Techno.COM, Vol. 18, No. 3, Agustus 2019 : 214-226

Convolutional Neural Network untuk Pengenalan Citra Notasi Musik

Convolutional Neural Network to Detect Image of Musical Notation

Dzikry Maulana Hakim[1], Ednawati Rainarli[2]
[1,2]Universitas Komputer Indonesia; Jl. Dipati Ukur No.112-116, Lebakgede, Coblong.
Telp/Fax (022) 2504119, Kota Bandung, Kode Pos 40132
[1,2]Jurusan Tenik Informatika, Teknik dan Ilmu Komputer UNIKOM, Bandung
E-mail: [1]dzikry.maulana.hakim@email.unikom.ac.id, [2]ednawati.rainarli@email.unikom.ac.id

Abstrak

Optical Music Recognition (OMR) adalah suatu cara untuk melakukan pengenalan pada notasi musik secara otomatis. Masalah utama dalam pendeteksian notasi musik adalah bagaimana sistem dapat mendeteksi sebuah notasi musik dan kemudian mengenali notasi musik tersebut. Notasi musik yang telah dikenali oleh mesin dapat dimanfaatkan untuk diproses kembali menjadi suara. Pada penelitian ini, proses segmentasi dilakukan untuk memotong setiap notasi. Untuk pengenalan notasi musik digunakan Convolutional Neural Network (CNN). Arsitektur CNN yang dipakai adalah kernel 3x3, jumlah layer pada feature learning sebanyak 3 convolutional layer dan 3 pooling layer, filter pada convolutional layer 64,128, 256 dan jumlah neuron pada hidden layer sebanyak 7168. Pengujian dilakukan dengan dua cara, yang pertama menguji performasi CNN menggunakan data notasi musik yang telah dipotong dan yang kedua adalah melakukan pengujian menggunakan sebaris notasi musik. Nilai akurasi yang didapatkan untuk pengenalan sebaris notasi musik tidak terlalu besar, yaitu 26,19%. Walaupun untuk proses segmentasi masih belum maksimal dalam memotong setiap notasi, namun metode CNN bekerja sangat baik untuk mengenali setiap notasi musik yang telah dipotong dengan benar. Hal ini ditunjukkan dari nilai akurasi yang mencapai 95,56%.

Kata Kunci: Optical Music Recognition, Deep Learning, Convolutional Neural Network, Notasi Musik, Segmentasi

Abstract

Optical Music Recognition (OMR) is a way to make recognition of musical notation automatically. The main problem of musical notation detection is how to detect a musical notation and how to recognize it. The recognizing musical notation can be used to reprocess into sound. In this study, the segmentation process is carried out to cut every notation. To recognize Musical notation is used Convolutional Neural Network (CNN). The CNN architecture uses a kernel 3x3. The layers of the feature learning process are 3 of convolutional layers and 3 of pooling layers. Filters used in the convolutional layer are 64,128, 256 and the number of neurons in the hidden layer is 7168. The experiments use two ways, the first, is testing the performance of CNN using the cut music notation data and the second is testing using a row of musical notations. The accuracy of a line of music notation recognition is not good, which is only 26.19%. Although the segmentation process is still not maximal in cutting every notation, the CNN method works very well to recognize each of cut musical notation. The accuracy reaches 95.56%.

Keywords: *Optical Music Recognition, Deep Learning, Convolutional Neural Network, Music Notation, Segmentation*

1. PENDAHULUAN

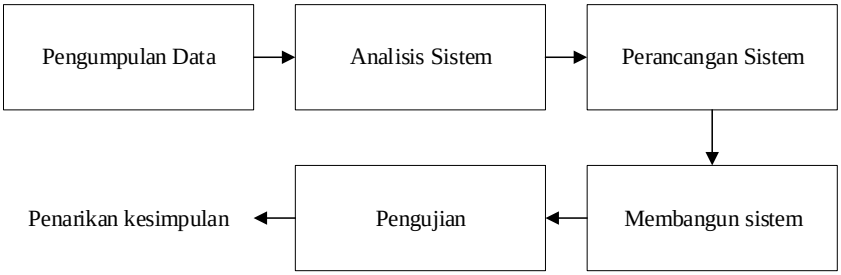
Optical Music Recognition (OMR) merupakan proses pengenalan notasi musik secara otomatis. Keberhasilan dalam *Optical Music Recognation* ditentukan dari seberapa baik sistem dapat mengenali notasi-notasi musik yang muncul dalam sebuah lembaran musik. Notasi dari nada, tanda diam, dinamika, tempo ataupun nada dasar adalah beberapa hal yang perlu dideteksi dalam pengenalan notasi musik. Beberapa penelitian telah dilakukan diantaranya adalah klasifikasi pada notasi musik dengan menggunakan model *Artificial Neural Network* (ANN) oleh Pascal Attwenger, mendapatkan hasil klasifikasi yang cukup baik. Pada penelitian tersebut mendapatkan hasil akurasi 97,02% untuk pengenalan tanda not dan 86.13% untuk pengenalan tanda istirahat [1].

