukan penelitian mengenai pengenalan pola citra sandi rumput.

Tujuan dari penelitian ini yaitu untuk mengetahui seberapa akurat metode yang digunakan pada penelitian ini untuk mengenali pola sandi rumput. Pada penelitian sebelumnya yang membahas kajian mengenai pengenalan sandi rumput didapat akurasi sebesar 76,28% untuk karakter tunggal dan 78,37% untuk kata bersambung [1]. Hasil tersebut cukup bagus, namun masih ada pola sandi rumput yang belum bisa dikenali.

Pengenalan pola citra merupakan salah satu kemampuan yang dimiliki oleh komputer. Sebagian besar pengenalan pola dilakukan pada karakter huruf latin dengan teknik *Optical Character Recognition* (OCR) yang dapat menerjemahkan karakter pada citra digital menjadi format teks [2]. Pada penelitian sebelumnya, OCR sudah digunakan untuk berbagai macam objek, seperti karakter aksara [3], dan tulisan latin bersambung [4].

Convolutional Neural Network (CNN) digunakan sebagai metode untuk mengklasifikasi jenis pola sandi rumput. CNN merupakan salah satu kelas *deep feed-forward artificial neural network* yang banyak diaplikasikan pada analisis citra [5]. Pada penelitian sebelumnya, CNN telah diaplikasikan pada berbagai macam objek dan menunjukkan efektifitasnya, seperti pada penelitian mengenai pengenalan aksara jawa [6] [7], aksara sunda [8], pengenalan wajah secara real-time [9], dan pengenalan simbol semapur [10]. Pada kasus pengenalan tulisan tangan pada aksara jawa, didapat tingkat akurasi terbaik sebesar 89% [6].

Maka, penelitian ini akan dibuat sebuah prototype dari penerapan *Convolutional Neural Network* pada pengenalan citra sandi rumput dengan tulisan tangan. Selanjutnya, akan dilakukan perhitungan akurasi dengan mengukur banyaknya pola sandi yang dapat dikenali.

2. ISI PENELITIAN

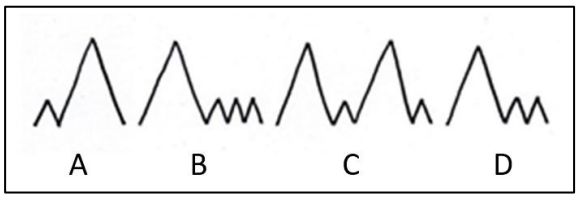
Pada bagian ini membahas mengenai kajian teori singkat, metode penelitian, pembahasan, serta hasil pengujian yag telah dilakukan.

2.1 Tinjauan Pustaka

Bagian ini akan membahas mengenai Sandi Rumput dan Pengolahan Citra.

2.1.1 Sandi Rumput

Sandi rumput pramuka merupakan sistem representasi huruf, angka, dan tanda baca yang dikembangkan oleh penggiat kepramukaan Indonesia [1]. Sandi rumput mempunyai bentuk yang khas yaitu menyerupai rumput dan cara pengerjaan yang baku. Pada dasarnya, sandi rumput bukanlah sandi asli, melainkan turunan dari sandi morse, sehingga representasi masing-masing huruf, angka, dan tanda baca yang ada di sandi rumput sama dengan sandi morse. Perbedaan diantara keduanya terletak pada cara penulisannya, dimana titik dan garis yang ada di sandi morse diganti dengan rumput kecil dan rumput besar



