PENGENALAN POLA CITRA TULISAN TANGAN AKSARA SUNDA DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

Imam Ramadhan[1], Galih Hermawan[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
Email : imamramadhan548@gmail.com[1], galih.hermawan@email.unikom.ac.id[2]

ABSTRAK

Aksara sunda merupakan huruf asli yang digunakan oleh suku Sunda untuk menuliskan kata-kata dalam Bahasa Sunda yang digunakan, *aksara Sunda juga cukup rumit dan unik dalam penulisannya.* Terkait dengan pola tulisan tangan aksara Sunda yang unik, maka dilakukanlah penelitian tentang pengenalan pola citra tulisan tangan aksara Sunda. Pada penelitian sebelumnya terkait dengan pengenalan pola tulisan tangan belum ditemukannya hasil yang optimal dari penggunaan metode-metode sebelumnya yang membahas tentang *Handwritting Recognition*, maka dari itu dibutuhkan sebuah metode yang mampu meningkatkan akurasi dalam hal pengenalan pola tulisan tangan. *Convolutional Neural Network* (CNN) adalah metode klasifikasi yang akan digunakan didalam penelitian ini. Pada penelitian sebelumnya CNN telah membuktikan keefektifitasannya dalam hal mendeteksi tulisan aksara Jawa pada gambar scence, dan memiliki hasil akurasi yang bagus untuk mengenali wajah secara real-time. Setelah melakukan pengujian sistem dengan metode *Black Box* dapat ditarik kesimpulan bahwa program berfungsi dengan baik. Pada proses pengujian dari 3 kombinasi pengujian yang berbeda dengan total 250 karakter yang telah disiapkan, program mampu mendeteksi tulisan tangan aksara Sunda dengan akurasi sebesar 62% pada kombinasi learning rate 0,1 dan epoch 10.

Kata kunci : Pengenalan Pola, Aksara Sunda, *Convolutional Neural Network.* Tulisan Tangan, Pengolahan Citra.

1. PENDAHULUAN

Aksara sunda merupakan huruf asli yang digunakan oleh suku Sunda untuk menuliskan kata-kata dalam Bahasa Sunda yang digunakan, selain memiliki karakter huruf yang unik, aksara sunda juga cukup rumit dalam penulisannya. Terkait dengan pola tulisan aksara Sunda yang unik, maka dilakukanlah penelitian tentang pengenalan pola citra tulisan tangan aksara Sunda.

Tulisan tangan merupakan salah satu hal unik yang dimiliki setiap orang karena tulisan seseorang yang satu dengan lainnya pasti berbeda, meskipun mendekati sama. Berdasarkan perbedaan tersebut banyak penelitian yang sudah dilakukan terkait tulisan tangan. Penelitian sebelumnya terkait dengan pengenalan citra tulisan tangan dapat mengenali huruf latin bersambung [1] dan pengenalan wajah secara real-time [2].

Penelitian ini dilakukan bertujuan untuk mengetahui seberapa akurat metode yang digunakan dalam penelitian ini untuk mengenali pola tulisan tangan, karena di penelitian sebelumnya belum ditemukan hasil optimal dari penggunaan metode-metode sebelumnya yang membahas kajian tentang pengenalan tulisan tangan *Handwritting Recognition*. Pada penelitian tersebut mendapatkan rata-rata akurasi terbaik sebesar 55,7 % [1].

Convolutional Neural Network (CNN) adalah metode klasifikasi yang akan digunakan didalam penelitian ini, CNN merupakan variasi dari *Multilayer Perceptron* (MLP) yang terinsfirasi dari jaringan saraf manusia [2]. Penelitian awal yang mendasari penelitian ini pertama kali dilakukan oleh Hubel dan Wiesel [3]. Secara umum jenis layer pada CNN dibedakan menjadi dua yaitu layer ekstraksi fitur gambar dan layer klasifikasi. Pada penelitian sebelumnya CNN telah membuktikan ke efektifitasannya dalam hal mendeteksi text aksara Jawa pada gambar scence [4] dan mendapatkan 87,48% akurasi untuk mengenali wajah secara real-time [2].

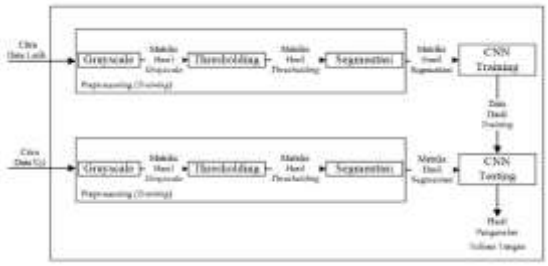
Maka, dalam penelitian ini akan dibuat aplikasi pengenalan pola tulisan tangan aksara sunda dengan menggunakan metode *Convolutional Neural Network* (CNN). Kemudian setelahnya akan dilakukan perhitungan akurasi dilakukan dengan menghitung banyaknya pola tulisan tangan aksara sunda yang dapat dikenali oleh aplikasi.