Model ANN ini memiliki banyak lapisan yang disebut sebagai *Multilayer Perceptron* (MLP), dimana menurut Samuel Sena MLP ini memiliki kelemahan ketika data masukan berupa citra [2], untuk mengoptimalkan kinerja maka citra harus dilakukan *preprocessing*, segmentasi, dan diekstrak. Pengembangan lain yang dapat menyelesaikan permasalahan MLP ini yaitu dengan menggunakan model *Convolutional Neural Network* (CNN). CNN ini merupakan pengembangan dari ANN dan merupakan salah satu metode *Deep Learning* (DL) dimana dapat digunakan untuk mendeteksi atau mengenali suatu objek pada sebuah citra digital. DL masih merupakan bagian kecil dari *Machine Learning* (ML). DL pada dasarnya penerapan konsep dasar dari ML yang menerapkan model ANN dengan lapisan layer yang lebih banyak.

Peneliti sebelumnya yang telah melakukan penelitian klasifikasi pada citra dengan menggunakan metode CNN memiliki akurasi yang cukup baik. CNN pada kasus tulisan aksara sunda memiliki akurasi sebesar 62% [3], pada kasus klasifikasi citra tomat memiliki akurasi sebesar 90% [4], pada kasus klasifikasi citra wayang golek memiliki akurasi sebesar 90% [5] dan pada kasus pengenalan notasi musik dengan arsitektur 2 *convolution* dan *pooling* mendapatkan hasil yang konsisten untuk not *eighteenth* dan *sixteenth* tetapi masih tidak konsisten untuk not *quarter* dan *half* [6] serta kasus yang sama dengan menggunakan arsitektur ResNet-v2 mendapatkan akurasi sebesar 98% [7]. Berdasarkan latar belakang di atas, dalam penelitian ini akan dilakukan klasifikasi notasi musik dengan menerapkan model arsitektur CNN yang telah disesuaikan untuk dapat mengukur nilai akurasi dari model tersebut.

2. METODE PENELITIAN

Jenis pendekatan yang digunakan pada penelitian ini yaitu penelitian kualitatif. Tujuan dari penelitian kualitatif adalah eksplorasi pada fokus permasalahan penelitian tersebut. Salah satu metode penelitian yang cocok digunakan pada penelitian kualitatif adalah metode penelitian eksperimen. Langkah-langkah tahapan yang dilakukan pada penelitian ini dapat dilihat pada Gambar 1.



Gambar 1 Tahapan penelitian

Berikut ini penjelasan dari tahapan penelitian yang dilakukan pada penelitian:

1. Pengumpulan data pada penelitian ini menggunakan metode studi literatur seperti buku, jurnal, artikel ilmiah dan *website* yang terkait dengan penelitian.
2. Analisis sistem pada penelitian ini melakukan analisis terhadap metode yang digunakan dalam pembangunan perangkat lunak.

3. Perancangan sistem pada penelitian ini melakukan analisis terhadap keperluan-keperluan untuk membangun mesin pengklasifikasi pada notasi musik. Keperluan tersebut berupa konsep dan analisis lainnya yang berkaitan dengan model CNN.
4. Membangun sistem pada penelitian ini melakukan implementasi terhadap perancangan yang telah dibuat.
5. Pengujian pada penelitian ini melakukan pengujian terhadap sistem yang dibuat untuk mendapatkan hasil akurasi dalam pengenalan notasi musik.
6. Penarikan kesimpulan pada penelitian ini untuk mengetahui kesimpulan dari hasil pengujian pengenalan notasi musik dengan menerapkan metode *convolutional neural network*.