Gambar 1 Sandi Rumput

2.1.2 Pengolahan Citra

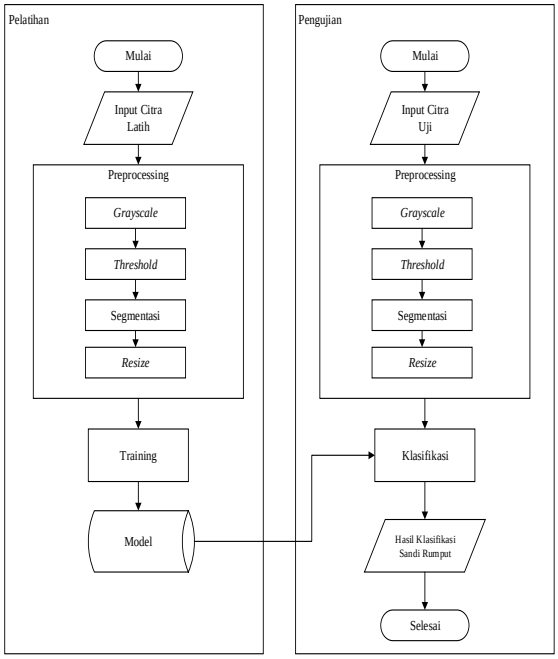
Menurut Arymurthy dalam [11], pengolahan citra merupakan bidang studi yang mempelajari proses pengolahan gambar dimana baik masukan maupun keluarannya berbentuk berkas citra digital. Foto adalah contoh gambar yang dapat diolah secara mudah. Setiap foto dalam bentuk citra digital dapat diolah melalui perangkat lunak tertentu.

2.2 Analisis Masalah

Masalah pada penelitian ini yaitu mengukur tingkat akurasi pengenalan citra sandi rumput dengan menggunakan metode CNN. Pada penelitian sebelumnya pengenalan sandi rumput sudah dilakukan [1], namun masih banyak karakter sandi rumput yang belum bisa dikenali. Hal ini disebabkan karena metode klasifikasi yang belum maksimal dalam mengenali sandi rumput, selain itu terdapat faktor lain diantaranya rendahnya resolusi citra serta jumlah citra uji yang terbatas. Pada penelitian ini metode CNN akan digunakan sebagai metode klasifikasi pada pengenalan pola sandi rumput

2.3 Analisis Sistem

Proses yang dilakukan dalam sistem ini dibagi ke dalam bagian-bagian dimana setiap proses memiliki peranan masing-masing dalam mengenali tulisan tangan.



Gambar 2 Gambaran Umum Sistem

Proses yang dilakukan pertama kali yaitu memasukkan citra tulisan tangan sandi rumput ke dalam sistem. Selanjutnya masuk ke dalam tahap *preprocessing* yang meliputi *grayscale*, *thresholding*, segmentasi, dan *resize*. Dari tahap preprocessing akan didapat array nilai desimal dari setiap karakter, array tersebut kemudian diolah kembali pada tahap klasifikasi dengan CNN baik pelatihan maupun pengujian. Hasil dari pengenalan berupa teks digital.

2.4 Analisis *Preprocessing*

Tahap preprocessing terdiri dari *grayscale*, *thresholding*, segmentasi, dan *resize*.

a. Grayscale

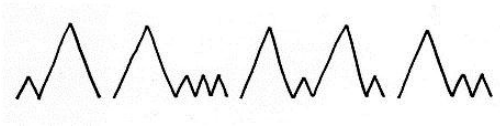
Citra keabuan atau *grayscale* merupakan citra yang hanya memiliki satu buah *channel* sehingga yang ditampilkan hanyalah nilai intensitas atau derajat keabuan [11]. Dalam hal ini, intensitas berkisar antara 0 sampai 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih [12].

Proses ini akan mengecilkan range warna citra menjadi 0 sampai 255. Proses ini akan memudahkan ketika akan melakukan *thresholding* citra menjadi citra biner. Adapun perhitungan untuk mencari nilai *grayscale* menggunakan persamaan sebagai berikut.

$$y = (0{,}2989 * R) + (0{,}5870 * G) + (0{,}1141 * B) \quad (1)$$

Keterangan

y : Nilai piksel *grayscale*
R : Nilai piksel Merah (*Red*)
G : Nilai piksel Hijau (*Green*)
B : Nilai piksel Biru (*Blue*)



Gambar 3 Contoh Hasil Citra *Grayscale*

b. *Thresholding*

Thresholding merupakan suatu proses dimana hasilnya berupa citra biner dari citra *grayscale* atau citra berwarna dengan mengatur nilai piksel ke nilai 0 atau 1 tergantung dari nilai ambang batasnya apakah nilai piksel tersebut berada dibawah atau diatas ambang batas [13]. *Thresholding* terbagi menjadi dua yaitu *global tresholding* dan *local thresholding*. *Global thresholding* merupakan *thresholding* yang apabila nilai ambang t bergantung hanya pada satu nilai aras keabuan f(x,y). Sedangkan *local thresholding* adalah *thresholding* yang dimana nilai ambang t bergantung pada f(x,y) dan g(y,x) dengan g(y,x) menyatakan properti citra local pada titik (y,x) [12]

Metode *thresholding* yang digunakan pada penelitian ini yaitu metode *Sauvola Threshold* [14]. Berikut ini adalah langkah-langkah perhitungannya.