2. ISI PENELITIAN

Masalah dari penelitian ini yaitu belum diketahuinya akurasi pengenalan pola tulisan tangan bahasa Sunda dengan metode klasifikasi *Convolutional Neural Network* (CNN). Metode CNN berdasarkan penelitian sebelumnya belum pernah dikombinasikan secara lengkap dengan metode *Sauvola Thresholding* dalam kasus pengenalan tulisan tangan aksara Sunda, dalam penelitian ini Metode CNN akan dikombinasikan dengan metode *Sauvola Thresholding* dengan menggabungkan kedua algoritma secara bersamaan kedalam penelitian,

kedua metode ini cukup baik pencapaian hasil akurasinya. Kedua metode ini digunakan untuk mengatasi permasalahan mengenai pengenalan tulisan tangan berbentuk huruf cetak. Proses *Thresholding* sangat penting dalam penelitian ini karena dari proses itulah baru bisa dilakukan metode klasifikasi dan mengetahui akurasi dari pengenalan tulisan tangan [5], sehingga perlu dilakukan proses *Sauvola Thresholding* terlebih dahulu.

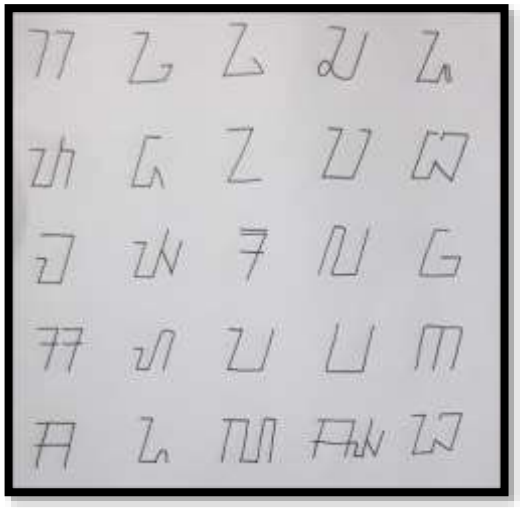
Sistem yang akan dibangun yaitu terbagi menjadi dua proses yaitu proses pelatihan dan pengujian, pada proses pelatihan dan pengujian didalamnya terdapat *preprocessing* yang meliputi *grayscale*, *sauvola threshold*, dan segmentasi citra dan setelah preprocessing selesai baru kemudian diakukan proses pelatihan begitupun juga pengujian. Untuk data masukannya yang akan digunakan yaitu citra tulisan tangan dengan format gambar .jpg. Citra dapat diperoleh melalui scan atau foto dengan resolusi kamera minimal 13 *megapixel.* Citra tulisan tangan yang digunakan ditulis menggunakan pulpen/spidol hitam dalam kertas HVS dengan *background* putih. Sedangkan citra yang digunakan hanya Aksara Sunda ngalagena saja. Berikut adalah arsitektur umum sistem yang akan dibangun.



Gambar 1 Gambaran Umum Sistem

2.1. Data Masukan

Citra tulisan tangan yang digunakan sebagai data latih merupakan citra tulisan yang tersusun dari karakter ka, ga, nga, ca, ja, nya, ta, da, na, pa, ba, ma, ya, ra, la, wa, sa, ha, fa, va, qa, xa, za, kha dan sya. Data masukan atau data latih ditulis diatas kertas dengan background putih polos. Dalam penelitian ini citra karakter yang digunakan hanya tiga karakter yaitu ca, ja dan nya digunakan untuk mempermudah peneliti dalam memvisualisasikan nilai-nilai yang dihasilkan dari setiap proses.



Gambar 2 Data Masukan

2.2. Pengolahan Citra Digital

Pengolahan citra digital artinya melakukan pemrosesan gambar berdimensi dua melalui komputer digital. pengolahan citra adalah istilah umum untuk berbagai teknik yang keberadaanya untuk memanipulasi dan memodifikasi citra dengan berbagai cara [6].Foto adalah contoh gambar berdimensi dua yang dapat diolah dengan mudah. Setiap foto dalam bentuk citra digital dapat diolah menggunakan perangkat lunak tertentu. Misalnya citra yang gelap dapat diolah supaya menjadi lebih terang, merubah format warna citra menjadi *grayscale*, *threshold*, *binerisasi*, segmentasi, *resize* citra dan lain sebagainya.

Pada pengolahan citra digital dalam penelitian ini bias disebut preprocessing. Pada proses *preprocessing* data masukan citra baik citra latih maupun citra uji akan diolah dalam beberapa tahapan yaitu *grayscale*, *thresholding*, *binerisasi*, segmentasi dan *resize*. Proses ini bertujuan untuk memenuhi syarat yang harus terpenuhi dalam penelitian ini sebelum dilakukan proses pelatihan maupun pengujian.