2.1. Data Masukan

Data masukan yang digunakan pada penelitian ini terdiri dari dua jenis yaitu data *training* dan data *testing.* Berikut data masukan yang akan digunakan :

a. Data *training* yang digunakan berasal dari penelitian sebelumnya yang dilakukan oleh Pascal Attwenger dari *University of Vienna* dan penambahan data yang diperlukan. Data *training* memiliki ukuran tinggi 50 *pixel* x lebar 30 *pixel*. Ekstensi data *training* yang digunakan berformat PNG (*Portable Network Graphics*). Data *training* disimpan pada folder-folder yang akan menjadi kelasnya.
b. Data *testing* yang digunakan yaitu satu baris notasi musik berisi partitur lagu. Data *testing* memiliki ekstensi berformat PNG. Data *testing* memiliki ukuran 2250 *pixel* x 200 *pixel*.
c. Banyak data *training* yang digunakan 4106 gambar, dibagi untuk *validation* sebesar 20% yaitu sebanyak 822 gambar. Data yang tidak berwarna atau hitam putih.
d. Target kelas yang digunakan sebanyak 90 kelas yaitu nada notasi penuh, 1/2, 1/4, 1/8, 1/16 dan notasi yang dikenal a, h, c1, d1, e1, f1, g1, a1, h1, c2, d2, e2, f2, g2, a2, h2, c3 serta tanda istirahat penuh, 1/2, 1/4, 1/8, 1/16.

2.2. *Preprocessing*

Preprocessing merupakan proses tahap awal yang dilakukan sebelum melakukan proses metode CNN. Ada beberapa proses yang dilakukan pada tahap *preprocessing* seperti: *grayscale*, *invert image* dan segmentasi.

2.2.1. *Grayscale*

Proses *grayscale* memiliki tujuan untuk menangani proses gradasi warna hitam dan putih. Hasil dari proses tersebut akan mengubah warna (RGB) menjadi warna abu-abu [8]. Melakukan proses *grayscale* berfungsi untuk mempersempit jarak warna dari 0 sampai 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih. Adapun persamaan yang digunakan dapat dilihat pada persamaan (1).

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

Keterangan:
I = Fungsi untuk mencari nilai skala keabu-abuan
R = Nilai merah (Red) dari suatu titik pixel
G = Nilai hijau (Green) dari suatu titik pixel
B = Nilai biru (Blue) dari suatu titik pixel

Pada penelitian ini hasilnya proses *grayscale* tidak terlalu terlihat, karena *input* gambar *dataset* telah berupa hitam putih.

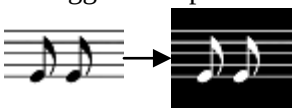
2.2.2. *Invert Color*

Proses *invert color* yaitu proses pengolahan citra yang memberikan hasil keluaran berupa kebalikan dari citra *input*. Adapun persamaan yang digunakan dalam proses *invert color* dapat dilihat pada persamaan (2).

$$o(x, y) = 255 - u(x, y) \quad (2)$$

Keterangan:
o(x,y) = Fungsi untuk mencari nilai *invert color*
u(x,y) = Nilai yang diambil dari citra *input*

Berikut ini hasil *invert color* menggunakan persamaan (2) dapat dilihat pada Gambar 2.



Input Output

Gambar 2 Hasil proses *invert color*

2.2.3. Segmentasi

Segmentasi citra yaitu suatu teknik untuk membagi beberapa daerah dimana setiap daerah memiliki kemiripan atribut [9]. Beberapa tahapan yang digunakan pada penelitian ini seperti: *threshold*, *morphology*, deteksi kontur dan *resize*.

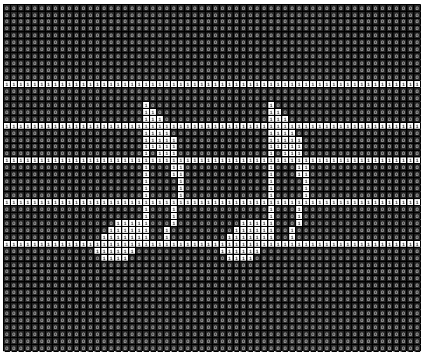
2.2.3.1. *Threshold*

Threshold merupakan segmentasi sederhana yang menggunakan ambang intensitas. Pada nilai *pixel* lebih kecil daripada nilai ambang batas termasuk sebagai area pertama dan nilai *pixel* yang lebih besar atau sama dengan nilai ambang batas dikelompokkan sebagai area kedua, salah satu pada area tersebut berperan sebagai latar belakang [8]. Adapun persamaan ambang batas yang digunakan pada penelitian ini dapat dilihat pada persamaan (3). Pengambilan ambang batas sebesar 127 karena nilai pixel dari 0-255 maka akan diambil posisi tengah pixel yaitu 127.

$$p(x,y) = f(x) = \begin{cases} 1, & untuk\ (x,y) \geq 127 \\ 0, & untuk\ (x,y) < 127 \end{cases} \quad (3)$$

Keterangan:
p(x,y) = Fungsi untuk mencari nilai *threshold*
(x,y) = Nilai yang diambil dari citra *input* pada posisi x dan y

Sebelum melakukan segmentasi fungsi *threshold* dilakukan pula fungsi *grayscale* dan *invert color*, data masukan proses *threshold* dapat dilihat pada *output* Gambar . Berikut ini hasil proses *threshold* dapat dilihat pada Gambar , dimana pada bagian berwarna hitam memiliki nilai *pixel* 0 dan bagian berwarna putih memiliki nilai *pixel* 1.