1. Hitung nilai rata-rata piksel dengan persamaan sebagai berikut

$$m(x,y) = \frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} img(i,j)}{i*j} \quad (2)$$

Keterangan:

m : Rata-Rata piksel citra
img : Nilai keabuan pada citra
i,j : Nilai piksel tetangga

2. Hitung nilai standar deviasi dengan persamaan sebagai berikut.

$$s = \sqrt{\frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} (img(i,j)-m(x,y))^2}{(i*j)-1}} \quad (3)$$

Keterangan:

s : Standar deviasi dari sejumlah piksel citra
m : Rata-Rata piksel citra
img : Nilai keabuan pada citra
i,j : Nilai piksel tetangga
x.y : Lebar dan tinggi citra

3. Lalu hitung nilai *threshold* dengan persamaan sebagai berikut.

$$T(x,y) = m(x,y) * \left[1 + k * \left(\frac{s(x,y)}{R} - 1\right)\right] \quad (4)$$

Keterangan:

T : Nilai *Threshold*
m : Rata-Rata piksel citra
k : Nilai antara 0.2-0.5
s : Standar deviasi dari sejumlah piksel citra
R : Nilai maksimum standar deviasi
i,j : Nilai piksel tetangga
x.y : Lebar dan tinggi citra

4. Selanjutnya masukkan nilai threshold yang didapat ke dalam persamaan berikut.

$$f(x,y) = \begin{cases} 0, & img(x,y) < T(x,y) \\ 255, & img(x,y) \geq T(x,y) \end{cases} \quad (5)$$

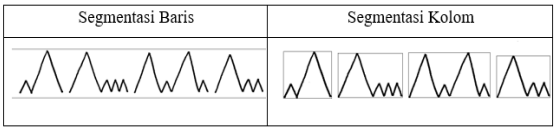
Keterangan:

f : Fungsi yang menghasilkan nilai antara 0 atau 255
img : Nilai grayscale pada citra

Adapun nilai konstanta yang digunakan dalam proses *thresholding* ini yaitu:
Nilai piksel tetangga = 21
Nilai R = 128
Nilai k = 0.3

c. Segmentasi

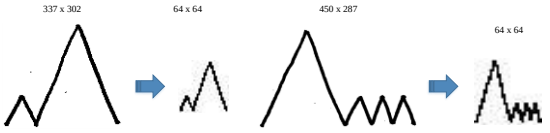
Pada proses ini, data yang digunakan yaitu citra biner hasil dari proses thresholding. Kemudian akan dilakukan pemotongan untuk mendapatkan citra sandi rumput. Pemotongan dilakukan untuk setiap baris (vertikal) terlebih dahulu, selanjutnya dilakukan pemotongan setiap kolom (horizontal) pada citra hasil dari pemotongan baris.



Gambar 4 Contoh Hasil Segmentasi

d. *Resize*

Resize merupakan proses mengubah ukuran suatu citra menjadi lebih besar ataupun lebih kecil dari ukuran citra sebelumnya dengan ukuran yang telah ditentukan sebelumnya. Dalam penelitian ini, citra sandi rumput yang sudah tersegmen pada tahap segmentasi di-*resize* menjadi berukuran 64 x 64 piksel

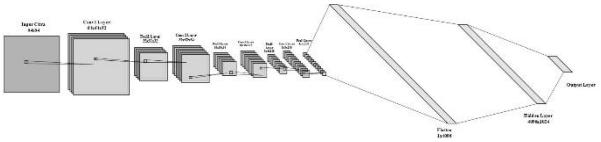


Gambar 5 Contoh Proses *Resize*

2.5 Analisis *Convolutional Neural Network* (CNN)

CNN termasuk dalam jenis *Deep Learning* yang banyak diaplikasikan pada data citra. CNN terinspirasi oleh proses-proses biologi dimana pola konektivitas antar neuron menyerupai organisasi visual cortex pada binatang [5]. Gagasan awal dari pengembangan *Deep Learning* yaitu mempelajari apa yang sebenarnya terjadi pada lapisan tersembunyi (*hidden layer*) di dalam sebuah jaringan syaraf tiruan[15]. Di dalam CNN terdapat empat lapisan utama antara lain lapisan konvolusi

(*Convolution Layer*), lapisan aktivasi (*Activation Layer*), lapisan penggabungan (*Pooling Layer*), dan lapisan terhubung penuh (*Fully-Connected Layer*).