2.2.1 *Grayscale*

Grayscale bertujuan untuk menangani gradasi warna hitam dan putih, dan hasilnya tentu saja akan merubah warna (RGB) menjadi efek warna abu-abu. Warna gambar dinyatakan dengan intensitas [7]. Pada proses ini, intensitas berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih. Mengubah format warna menjadi *grayscale* berfungsi untuk mengecilkan range warna dari 0 sampai dengan 255. Berikut adalah rumus konversi citra berwarna (RGB) menjadi *grayscale*.

$$I = (0.2989*R)+(0.587*G)+(0.1141*B) \quad (1)$$

Keterangan:

I = Fungsi pencarian nilai skala keabu-abuan
R = Komponen nilai merah (*Red*) dari suatu titik *pixel*
G = Komponen nilai hijau (*Green*) dari suatu titik *pixel*
B = Komponen nilai biru (*Blue*) dari suatu titik *pixel*

Persamaan diatas dipilih dalam penelitian ini karena mata manusia secara alami lebih sensitif terhadap cahaya merah dan hijau. Maka dari itu, warna-warna ini diberi bobot yang lebih tinggi untuk memastikan bahwa keseimbangan intensitas relatif dalam citra grayscale yang dihasilkan mirip dengan citra warna RGB [7].



Gambar 3 Contoh Hasil *Grayscale*

2.2.2 *Thresholding*

Sauvola threshold merupakan metode *threshold* yang mampu mendeteksi citra tulisan tangan dengan sangat baik, dengan cara menghitung ambang menggunakan rentang nilai dinamis dari nilai standar deviasi citra *grayscale* [8]. *Sauvola* termasuk dalam *local threshold* dimana nilai ambang ditentukan oleh nilai pixel tetangga, tujuan dilakukannya proses *threshold* adalah untuk menyederhanakan bentuk citra. Berikut adalah rumus yang digunakan pada proses *thresholding*.

$$T(x,y) = m(x,y) * (1 + k * (\frac{s(x,y)}{R} - 1)) \quad (2)$$

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

$$s(x,y) = \sqrt{\frac{\sum_{i=min}^{i=max}\sum_{j=min}^{j=max} (img(i,j)-m(x,y))^2}{(i*j)-1}} \quad (4)$$

Keterangan:

R = Nilai maksimum dari standar deviasi (128 untuk citra *grayscale*)
k = Kernel dengan nilai antara 0.2 – 0.5
m = Fungsi yang menghasilkan nilai rata-rata dari sejumlah *pixel* citra
s = Fungsi yang menghasilkan nilai standar deviasi dari sejumlah *pixel* citra
T = Fungsi yang menghasilkan nilai *threshold*
x = Nilai Koordinat lebar citra dalam rumus (i)
y = Nilai koordinat tinggi citra dalam rumus (j)

Setelah diproses maka didapatkan nilai ambang T(x,y), untuk proses selanjutnya yaitu masukkan hasil tadi ke persamaan berikut.

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

Keterangan:

f = Fungsi yang menghasilkan nilai 0 atau 255
Img = Nilai citra *grayscale*



Gambar 4 Contoh Hasil Sauvola *Threshold*

2.2.3 Segmentasi

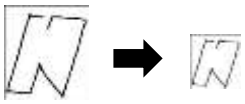
Segmentasi merupakan teknik untuk membagi suatu citra menjadi beberapa daerah (*region*) dimana setiap daerah memiliki kemiripan atribut [7]. Beberapa teknik segmentasi antaralain meliputi pengambangan, penandaan komponen terhubung, segmentassi berbasis cluster, dan transformasi hough. Fokus pada segmentasi citra pada penelitian ini adalah menggunakan pengambangan dan penandaan komponen terhubung. Pada proses ini, input yang digunakan yaitu citra hitam putih hasil *sauvola threshold.* Selanjutnya akan dilakukan pemotongan untuk mendapatkan citra huruf tulisan tangan.



Gambar 2 Contoh Segmentasi

2.2.4 *Resize*

Resize yaitu merubah pixel atau ukuran suatu gambar sesuai dengan yang dibutuhkan. Pada prosesnya resize merubah ukuran pixel bukan hanya memperkecil tetapi juga bisa memperbesar ukuran pixel, selain itu resize juga dapat memperbaiki bahkan mungkin memperburuk kualitas citra suatu gambar sesuai. Pada penelitian ini *Resize/*Normalisasi pixel hasil segmentasi diperkecil menjadi ukuran 6x6.

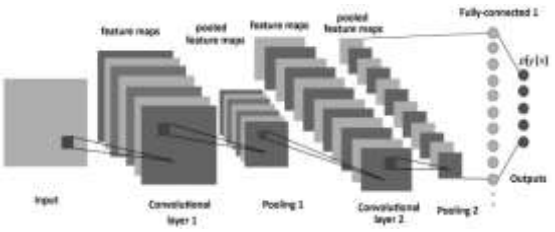


Gambar 3 Contoh Resize

2.2.5 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu jenis *neural network* yang biasa digunakan

pada data image. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada sebuah citra [9]. Arsitektur dari CNN dibagi menjadi 3 bagian besar, *Convolutional Layer*, *Pooling Layer*, dan *Fully-Connected Layer*. Berikut arsitektur *convolutional neural network.*