Gambar 3 Hasil proses *threshold*

2.2.3.2. *Morphology*

Morphology merupakan suatu teknik pengolahan citra yang berdasarkan bentuk segmen citra. Untuk melakukan operasi morfologi citra perlu dilakukan proses *thresholding* terlebih dahulu dan dibutuhkan dua data masukan yaitu citra masukan dan kernel atau dalam *opencv* memakai nama *structuring element*. Beberapa fungsi morfologi yang digunakan pada penelitian ini yaitu operasi erosi dan operasi dilasi.

A. Operasi Erosi

Operasi erosi mempunyai efek memperkecil struktur citra [8]. Operasi ini dapat dirumuskan seperti persamaan (*4*).

$$A \ominus B = \{z|(B)_z \subseteq A\} \quad (4)$$

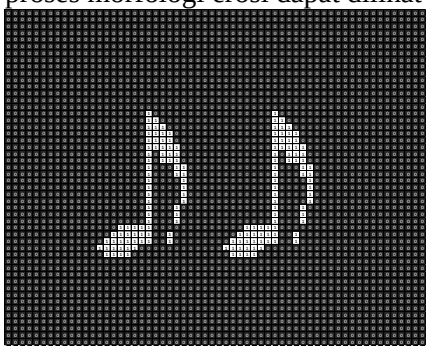
Keterangan:
A = Matriks citra data masukan
B = Matriks kernel

Untuk ukuran kernel yang digunakan pada aturan *morphology* tidak ditentukan secara spesifik, namun pada penelitian ini dengan tujuan untuk menghilangkan garis paranada maka digunkan ukuran 2x1. Kernel berisi nilai 1 dimana nilai ini akan berfungsi untuk gambar *input* yang bernilai 1 pada proses erosi dan dilasi. Pada proses erosi apabila saat kernel berjalan ke seluruh gambar *input* nilai *input* 1 hanya memenuhi 1 posisi kernel tidak keduanya maka nilai *input* tersebut akan dilakukan perubahan nilai menjadi 0. Untuk proses dilasi proses yang dilakukan sama seperti erosi namun kebalikannya dimana akan dilakukan perubahan nilai *input* menjadi 1.

1
1

Gambar 4 Ukuran kernel

Proses erosi memiliki tujuan yaitu untuk menghilangkan garis paranada pada Gambar , namun hasil erosi menyebabkan bentuk notasi menjadi tipis maka akan dilanjutkan dengan proses dilasi. Berikut ini hasil proses morfologi erosi dapat dilihat pada Gambar 5.



Gambar 5 Hasil proses erosi

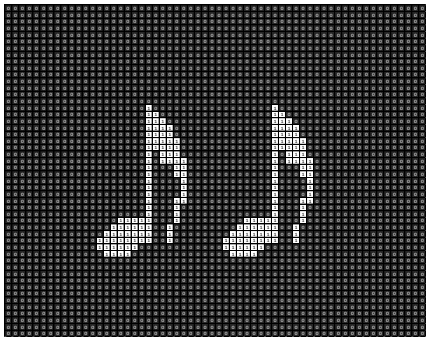
B. Operasi Dilasi

Proses dilasi dapat dipakai untuk mendapatkan efek pelebaran terhadap *pixel* yang bernilai 1. Operasi ini dapat dirumuskan seperti persamaan (5).

$$A \oplus B = \left\{z|\left[(\hat{B})_z \cap A\right] \subseteq A\right\} \quad (5)$$

Keterangan:
A = Matriks citra data masukan
B = Matriks kernel

Ukuran kernel yang digunakan untuk proses dilasi dapat dilihat pada Gambar dan data masukan dapat dilihat pada Gambar . Proses dilasi memiliki tujuan yaitu untuk mempertebal bentuk notasi yang disebabkan proses sebelumnya yaitu erosi agar gambar notasi kembali seperti semula. Berikut hasil proses morfologi dilasi dapat dilihat pada Gambar .



Gambar 6 Hasil proses dilasi

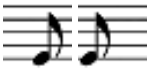
2.2.3.3. Deteksi Kontur

Deteksi kontur merupakan mendeteksi kontur terluar pada objek yang memiliki warna atau intensitas yang sama. Pada penelitian ini menggunakan *library* opencv, dimana fungsi *findContours* hanya akan mendeteksi bagian tepi objek pada opencv memiliki fungsi *drawContours* yang akan memberikan nilai pada bagian dalam objek yang telah dideteksi *findContours.*

Gambar 7 Citra masukan deteksi kontur

Gambar 8 Hasil posisi titik pemotongan citra

Setelah mengetahui titik citra yang akan dipotong maka dilakukan pemotongan terhadap data masukan awal yang masih memiliki garis paranada. Berikut ini hasil akhir pemotongan citra dapat dilihat pada Gambar .