Gambar 6 Gambaran Arsitektur CNN

Adapun tahapan pelatihan (*training*) pada CNN terdiri dari tahap inisialisasi, *feedforward*, *backpropagation*, dan *update* bobot. Masukan data berupa citra biner dan keluaran berupa klasifikasi sandi rumput.

2.5.1 Inisialisasi

Tahap inisialisasi terdiri dari inisialisasi parameter dan inisialisasi bobot. Inisialisasi parameter meliputi penentuan jumlah maksimum *epoch*, nilai *learning rate*, dan minimum *error*. Adapun inisialisasi bobot yaitu menentukan nilai-nilai awal bobot dan bias pada filter *convolution layer*, *hidden layer*, dan *fully connected layer*. Semua nilai bobot diisi dengan menggunakan metode *He Initialization*[16].

2.5.2 Feedforward

Pada tahap *feedforward* akan dilakukan proses CNN dari awal masukan melewati *convolution*, *activation*, *pooling*, dan *fully-connected layer* hingga menghasilkan sebuah vektor *one-hot* klasifikasi kelas sandi.

a. Convolution Layer

Pada lapisan ini akan dilakukan operasi konvolusi antara matriks citra masukan dengan matriks-matriks filter. Filter-filter ini akan digeser ke seluruh permukaan citra sehingga akan menghasilkan keluaran matriks *feature map. Feature Map* dapat yang akan dihasilkan didapat dari rumus berikut.

$$n_{out} = \left(\frac{n_{in}-k+2p}{s}\right) + 1 \qquad (6)$$

dimana:

n_{out} : Ukuran *feature map*
n_{in} : Ukuran matriks masukan
k : Ukuran matriks filter
p : ukuran padding
s : *stride*

Adapun rumus operasi konvolusi adalah sebagai berikut.

$$FM[i]_{j,k} = (\sum_m \sum_n N_{[j-m,\ k-n]} F_{[m,n]}) + bF \qquad (7)$$

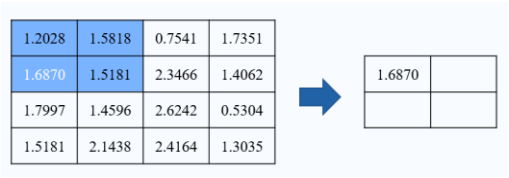
dimana :

FM[i] : Matriks Feature Map ke-i
N : Matriks citra masukan
F : Matriks filter konvolusi
bF : Nilai bias pada filter
j,k : Posisi piksel pada matriks citra masukan
m,n : Posisi piksel pada matriks filter konvolusi

Setelah dilakukan proses konvolusi, selanjutnya lakukan aktivasi fungsi aktivasi menggunakan *fungsi Rectified Linear Unit (ReLU)*. Setiap piksel pada feature map akan dimasukkan ke dalam fungsi ReLU, dimana piksel yang memiliki nilai kurang dari 0 akan diubah nilainya menjadi 0, dengan rumus f(x) = max(0,x).

b. Pooling Layer

Pooling layer digunakan untuk mengurangi ukuran dari *feature map*. Jenis *pooling* yang digunakan yaitu *max pooling*, yaitu memilih nilai maksimun pada suatu jendela tertentu. Prosesnya mirip seperti *convolution layer* yaitu menggeser jendela ke seluruh permukaan citra, namun disini jendela digunakan sebagai acuan untuk memilih nilai maksimum pada suatu area tertentu. Proses ini akan menghasilkan *output* berupa matriks *feature map* yang berisi nilai-nilai maksimum yang terpilih.