Gambar 7 arsitektur CNN

A. *Convolutional Layer*

Convolutional layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sekumpulan filter dengan panjang dan tinggi (*pixels*). Masing-masing filter ini akan digeser keseluruh bagian dari gambar satu persatu. Setiap pergeseran akan dilakukan operasi "dot" antara input dan nilai dari filter tersebut sehingga menghasilkan sebuah output atau biasa disebut sebagai *activation map* atau *feature map*. Pada *convolutional layer* akan dilakukan konvolusi antara matriks citra masukan dengan matriks-matriks filter F. Filter-filter ini akan digeser ke seluruh permukaan gambar sehingga menghasilkan keluaran matriks *feature map* FM.
Dengan *hyperparameter* di atas maka akan dihasilkan *feature map* FM dengan ukuran yang didapat dari rumus berikut:

$$featuresize = \frac{inputsize - filtersize + 2pad}{str} + 1 \quad (6)$$

Keterangan:
featurezise = Ukuran *feature map*
inputsize = Ukuran matriks masukan
filtersize = Ukuran filter
Pad = *zero padding*
str = *stride*

Langkah-langkah yang dilakukan pada *convolutional layer* adalah sebagai berikut.

$$FM_{x',y'} = \sum_{x=1}^{size} \sum_{y=1}^{size} (A_{x,y} * F_{x,y}) + bF \quad (7)$$

Keterangan:
$FM_{x',y'}$ = pixel *feature map* ke x', y'
$A_{x,y}$ = pixel matriks input yang telah diberi *zero padding* ke x, y
$F_{x,y}$ = pixel filter ke x, y
size = Ukuran filter

Lalu jalankan fungsi aktivasi pada tiap *feature map* FM. Setiap pixel *feature map* FM akan dimasukkan ke dalam fungsi aktivasi ReLU, dimana setiap pixel yang memiliki nilai < 0 akan dijadikan 0, dengan rumus.

$$ReLU(x) = \max(0, x) \quad (8)$$

Keterangan:
$ReLU(x)$ = Fungsi aktivasi ReLU
x = Pixel

B. *Pooling Layer*

Pooling layer terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan bergeser pada seluruh area feature map. *Pooling* yang akan digunakan dalam penelitian ini adalah *Max Pooling*. Tujuan dari penggunaan *pooling layer* adalah mengurangi dimensi dari *feature map* (*downsampling*), sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit. Cara kerja *pooling layer* adalah sebagai berikut. Letakkan jendela pada pojok kiri atas *feature map.* Berikut adalah rumusnya.

$$P_{x',y'} = \max(R_{x,y}, \dots, R_{x+window,y+window}) \quad (9)$$

Keterangan:
$P_{x',y'}$ = pixel *feature map pooling* ke x', y'
$R_{x,y}$ = pixel *feature map ReLU* ke x, y
Window = ukuran jendela

2.2.6 *Fully Connected Layer*

Fully Connected Layer pada intinya adalah sebuah neural netrowk multilayer perceptron (MLP), yang memiliki beberapa *hidden layer*, *activation function*, *output layer* dan *loss function*. *Fully connected layer*-lah yang akan berperan untuk mengklasifikasi data masukan. Output yang dihasilkan dari *pooling layer* masih berbentuk *multidimensional array*, sehingga akan di-*flatten* terlebih dahulu untuk mengubah data menjadi vektor sebelum dijadikan input untuk *fully connected layer*.

A. *Input Layer*

Input layer X adalah penggabungan matriks-matriks *feature map* P yang didapat dari *pooling layer* dan direntangkan menjadi sebuah vektor sepanjang jumlah pixel keseluruhan ketiga *feature map.* Total *node input layer* adalah jumlah dari *pixel feature map* P. Nilai-nilai *node input layer* X akan digunakan pada perhitungan *hidden layer* Z.

B. *Hidden Layer*

Setiap *node* di *input layer* X akan mengirimkan sinyal input kepada setiap *hidden layer* Z. Setiap *node* X_i akan dikalikan dengan bobot $V_{j,i}$ lalu ditambahkan dengan bias $V_{0,i}$ untuk menghasilkan nilai masukan z_in_i. Rumus penghitungan masukan *hidden layer* z_in adalah sebagai berikut.

$$z_in_i = \sum_{i=1}^{n} X_j * V_{j,i} + V_{0,i} \quad (10)$$

Keterangan:

z_in_i = masukan untuk *node hidden layer Z* ke-*i* dengan jumlah node *n*

X_j = *node X* ke-*j*

$V_{j,i}$ = *weight V* untuk *node Xj* dan *node Zi*

$V_{o,i}$ = bias *V* untuk *node* z_in_i

Setelah itu hitung nilai keluaran Z dengan mengaktifkan nilai masukan z_in menggunakan fungsi aktivasi ReLU, sebagai berikut.

$$Z_i = ReLU(z_in_i) = \max(0, z_in_i) \quad (11)$$

Keterangan:

Z_i = keluaran untuk *node hidden layer* ke-*i*

z_in_i = masukan untuk *node hidden layer* ke-*i*

C. *Output Layer*

Setiap *node* di *output layer* Z_j akan mengirimkan sinyal input kepada setiap *output layer* Y_i. Setiap *node* Z akan dikalikan dengan bobot $W_{j,i}$ lalu ditambahkan dengan bias $W_{0,i}$. Penghitungan masukan *output layer* y_in adalah sebagai berikut.

$$y_{in_i} = \sum_{i=1}^{n} Z_j * W_{j,i} + W_{0,i} \quad (12)$$

Keterangan:

y_in_i = masukan untuk *node hidden layer Z* ke-*i* dengan jumlah node *n* buah

Z_j = *node Z* ke-*j*

$W_{j,i}$ = *weight W* untuk *node Zj* dan *node Yi* bias *V* untuk *node* z_in_i

$W_{o,i}$ = bias W untuk *node* y_in_i

Setelah itu aktifkan nilai keluaran Y dengan menggunakan fungsi aktivasi *softmax*.

$$Y_i = \frac{e^{y_{in_i}}}{\sum_{i=1}^{m} e^{M}} \quad (13)$$

Keterangan:

Y_i = keluaran untuk *node output layer*

y_in_i = masukan untuk *node output layer* ke-*i*

M = semua masukan untuk *node output layer*, berjumlah *m* buah

2.2.7 *Backpropagation*

Pada tahap *backpropagation* akan dilakukan pelatihan pada kelas huruf. Pada tahap ini akan dilakukan proses penyesuaian tiap *weight* dan *bias* berdasarkan *error* yang didapat pada tahap *feedforward.* Pertama akan dihitung gradien dari *loss function* terhadap semua parameter (*weight* dan *bias*) yang ada dengan mencari turunan parsial (*partial derivative*) dari fungsi tersebut. Setelah itu *update* semua parameter dengan menggunakan *Stochastic Gradient Descent (SGD).*

A. Perhitungan *Loss Function*

Loss function adalah sebuah fungsi yang mengukur seberapa bagus performa dari *neural network* dalam melakukan prediksi terhadap target. *Loss function* akan menghitung *loss*, yaitu selisih dari nilai *output* dengan nilai target yang diharapkan. Untuk masalah klasifikasi, *loss function* yang biasa digunakan adalah *cross-entropy loss function* :

$$L = -\sum_{i=1}^{m} t_i \log(Y_i) \quad (14)$$

Keterangan:

L = *loss / error*

t_i = nilai vektor ke-i *one-hot* target yang diharapkan

Y_i = nilai matriks output pada tahap *feedforward* ke-i dengan jumlah kelas *m* buah

B. Perhitungan Gradien Terhadap Parameter W_{ji}

Setelah itu hitung gradien kesalahan terhadap parameter bobot W_{ji}, dengan menggunakan rumus *chain rule*.

$$\frac{\partial L}{\partial W_{ji}} = \frac{\partial L}{\partial Y_i} \frac{\partial Y_i}{\partial y_in_i} \frac{\partial y_in_i}{\partial W_{ji}} \quad (15)$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1 - Y_i)} \quad (16)$$

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

$$\frac{\partial y_in_i}{\partial W_{ji}} = Z_j \quad (18)$$

Maka untuk filter W_{ji}

$$\frac{\partial L}{\partial W_{ji}} = Y_i(1 - Y_i)Z_j \quad (19)$$

dan untuk bias $W_{0,i}$

$$\frac{\partial L}{\partial W_{ji}} = Y_i(1 - Y_i) \quad (20)$$

Keterangan:

$\frac{\partial L}{\partial W_{ji}}$ = Gradien bobot W_{ji} terhadap *loss* L

$\frac{\partial L}{\partial W_{0,i}}$ = Gradien bias $W_{0,i}$ terhadap *loss* L

$\frac{\partial L}{\partial Y_i}$ = Gradien nilai *node* Y_i terhadap *loss* L

$\frac{\partial Y_i}{\partial y_in_i}$ = Gradien nilai matriks y_in_i terhadap nilai *node* Y_i

$\frac{\partial y_{in_i}}{\partial W_{ji}}$ = Gradien bobot W_{ji} terhadap nilai matriks y_in_i

Y_i = nilai *node output layer* ke-i

y_{in_i} = nilai masukan *output layer* ke-i

Z_j = nilai *node hidden layer* ke-j

C. Penghitungan Gradien Terhadap Parameter Bobot V_{kj} dan Bias $V_{0,j}$

Setelah itu hitung gradien kesalahan terhadap parameter bobot V_{kj}, dengan menggunakan rumus *chain rule*.