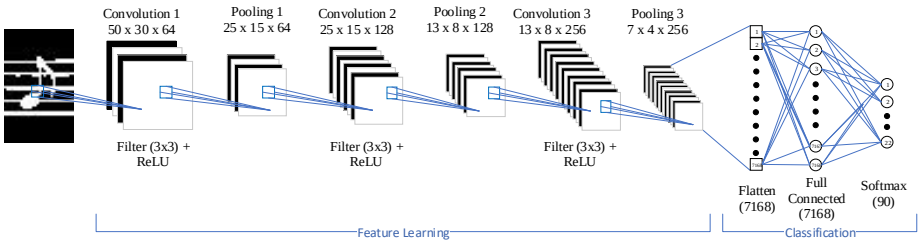
Gambar 9 Hasil pemotongan citra masukan

2.2.3.4. *Resize*

Proses *resize* dilakukan karena setelah proses pemotongan citra masih memiliki ukuran yang besar, maka dilakukan proses *resize* dimana bertujuan untuk menyesuaikan ukuran citra terhadap ukuran data masukan sistem. Setelah dilakukan proses resize maka ukuran citra yang didapatkan sebesar tinggi 50-*pixel* dan lebar 30-*pixel*.

2.3. *Convolutional Neural Network*

Arsitektur CNN terbagi menjadi dua bagian penting yaitu *feature learning* dan *classification*. Pada bagian *feature learning* terdiri dari dua tahapan yaitu *convolutional layer* dan *pooling layer*. Pada penelitian ini *convolutional layer* yang digunakan kernel berukuran 3x3, filter yang digunakan 64, 128 dan 256. Pada proses lainnya *pooling layer* menggunakan kernel 3x3. Pada bagian lainnya yaitu bagian *classification*, akan melakukan tahapan *fully-connected layer*. Pada penelitian ini jumlah neuron pada flatten layer bernilai 7168 dari hasil semua nilai pixel yang ada pada proses sebelumnya, jumlah neuron pada *hidden layer* yang digunakan sebesar 7168 dan target kelas sebanyak 90. Berikut ini arsitektur CNN yang digunakan pada penelitian ini dapat dilihat pada Gambar .



Gambar 10 Arsitektur CNN

Pada penelitian ini inisialisasi bobot menggunakan metode *glorot uniform* dan inisialisasi bias dengan angka 0. Berikut ini persamaan yang digunakan dapat dilihat pada persamaan (6).

$$w = \left[-\frac{\sqrt{6}}{\sqrt{hi * wi * fi + ho * wo * fo}}, \frac{\sqrt{6}}{\sqrt{hi * wi * fi + ho * wo * fo}}\right] \quad (6)$$

Keterangan:

hi = tinggi masukan
ho = tinggi keluaran
wi = lebar masukan
wo = lebar keluaran
fi = filter masukan
fo = filter keluaran

2.3.1. Convolution Layer

Convolution layer adalah bagian tahap awal setelah *input layer* pada arsitektur CNN. Tahap ini melakukan proses konvolusi pada *output* dari layer sebelumnya dan proses utama yang mendasari dari arsitektur CNN. Pada tahap ini data *input* akan dilakukan *dot product* dengan kernel dimana pada penelitian ini kernel berukuran 3x3. Kernel tersebut akan bergerak ke seluruh data *input*, dimana data *input* berukuran lebar 30 *pixel* x tinggi 50 *pixel*. Sebelum dilakukan *dot product* pada penelitian ini telah diatur data *input* akan ditambah *zero padding* sebanyak 1 agar informasi pada data *input* tidak berkurang dan pergeseran kernel atau *stride* sebanyak 1 kali. Hasil keluaran dari proses konvolusi dapat di hitung dengan menggunakan persamaan (7).

$$(W - F + 2P)/S + 1 \quad (7)$$

Keterangan:

W = Ukuran volume gambar
F = Ukuran filter
P = Nilai padding yang digunakan
S = Ukuran pergeseran (Stride)

Persamaan yang digunakan untuk melakukan proses konvolusi dapat dilihat pada persamaan (8).

$$s(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \quad (8)$$

Keterangan:

s(i,j) = Posisi feature map yang diisi hasil dot product
I = Matrix input
K = Matrix kernel
m,n = Jumlah ukuran kernel