Gambar 7 Contoh *Max Pooling*

c. Fully-Connected Layer

Fully-connected layer terdiri dari *input layer*, *hidden layer* dan *output layer*.

d. Input Layer

Input Layer merupakan penggabungan keseluruhan matriks *feature map* yang didapat dari proses *pooling layer* lalu semua piksel tersebut direntangkan menjadi sebuah vektor sepanjang jumlah piksel dari matriks yang didapat pada *pooling layer*. Kemudian seluruh nilai pada *input layer* digunakan untuk perhitungan pada *hidden layer.*

e. Hidden Layer

Perhitungan pada layer ini yaitu mengkalikan nilai-nilai yang ada di *input layer* dengan bobot yang sudah diinisialisasi sebelumnya lalu ditambahkan dengan nilai bias. Adapun rumus perhitungannya adalah sebagai berikut.

$$z\_in_i = \sum_{j=1}^{n} X_j * V_{j,i} + V_{0,i} \qquad (8)$$

dimana:

z_in_i : masukkan untuk node *hidden layer* ke-i dengan jumlah node n
X_j : node X ke-j

$V_{j,1}$: bobot V untuk X_j dan node Z_i
V_0 : bias V untuk z_in_i

Setelah dilakukan perhitungan, selanjutnya masukkan fungsi aktivasi ReLU untuk semua hasil perhitungan, maka akan diperoleh nilai keluaran Z. Adapun hasil dari perhitungan ini akan digunakan untuk proses perhitungan pada *output layer*.

f. Output Layer

Perhitungan pada layer ini yaitu mengkalikan nilai-nilai hasil perhitungan pada hidden layer dengan bobot yang sudah diinisialisasi sebelumnya lalu ditambahkan dengan nilai bias. Adapun rumus perhitungannya adalah sebagai berikut.

$$y\_in_i = \sum_{j=1}^{m} Z_j * W_{j,i} + W_{o,i} \quad (9)$$

dimana:
y_in_i : Masukkan untuk node *hidden layer* Z ke-i dengan jumlah node m
Z_j : node Z ke-j
$W_{j,i}$: bobot W untuk Z_j dan node Y_i
W_0 : bias W untuk y_in_i

Setelah dilakukan perhitungan, selanjutnya masukkan fungsi aktivasi Softmax untuk semua hasil perhitungan sehingga akan diperoleh nilai keluaran Y. Adapun rumus perhitungan fungsi softmax adalah sebagai berikut.

$$Y_i = \frac{e^{y\_in_i}}{\sum_{i=1}^{m} e^{M}} \quad (10)$$

dimana:
Y_i : keluaran untuk *output layer* ke-i
y_in_i : masukan untuk *node layer* ke-i
M : semua masukan untuk *output layer* sejumlah *m* buah

2.5.3 Backpropagation

Tahap ini bermaksud untuk menyesuaikan kembali tiap bobot dan bias berdasarkan *error* yang didapat pada saat proses *feedforward.*

a. Menghitung Gradien W

Pada tahap ini akan menghitung gradien kesalahan terhadap bobot W_{ji} dengan menggunakan aturan rantai (*chain rule*). Adapun rumusnya sebagai berikut [17].

$$\frac{\partial L}{\partial W_{ji}} = \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y\_in_i}\frac{\partial y\_in_i}{\partial W_{ji}} \quad (11)$$

dimana,

$$\frac{\partial L}{\partial Y_i} = \frac{-t_i}{Y_i} + \frac{1-t_i}{1-Y_i} = \frac{Y_i - t_i}{Y_i(1-Y_i)} \quad (12)$$

$$\frac{\partial Y_i}{\partial y\_in_i} = Y_i(1-Y_i) \quad (13)$$

$$\frac{\partial y\_in_i}{\partial W_{ji}} = Z_j \quad (14)$$

maka,

$$\frac{\partial L}{\partial W_{ji}} = (Y_i - t_i)\, Z_j \quad (15)$$

Keterangan
Y_i : keluaran untuk *output layer* ke-i
t_i : nilai vektor target (vektor *one-hot*)
Z_j : masukan untuk *node layer* ke-i

b. Menghitung Gradien V

Menghitung gradien kesalahan terhadap bobot V_{kj} dengan menggunakan aturan rantai sebagai berikut [17].