$$\frac{\partial L}{\partial V_{kj}} = \sum_i^m \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y_in_i}\frac{\partial y_in_i}{\partial Z_j}\frac{\partial Z_j}{\partial z_in_j}\frac{\partial z_in_j}{\partial V_{kj}} \quad (21)$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1-Y_i)} \quad (22)$$

$$\frac{\partial Y_i}{\partial y_in_i} = Y_i(1-Y_i) \quad (23)$$

$$\frac{\partial y_in_i}{\partial Z_j} = W_{ji} \quad (24)$$

$$\frac{\partial Z_j}{\partial z_in_j} = Z_j\left(1-Z_j\right) \quad (25)$$

$$\frac{\partial z_in_j}{\partial V_{kj}} = X_k \quad (26)$$

Maka untuk filter V_{kj}

$$\frac{\partial L}{\partial V_{kj}} = \sum_i^m (Y_i - t_i)(W_{ji})(Z_j\left(1-Z_j\right))(X_k) \quad (27)$$

dan untuk bias $V_{0,j}$

$$\frac{\partial L}{\partial V_{kj}} = \sum_i^m (Y_i - t_i)(W_{ji})(Z_j\left(1-Z_j\right)) \quad (28)$$

Keterangan:

$\frac{\partial L}{\partial W_{ji}}$ = Gradien bobot V_{kj} terhadap *loss* L

$\frac{\partial L}{\partial W_{0,i}}$ = Gradien bias $V_{0,i}$ terhadap *loss* L

$\frac{\partial L}{\partial Y_i}$ = Gradien nilai *node* Y_i terhadap *loss* L

$\frac{\partial Y_i}{\partial y_in_i}$ = Gradien nilai matriks y_in_i terhadap nilai *node* Y_i

$\frac{\partial y_{in_i}}{\partial W_{ji}}$ = Gradien bobot W_{ji} terhadap nilai matriks y_in_i

Y_i = nilai *node output layer* ke-i

y_{in_i} = nilai masukan *output layer* ke-i

Z_j = nilai *node hidden layer* ke-j

D. Penghitungan gradient terhadap parameter filter F dan bias bF

Setelah itu hitung gradien kesalahan terhadap parameter filter F. Sesuai aturan *chain rule*, sebelumnya kita harus mencari gradien untuk *layer-layer* sebelumnya terlebih dahulu. Lalu hitung gradien masing-masing filter dengan gradien yang didapat sebelumnya.

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial X}\frac{\partial X}{\partial P}\frac{\partial P}{\partial F} \quad (29)$$

Keterangan:

$\frac{\partial L}{\partial F}$ = gradien nilai filter F terhadap *loss* L

$\frac{\partial L}{\partial X}$ = gradien nilai *node* X terhadap *loss* L

$\frac{\partial X}{\partial P}$ = gradien nilai *feature map* P terhadap nilai *node* X

$\frac{\partial P}{\partial F}$ = gradien nilai filter F terhadap nilai *feature map* P

Pertama hitung nilai $\frac{\partial L}{\partial X}$. Hitung gradien kesalahan terhadap parameter bobot X_k, dengan menggunakan rumus *chain rule*.

$$\frac{\partial L}{\partial X_k} = \sum_j^n \sum_i^m \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y_in_i}\frac{\partial y_in_i}{\partial Z_j}\frac{\partial Z_j}{\partial z_in_j}\frac{\partial z_in_j}{\partial X_k} \quad (30)$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1-Y_i)} \quad (31)$$

$$\frac{\partial Y_i}{\partial y_{in_i}} = Y_i(1-Y_i) \quad (32)$$

$$\frac{\partial y_in_i}{\partial Z_j} = W_{ji} \quad (33)$$

$$\frac{\partial Z_j}{\partial z_in_j} = Z_j\left(1-Z_j\right) \quad (34)$$

$$\frac{\partial z_in_j}{\partial X_k} = V_{kj} \quad (35)$$

Maka

$$\frac{\partial L}{\partial X_k} = \sum_j^n \sum_i^m (Y_i - t_i)(W_{ji})(Z_j\left(1-Z_j\right))(V_{kj}) \quad (36)$$

Keterangan:

$\frac{\partial L}{\partial X_k}$ = Gradien nilai *node* X_k terhadap *loss* L

$\frac{\partial L}{\partial Y_i}$ = Gradien nilai *node* Y_i terhadap *loss* L

$\frac{\partial L}{\partial Y_i}$ = Gradien nilai *node* Y_i terhadap *loss* L

$\frac{\partial Y_i}{\partial y_in_i}$ = Gradien nilai matriks y_in_i terhadap nilai *node* Y_i

$\frac{\partial y_{in_i}}{\partial Z_j}$ = Gradien bobot W_{ji} terhadap nilai matriks y_in_i

$\frac{\partial Z_j}{\partial z_{in_j}}$ = Gradien nilai matriks z_{in_j} terhadap nilai *node* Z_j

$\frac{\partial z_{in_j}}{\partial X_k}$ = gradien nilai *node* X_k terhadap nilai matriks z_{in_j}