Setelah proses konvolusi selesai *feature map* akan dilakukan proses aktivasi *Rectified Linear Units* (ReLU) ke seluruh area *feature map* yang bertujuan agar nilai pada *feature map* tidak ada bernilai negatif. Berikut persamaan yang digunakan dapat dilihat pada persamaan (9).

$$f(x) = \max(x, 0) \quad (9)$$

Keterangan:

x = nilai pada *feature map*

2.3.2. Pooling Layer

Pooling layer adalah suatu cara untuk melakukan pengurangan ukuran *matrix*. *Pooling layer* terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan secara bergantian bergeser pada seluruh area *feature map*. Penelitian ini menggunakan fungsi *max-pooling* dimana akan dilakukan filter terhadap nilai tertinggi yang berada pada posisi filter. Filter kernel berukuran 3x3, penambahan *zero padding* pada *feature map* sebanyak 1 dan pergeseran atau *stride* filter kernel sebanyak 2 kali.

2.3.3. Fully Connected Layer

Fully connected layer bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. Data masukan untuk *fully connected layer* pada penelitian ini adalah *output* dari layer sebelumnya dimana masih berbentuk *matrix*. Sebelum masuk ke dalam *fully connected layer*, data masukan akan dilakukan *flatten* yaitu mengubah *matrix* menjadi vektor. Setelah data menjadi vektor, hasil tersebut akan dilakukan proses *fully connected layer* dengan persamaan yang dapat dilihat pada persamaan (10).

$$fc_k = b_k + \sum_{i=0}^{n} x_i w_{i,k} \ , k = 0,1,2,\cdots,t \tag{10}$$

Keterangan:

y = keluaran
x = nilai *flatten*
b = bias
w = bobot
n = banyaknya data pada *flatten*
t = banyaknya target pada layer FC

Setelah selesai melakukan proses *fully connected layer*, hasil keluaran akan dilakukan fungsi aktivasi ReLU agar hasil *fully connected layer* tidak ada yang bernilai negatif. Persamaan yang digunakan aktivasi ReLU dapat dilihat pada persamaan (9).

2.3.4. Softmax Classifier

Softmax Classifier atau layer prediksi merupakan algoritma *Logistic Regression* yang dapat digunakan untuk pengklasifikasian. Hasil dari layer sebelumnya yaitu *fully connected layer* akan dilakukan klasifikasi dengan menggunakan fungsi aktivasi *softmax*. Berikut ini persamaan aktivasi *softmax* dapat dilihat pada persamaan (11).

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} = \frac{e^{(w^T x)_j}}{\sum_{k=1}^{K} e^{(w^T x)_k}} \tag{11}$$

Keterangan:

f(z) = hasil fungsi
j = pengulangan sebanyak kelas
w^T = bobot yang telah dilakukan transpose
x = nilai pada hidden layer
K = banyaknya kelas

Hasil yang didapatkan berupa nilai prediksi probabilitas dan nilai y. Nilai prediksi probabilitas dimana yang mendekati angka 1 maka merupakan prediksi kelasnya. Nilai y dimana akan bernilai 1 untuk pediksi kelasnya dan 0 untuk kelas yang lainnya, nilai y akan digunakan untuk mencari *loss function*.

2.3.5. Cross Entropy Loss Function

Cross Entropy Loss Function adalah fungsi untuk mengetahui seberapa tinggi *error* atau *loss* yang dilakukan *machine learning* dengan tujuan mengukur seberapa bagus performa yang dihasilkan oleh model dalam melakukan prediksi terhadap target. Berikut ini persamaan yang dapat digunakan untuk menghitung *loss* dapat dilihat pada persamaan (12).

$$L = -\sum_{i=0}^{N} y_i \log(p_i) \tag{12}$$

Keterangan:

N = banyaknya kelas
y = nilai *output* (1 untuk prediksi kelasnya dan 0 untuk kelas lainnya)

p = nilai prediksi atau keluaran aktivasi *softmax*

2.3.6. Back-propagation

Back-propagation pada penelitian ini menggunakan metode *chain rule* dengan tujuan untuk memperbaiki bobot dan bias berdasarkan *error* yang didapat pada proses *pass-forward* dengan cara menghitung gradien pada setiap parameternya. Berikut ini tahapan beserta persamaan yang digunakan untuk melakukan proses *back-propagation:*

A. Menghitung *gradient* pada bobot *fully-connected layer*.

$$\frac{\partial L}{\partial w_{i,j}} = \frac{\partial L}{\partial fcO_i} * \frac{\partial fcO_i}{\partial fcI_i} * \frac{\partial fcI_i}{\partial w_{i,j}} \tag{13}$$