$$\frac{\partial L}{\partial V_{kj}} = \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y\_in_i}\frac{\partial y\_in_i}{\partial Z_j}\frac{\partial Z_j}{\partial z\_in_j}\frac{\partial z\_in_i}{\partial V_{kj}} \quad (16)$$

dimana,

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1-Y_i)} \quad (17)$$

$$\frac{\partial Y_i}{\partial y\_in_i} = Y_i(1-Y_i) \quad (18)$$

$$\frac{\partial y\_in_i}{\partial Z_j} = W_{ji} \quad (19)$$

$$\frac{\partial Z_j}{\partial z\_in_j} = Z_j(1-Z_j) \quad (20)$$

$$\frac{\partial z\_in_i}{\partial V_{kj}} = X_k \quad (21)$$

maka,

$$\frac{\partial L}{\partial V_{kj}} = \sum_{i}^{m} (Y_i - t_i)(W_{ji})(Z_j(1-Z_j))(X_k) \quad (22)$$

Keterangan:
Y_i : Keluaran untuk *output layer* ke-i
t_i : nilai vektor target (vektor *one-hot*)
Z_j : nilai *hidden layer* ke-j
W_{ji} : nilai bobot W_j ke-i
X_k : nilai input *layer* ke-k

c. Menghitung Gradien Filter F

Tahap ini akan menghitung gradien terhadap filter. Sebelumnya terlebih dahulu gradien pada layer sebelumnya sesuai dengan *chain rule.*

Pertama, hitung gradien terhadap bobot X_k (*Input Layer*) dengan menggunakan rumus sebagai berikut.

$$\frac{\partial L}{\partial X_k} = \sum_{j}^{n}\sum_{i}^{m} (Y_i - t_i)(W_{ji})(Z_j(1-Z_j))(V_{kj}) \quad (23)$$

Keterangan:
Y_i : Keluaran untuk *output layer* ke-i
t_i : nilai vektor target (vektor *one-hot*)
Z_j : nilai *hidden layer* ke-j

W_{ji} : nilai bobot W_j ke-i
V_{kj} : nilai bobot V_k ke-j

Setelah didapat nilai gradien pada Input Layer, lalu kembalikan nilai-nilai yang didapat kedalam bentuk *feature map*, kemudian hitung nilai gradien terhadap *Pooling Layer* dengan mengacu pada nilai maksimum pada feature map sesuai hasil proses *max-pooling* pada tahap *feedforward*. Nilai gradien dari *pooling layer* untuk piksel-piksel dengan nilai maksimum adalah nilai gradien dari *Input Layer*, sedangkan sisanya adalah 0. Setelah didapat hasilnya, maka lakukan operasi konvolusi antara gradien *feature map* dengan matriks input yang sama seperti tahap *feedforward*. Maka akan didapat gradien untuk filter.

2.5.4 Update Parameter

Metode yang digunakan untuk meng-update parameter yang terlibat yaitu metode *Adam Optimizer* [18]. Adapun langkah-langkahnya sebagai berikut.

Pertama, hitung nilai m_t menggunakan persamaan berikut.

$$m_t = \beta_1 * m_{t-1} + (1-\beta_1) * g_t \quad (24)$$

dimana,
m_t : Nilai momentum pertama
β_1 : Konstanta Laju Peluruhan (0.9)
m_{t-1} : Inisialisasi vektor momentum pertama
g_t : Nilai gradien bobot

Kedua, hitung nilai v_t dengan persamaan berikut.

$$v_t = \beta_2 * v_{t-1} + (1-\beta_2) * g_t^2 \quad (25)$$

dimana,
v_t : Nilai momentum kedua
β_2 : Konstanta Laju Peluruhan (0.999)
v_{t-1} : Inisialisasi vektor momentum kedua
g_t : Nilai gradien bobot

Ketiga, hitung nilai $\breve{m}_t$ menggunakan persamaan berikut.

$$\breve{m}_t = \frac{m_t}{\sqrt{1-\beta_1^t}} \quad (26)$$

dimana,
$\breve{m}_t$: Nilai momentum pertama dengan pembaruan bias
β_1 : Konstanta Laju Peluruhan (0.9)
m_t : Nilai momentum pertama
t : *Epoch*

Keempat, hitung nilai $\breve{v}_t$ menggunakan persamaan berikut.

$$\breve{v}_t = \frac{v_t}{\sqrt{1-\beta_2^t}} \quad (27)$$

dimana,
$\breve{v}_t$: Nilai momentum kedua dengan pembaruan bias
β_1 : Konstanta Laju Peluruhan (0.999)
v_t : Nilai momentum kedua
t : *Epoch*

Langkah terakhir, *update* bobot dengan menggunakan persamaan berikut.