Y_i = nilai *node output layer* ke-i

y_{in_i} = nilai masukan *output layer* ke-i

Z_j = nilai *node hidden layer* ke-j

z_{in_j} = nilai masukan *hidden layer* ke-j

X_k = nilai *node input layer* ke-k

m = jumlah *node output layer* Y

n = jumlah *node hidden layer* Z

Setelah itu kita hitung hitung nilai $\frac{\partial X}{\partial P}$ dengan menggunakan nilai yang didapat dari perhitungan $\frac{\partial L}{\partial X}$. Nilai ini didapat dari mencari gradien pada layer *pooling layer*. Pertama kita kembalikan lagi bentuk vektor $\frac{\partial L}{\partial X}$ menjadi gradien *feature map*. Untuk operasi *max-pooling feature map* P', gradien $\frac{\partial X}{\partial P}$ melihat kepada nilai-nilai maksimum *feature map* sesuai dengan hasil *max-pooling* pada tahap *feedforward*. Gradien $\frac{\partial X}{\partial P}$ untuk pixel-pixel dengan

nilai maksimum adalah $\frac{\partial L}{\partial X}$, sedangkan gradien untuk pixel-pixel lainnya adalah 0. Sedangkan untuk gradien bias filter, jumlahkan setiap nilai pada gradien filter. Lakukan untuk semua filter.

$$\frac{\partial L}{\partial bF} = \sum_x^{size} \sum_y^{size} \frac{\partial P_{x,y}}{\partial F_{x,y}} \quad (37)$$

Keterangan:

$\frac{\partial L}{\partial bF}$ = Gradien bias filter bF terhadap *loss* L

$\frac{\partial P_{x,y}}{\partial F_{x,y}}$ = Gradient nilai filter F ke-x,y terhadap nilai *feature map* P ke-x,y

size = ukuran filter

E. Update Nilai Parameter

Setelah semua gradien parameter dihitung, *update* parameter-parameter dengan menggunakan *Stochastic Gradient Descent*. *Stochastic Gradient Descent* adalah metode optimasi yang digunakan untuk menemukan nilai bobot optimal agar didapatkan nilai *loss* yang kecil, di mana nilai *loss* merepresentasikan tingkat kesalahan (*error*) pada sistem. Pencarian nilai bobot dilakukan dengan cara merubah bobot secara bertahap ketika pelatihan, dengan menggunakan parameter yang disebut *learning rate*, yaitu seberapa cepat *neural network* melakukan pembelajaran. Berikut adalah perhitungan untuk *update* nilai parameter bobot W'.

$$W_{x,y}{}' = W_{x,y} - \alpha(\frac{\partial L}{\partial W_{x,y}}) \quad (38)$$

Keterangan:

$W_{x,y}{}'$ = Bobot W ke-x,y yang telah di-*update*

$W_{x,y}$ = Bobot W ke-x,y awal

$\frac{\partial L}{\partial W_{x,y}}$ = Gradien bobot $W_{x,y}$ terhadap *loss* L

α = *learning rate*

Berikut adalah perhitungan untuk *update* nilai parameter bias W_0'.

$$V_{x,y}{}' = V_{x,y} - \alpha(\frac{\partial L}{\partial V_{x,y}}) \quad (39)$$

Keterangan:

$V_{x,y}{}'$ = Bobot V ke-x,y yang telah di-*update*

$V_{x,y}$ = Bobot V ke-x,y awal

$\frac{\partial L}{\partial V_{x,y}}$ = Gradien bobot $V_{x,y}$ terhadap *loss* L

α = *learning rate*

Berikut adalah perhitungan untuk *update* nilai parameter bias V_0'.

$$V_{0,y}{}' = V_{0,y} - \alpha(\frac{\partial L}{\partial V_{x,y}}) \quad (40)$$

Keterangan:

$V_{0,y}{}'$ = Bobot V_0 ke-x,y yang telah di-*update*

$V_{0,y}$ = Bobot V_0 ke-x,y awal

$\frac{\partial L}{\partial V_{0,y}}$ = Gradien bobot $V_{0,y}$ terhadap *loss* L

α = *learning rate*

Lalu berikut adalah perhitungan untuk *update* nilai parameter filter F'.

$$F_{x,y}{}' = F_{x,y} - \alpha(\frac{\partial L}{\partial F_{x,y}}) \quad (41)$$

Keterangan:

$F_{x,y}{}'$ = Bobot F' ke-x,y yang telah di-*update*

$F_{x,y}$ = Bobot F' ke-x,y awal

$\frac{\partial L}{\partial F_{x,y}}$ = Gradien bobot $F_{x,y}$ terhadap *loss* L

α = *learning rate*

Terakhir berikut adalah perhitungan untuk *update* nilai parameter filter bF'.

$$bF' = bF - \alpha(\frac{\partial L}{\partial bF}) \quad (42)$$

Keterangan:

bF' = Bobot bF ke-x,y yang telah di-*update*

bF = Bobot bF ke-x,y awal

$\frac{\partial L}{\partial bF}$ = Gradien bobot bF terhadap *loss* L

α = *learning rate*

Hasil *update* parameter-parameter bobot V', bias V_0', bobot W', bias W_0', dan filter F' dan bias filter bF' akan digunakan pada iterasi *feedforward* selanjutnya.

Pada proses pengujian untuk mengetahui hasil akurasi pengenalan hanya dilakukan perhitungan sampai feedforward saja.

2.3 Pengujian Akurasi

Pada pengujian akurasi akan dihitung nilai ketepatan atau kecocokan dari hasil aplikasi pengenalan tulisan tangan aksara Sunda. Pengujian dilakukan setelah data training sudah dilakukan proses pelatihan.