B. Menghitung turunan dari fungsi cross entropy.

$$\frac{\partial L}{\partial fcO_i} = \left(1\{y = y_i\} - \frac{e^{fcI_i}}{\sum_{j=0}^{t} e^{fcI_j}}\right) fcO_i, \qquad i = 0,1,2,\dots,t \tag{14}$$

C. Menghitung turunan *output* dari *fully-connected layer* setelah melalui fungsi aktivasi *softmax*.

$$\frac{\partial fcO_i}{\partial fcI_i} = \frac{e^{fcI_i} * (\sum_{n \neq i}^{t} e^{fcI_n})}{\sum_{n=0}^{t} {e^{fcI_n}}^2}, i = 0,1,2,\dots,t \tag{15}$$

D. Menghitung turunan bobot dari *fully-connected layer*.

$$\frac{\partial fcI_i}{\partial w_{i,j}} = OutputLayerSebelumnya_j, \qquad i = 0,1,2,\dots,t \tag{16}$$

Keterangan:
fcI_i = Masukan *fully-connected layer* ke-i
fcO_i = Keluaran *fully-connected layer* ke-i
L = fungsi loss (*cross entropy*)
w = bobot
t = jumlah neuron
j = 0,1,2, ... jumlah neuron pada layer sebelumnya
b = bias

Hasil yang didapatkan untuk mendapatkan nilai gradien pada setiap bobotnya agar dapat digunakan pada proses *adam optimizer*. Proses perhitungan gradien akan terus diulang sampai sebanyak jumlah neuron yang ada.

2.3.7. Adam Optimizer

Adam optimizer merupakan suatu cara untuk mengoptimasi suatu parameter, optimasi tersebut dapat membuat parameter menjadi maksimum atau minimum [10]. Ada beberapa parameter yang perlu diinisialisasi dari awal yaitu:

$m_0 = 0$ $\quad v_0 = 0$ $\quad t = 0$
$\alpha = 0.001$ $\quad \beta_1 = 0.9$ $\quad \beta_2 = 0.999$
$\epsilon = 10^{-8}$

Berikut ini tahapan serta persamaan yang digunakan untuk melakukan optimasi yaitu:

A. Menambah t pada setiap iterasi

$$t = t + 1 \tag{17}$$

B. Mengambil nilai bobot

$$\theta_{t-1} = bobot\ yang\ akan\ diupdate \tag{18}$$

C. Mengambil nilai gradien

$$g_t = nilai\ gradien\ yang\ didapatkan \tag{19}$$

D. Memperbaharui bias momen pertama

$$m_t = \beta_1 . m_{t-1} + (1 - \beta_1) . g_t \tag{20}$$

E. Memperbaharui bias momen kedua

$$v_t = \beta_2 . v_{t-1} + (1 - \beta_2) . g_t^2 \tag{21}$$

F. Menghitung koreksi bias momen pertama

$$\widehat{m_t} = \frac{m_t}{(1 - \beta_1^t)} \tag{22}$$

G. Menghitung koreksi bias momen kedua

$$\widehat{v_t} = \frac{v_t}{(1 - \beta_2^t)} \tag{23}$$

H. Memperbaharui parameter

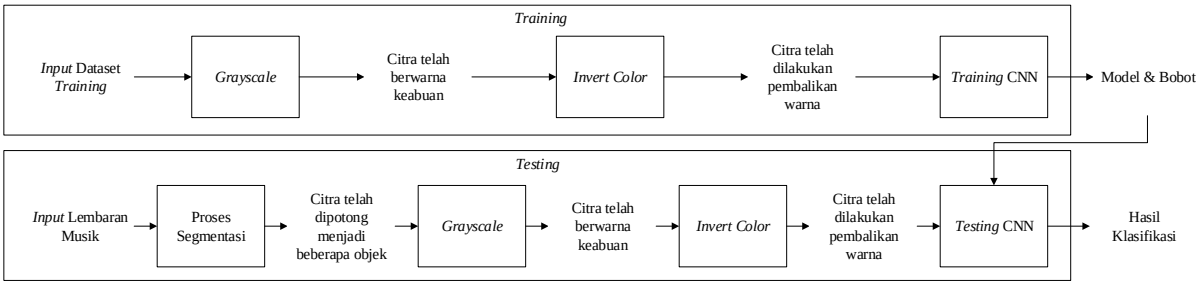
$$\theta_t = \theta_{t-1} - \alpha . \widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon) \tag{24}$$

Proses memperbaharui bobot akan terus diulang sebanyak jumlah neuron yang ada. Setelah *update* bobot selesai bobot dan model arsitektur akan disimpan secara eksternal untuk proses *testing* gambar agar tidak diperlukan lagi proses *training* untuk melakukan prediksi pada gambar baru. Untuk proses pengujian hanya berjalan sampai proses *feed-forward* pada CNN.