$$\theta_t = \theta_{t-1} - \alpha\left(\frac{\breve{m}_t}{\sqrt{\breve{v}_t + \varepsilon}}\right) \quad (28)$$

dimana,
θ_t : Nilai bobot yang telah diperbaharui
θ_{t-1} : Nilai bobot lama
$\breve{m}_t$: Nilai momentum pertama dengan pembaruan bias
$\breve{v}_t$: Nilai momentum kedua dengan pembaruan bias
α : Laju Pembelajaran (*Learning Rate*)

2.6 Pengujian Akurasi

Pada tahap pengujian akurasi akan dilakukan perhitungan nilai akurasi atau kecocokan dari data baru yang masuk ke dalam aplikasi dengan data yang sudah dilatih sebelumnya. Adapun metode perhitungan akurasi yang akan dilakukan yaitu dengan metode *Confusion Matrix* [19]. Untuk menghitung akurasi terdapat 3 rumus, yaitu Akurasi, *Precision*, dan *Recall*. Adapun rumus menghitung akurasi, *presicion*, dan *recall* pada *confusion matrix* menggunakan persamaan sebagai berikut.

$$\text{Akurasi} = \frac{\sum_{i=1}^{L} \frac{TP_i + TN_i}{TP_i + FN_i + FP_i + TN_i}}{L} \times 100\% \quad (29)$$

$$Presicion = \frac{\sum_{i=1}^{L} \frac{TP_i}{TP_i + FP_i}}{L} \times 100\% \quad (30)$$

$$Recall = \frac{\sum_{i=1}^{L} \frac{TP_i}{TP_i + FN_i}}{L} \times 100\% \quad (31)$$

Keterangan:
TP_i : *True Positive* ke-i
TN_i : *True Negative* ke-i
FP_i : *False Positive* ke-i
FN_i : *False Negative* ke-i
L : Jumlah Keseluruhan Data Yang Diuji/Jumlah Kelas

Pengujian dilakukan dengan menggunakan nilai *learning rate* mulai dari 0.0001 sampai 0.0009 dengan penambahan setiap nilai sebesar 0.0002 serta nilai maksimum *epoch* sebesar 100 untuk setiap *learning rate*. Berikut ini merupakan hasil rekapitulasi dari keseluruhan proses pengujian yang sudah dilakukan.

Tabel 1 Rekapitulasi Pengujian Akurasi

No	*Learning Rate*	Akurasi	*Precision*	*Recall*
1	0.0001	93.46%	94.66%	93.46%

2	0.0003	92.69%	94.27%	92.69%
3	0.0005	96.92%	97.23%	96.92%
4	0.0007	96.92%	97.29%	96.92%
5	0.0009	95.00%	96.54%	95.00%

2.7 Temuan Pengujian

Pada saat proses pengujian, dilakukan dua kali percobaan yang bertujuan untuk membandingkan nilai akurasi dengan konsentrasi penambahan jumlah lapisan pada arsitektur CNN tanpa merubah nilai parameter yang sudah ditentukan sebelumnya seperti nilai *epoch* dan *learning rate*. Pada percobaan pertama, *convolution layer* pada arsiterktur CNN berjumlah 2 *layer* dan mendapatkan rata-rata akurasi sebesar 83%. Pada percobaan kedua, lapisan pada *convolution layer* ditambahkan 2 *layer* sehingga jumlah *layer*-nya menjadi 4 *layer*. Hasilnya didapat rata-rata akurasi sebesar 95% dan mendapat akurasi tertinggi sebesar 96.92%. Dari penjelasan tersebut dapat ditemukan bahwa jumlah *layer* pada arsitektur CNN dapat mempengaruhi tingkat akurasi pada pengujian.

3. PENUTUP

3.1 Kesimpulan

Berdasarkan seluruh rangkaian penelitian yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut.

1. Sistem pengenalan pola sandi rumput dengan menggunakan metode *Convolutional Neural Network* diperoleh hasil akurasi terbaik sebesar 96.92%.
2. Tingkat akurasi ini dipengaruhi oleh nilai *learning rate* pada proses pelatihan, selain itu tingkat kerapihan pola sandi rumput, jumlah *dataset* dan jumlah *layer* pada arsitektur CNN pun berpengaruh terhadap tingkat akurasi.
3. Sistem dapat mengenali pola sandi rumput.