Pengujian ini dilakukan terhadap 250 data *set*. Berikut merupakan percobaan yang dilakukan dengan parameter yang berbeda pada pengujian citra.

Tabel 1 Hasil Pengujian

No.	Pengujian	Learning Rate	Epoch	Akurasi
1.	1	0,1	50	62,0%
2.	2	0,2	20	56,0%
3.	3	0,5	50	61,6%

Berdasarkan hasil pengujian akurasi terhadap 10 data uji yang telah dilakukan, evaluasi hasil akurasi yang didapatkan terhadap pengenalan tulisan tangan aksara Sunda yaitu sebagai berikut :

1. Hasil akurasi yang memiliki persentasi paling tinggi yaitu pengujian ke-1 dengan learning rate

0,1 dan epoch 50 dan didapatkan rata-rata akurasi sebesar 62%.
2. Hasil akurasi yang memiliki persentasi paling rendah yaitu pengujian ke-2 dengan learning rate 0,2 dan epoch 20 dan didapatkan rata-rata akurasi sebesar 56%.

Pengisian learning rate dan epoch mempengaruhi seberapa banyak jumlah karakter tulisan tangan yang dapat terdeteksi oleh program, namun semakin besar jumlah learing rate dan epoch membuat proses pelatihan semakin lama. Pada data uji sendiri juga mempengaruhi akurasi, semakin rapih penulisan pada data uji maka akan semakin gampang program untuk mengenali aksara Sunda.

3. PENUTUP

Pada bab ini berisi hasil dari penelitian diantaranya yaitu kesimpulan dan saran.

3.1 Kesimpulan

Pada penelitian ini telah menghasilkan sebuah sistem pengenalan tulisan tangan aksara Sunda menggunakan metode CNN. Pengenalan dilakukan terhadap 25 jenis *output* Aksara Sunda ngalagena.

Berdasarkan hasil pengujian yang telah dilakukan, pengujian terhadap sistem pengenalan tulisan tangan sebagai berikut :

1. Sistem pengenalan tulisan tangan menggunakan metode CNN diperoleh akurasi terbaik sebesar 62%. Akurasi ini dipengaruhi oleh berapa banyak layer yang digunakan di program dan dihasilkan pada proses pelatihan.
2. Jumlah learning rate dan epoch juga mempengaruhi akurasi pada proses pengujian.
3. Banyaknya karakter tulisan tangan yang tidak terdeteksi dengan baik dipengaruhi oleh proses penulisan aksara Sunda pada data uji yang jauh dari kata rapih.
4. Sistem pengenalan tulisan tangan dapat mengenali aksara Sunda.

3.2 Saran

Sistem yang saat ini dibangun masih dapat dikembangkan lagi sesuai dengan kebutuhan pengguna dalam mendapatkan hasil akurasi yang lebih baik. Saran untuk pengembangan selanjutnya pada sistem yang dapat dijadikan pertimbangan yaitu:

1. Menggunakan metode ekstraksi ciri dan klasifikasi yang lebih baik untuk mendapatkan akurasi terbaik.
2. Menggunakan metode *preprocessing* yang berbeda agar bisa membandingkan akurasi dengan penelitian yang lain.
3. Memperbanyak data latih agar bisa memperbaiki akurasi ke hasil yang lebih maksimal.
4. Membuat data latih dengan karakter tulisan yang bervariatif karena akan membantu akurasi dalam proses pengujian.

DAFTAR PUSTAKA

[1] U. Rohwana and M. I. 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.

[2] M. d. B. S.tiyono, “Convolutional Neural Network untuk Pengenal Wajah Secara Real-time,”*Jurnal Sains dan Seni ITS*, vol. 5, no. 2, pp. 72-77, 2016.

[3] W. Hubel.D, “Receptive fields and Functional Architecture of Monkey striate Cortex,” *Journal of Physiology (London)*, no. 195, pp. 215-243, 1968.

[4] M.Afakh, A.Risnumawan, MAnggraeni et al, “Aksara Jawa Text Detection in Scence Images Using Convolutional Neural Network”, *Proceedings - International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), Vol.2017-Janua*, 2017.

[5] Z.Hadjadj, A.Meziane, Y.Cherfa, “Improved Sauvola's Algorithm for Document Image Binarization,” *Reasearch Center in Scientific and Technical Information (CERIST)*, vol.9730, no.1, pp.737-745, 2016.

[6] B. B. Agarwal, “ S. P. Tayal and M,"Software Engineering & Testing”, Boston*:Jones And Bartlett Publishers,*2010.

[7] A. Kadir and A. Susanto,"Teori dan Aplikasi Pengolahan Citra", Yogyakarta: ANDI, 2013.

[8] C. Solomon, and T. Breckon,"Fundamental of Digital Image Processing", West Sussex: John Wiley & Sons, Ltd, 2011..

[9] Y.Lecun, Y.Bengio, G.Hinton, “Deep Learning,” *International Journal of Science*, vol.521, no.7534, p.436-444, 2015.