3.2 Saran

Adapun saran yang dapat digunakan dalam pengembangan penelitian selanjutnya adalah sebagai berikut.

1. Menerapkan teknik *preprocessing* yang berbeda dan menambahkan metode ekstraksi ciri agar proses pengenalan semakin baik.
2. Menggunakan teknik segmentasi yang dapat memproses pola yang bersambung.
3. Menggunakan teknik *Deep Learning* yang berbeda untuk proses klasifikasi, seperti R-CNN, Faster R-CNN dan GAN (*Generative Adversary Network*)
4. Memperkaya dan memvariasikan data latih agar pengenalan lebih baik lagi
5. Menggunakan perangkat keras dengan spesifikasi yang tinggi agar proses pelatihan dan pengujian dapat berjalan dengan performa yang baik.

DAFTAR PUSTAKA

[1] S. Zubair and A. Solichin, "Pengenalan karakter sandi rumput pramuka menggunakan jaringan saraf tiruan dengan metode backpropagation," *Seminar Nasional Teknologi Informasi dan Multimedia*, vol. 3, no. 9, pp. 1–6, 2017.

[2] R. Sofian Bahri and I. Maliki, "Perbandingan Algoritma Template Matching dan Feature Extraction pada Optical Character Recognition," *Jurnal Komputer dan Informatika, vol. 1, no. 1, pp. 29–35*, 2012.

[3] M. C. Wibowo and S. Wirakusuma, "Pengenalan Pola Tulisan Tangan Aksara Jawa," *SNASTI 2013*, vol. 3, no. 1, pp. 27–32, 2013.

[4] U. Rohwana and M. isa Irawan, "Pengenalan Tulisan Tangan Huruf Latin Bersambung Secara Real Time Menggunakan Algoritma Learning Vector Quantization," *Jurnal Sains Dan Seni Pomits*, vol. 2, no. 1, pp. 1–6, 2013.

[5] Suyanto, *Machine Learning: Tingkat Dasar dan Lanjut*. Bandung: Penerbit Informatika, 2018.

[6] M. L. Afakh, A. Risnumawan, M. E. Anggraeni, M. N. Tamara, and E. S. Ningrum, "Aksara jawa text detection in scene images using convolutional neural network," *Proceedings - International Electronics Symposium on Knowledge Creation and Intelligent Computing IES-KCIC 2017, vol. 2017-Janua, pp. 77–82*, 2017.

[7] C. K. Dewa, A. L. Fadhilah, and A. Afiahayati, "Convolutional Neural Networks for Handwritten Javanese Character Recognition," *IJCCS*, vol. 12, no. 1, p. 83, 2018.

[8] I. Ramadhan, "Pengenalan Pola Citra Tulisan Tangan Aksara Sunda dengan Metode Convolutional Neural Network," *SKRIPSI Program Studi Teknik Informatika UNIKOM*, 2018.

[9] M. & B. S. Zufar, "Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time," *Jurnal Sains dan Seni*, vol. 5, no. 3, pp. 1–6, 2016.

[10] Q. Zhao, Y. Li, N. Yang, Y. Yang, and M. Zhu, "A convolutional neural network approach for semaphore flag signaling recognition," *2016 IEEE International Conference on Signal and Image*

Processing. ICSIP 2016, pp. 466–470, 2017.
[11] P. Hidayatullah, *Pengolahan Citra Digital Teori dan Aplikasi Nyata*. Bandung: Penerbit Informatika, 2018.
[12] A. Kadir and A. Susanto, *Teori dan Aplikasi Pengolahan Citra*. Yogyakarta: Penerbit ANDI, 2013.
[13] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*, 1st ed. Wiley Publishing, 2011.
[14] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognit.*, vol. 33, no. 2, pp. 225–236, 2000.
[15] N. I. Widiastuti, "Deep Learning - Now and Next in Text Mining and Natural Language Processing," *IOP Conference Series: Materials Science and Engineering*, vol. 407, no. 1, 2018.
[16] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proceedings of the IEEE International Conference on Computer Vision.*, vol. 2015 Inter, pp. 1026–1034, 2015.
[17] P. Sadowski, "Notes on Backpropagation," *Department of Computer Science University of California Irvine*, vol. 1, no. 4, pp. 1–3, 2017.
[18] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ICLR 2015*, pp. 1–15, 2015.
[19] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.