3. HASIL DAN PEMBAHASAN

Pada penelitian sebelumnya yang dilakukan oleh Pascal Attwenger, penggunaan ANN untuk deteksi nada dan tanda diam berhasil dilakukan dan mendapatkan hasil yang cukup baik. Namun masalah yang dapat muncul dari penelitian tersebut bukan terdapat pada hasil akhir yang bagus, akan tetapi pada MLP yang ada di dalam metode ANN. MLP memiliki kelemahan apabila yang diproses berupa data citra. Data set yang digunakan diambil dari data MNIST (*Modified National Institute of Standards and Technology*) yang memiliki nilai dari 0-255 pada setiap *pixel* yang ada. Penggunaan MLP untuk melakukan klasifikasi semua digit akan mendapatkan hasil yang baik karena sebagian besar data MNIST objek terletak pada posisi tengah-tengah gambar. Lalu apabila objek gambar yang akan dikenali tidak berada pada posisi tengah-tengah maka gambar tersebut tidak akan dikenali [2]. Solusi yang ditawarkan pada penelitian ini yaitu menggunakan metode CNN. Banyak kelas target pada penelitian ini sebanyak 90 kelas yaitu nada notasi yang dikenal penuh, 1/2, 1/4, 1/8, 1/16 dan notasi yang dikenal a, h, c1, d1, e1, f1, g1, a1, h1, c2, d2, e2, f2, g2, a2, h2, c3 serta tanda istirahat penuh, 1/2, 1/4, 1/8, 1/16.

Tahapan yang digunakan untuk pengenalan citra notasi musik di dalam penelitian ini diberikan pada Gambar . Ada dua tahapan yang dilakukan yaitu *training* dan *testing*. Tahapan pertama pada proses *training* yaitu *input dataset training* akan melalui proses *grayscale* terlebih dahulu. Keluaran proses *grayscale* akan masuk ke proses *invert color*. Keluaran proses *invert color* inilah yang akan masuk ke dalam proses *training* menggunakan metode CNN. Proses *testing* dapat dilakukan apabila proses *training* telah selesai. Tahapan pertama pada proses *testing* yaitu *input* data *testing*. Data tersebut berupa satu baris notasi musik berupa partitur musik. Selanjutnya akan dilakukan proses segmentasi dengan tujuan mengubah partitur lagu menjadi *dataset testing.* Setelah mendapatkan *dataset testing* akan dilakukan proses *grayscale* dan proses *invert color*. Proses selanjutnya akan dilakukan proses *testing* menggunakan CNN. Pada proses tersebut hasil dari proses *training* akan digunakan untuk melakukan pengenalan terhadap data *testing*. Hasil dari proses ini berupa hasil klasifikasi terhadap data *testing* tersebut.



Gambar 11 Tahapan Sistem

3.1. Pengujian Akurasi

Pengujian akurasi dilakukan untuk mendapatkan akurasi dari pengujian prediksi data baru. Sebelumnya akan dilakukan pengujian terhadap arsitektur yang akan digunakan. Pengujian ini meliputi pengujian ukuran kernel yang akan digunakan, pengujian jumlah layer dan neuron yang akan dipilih dan penentuan nilai *leaning rate* dan *epoch* yang akan digunakan dalam pengujian akurasi. Pengujian arsitektur dapat dilihat pada hasil *validation accuracy* dan *validation loss* yang didapatkan. Semakin besar *validation accuracy* yang didapatkan semakin baik sedangkan semakin kecil *validation loss* yang didapatkan sangat baik.

3.1.1. Pengujian Ukuran Kernel

Pengujian ukuran kernel dilakukan untuk mendapatkan ukuran kernel yang sesuai untuk kasus penelitian ini. Berikut ini hasil dari pengujian ukuran kernel dapat dilihat pada Tabel 1.

Tabel 1 Hasil pengujian ukuran kernel

Ukuran Kernel	***Validation Accuracy***	***Validation Loss***
3x3	0.86	0.54
5x5	0.84	0.62

Berdasarkan Tabel 1 menunjukkan bahwa kernel dengan ukuran 3x3 lebih baik dengan mendapatkan akurasi sebesar 86% dan loss yang paling kecil sebesar 0.54. Ukuran kernel terbaik akan dipakai untuk pengujian selanjutnya.

3.1.2. Pengujian Jumlah Layer pada CNN

Pengujian jumlah layer pada CNN dilakukan untuk mendapatkan jumlah layer yang sesuai untuk kasus penelitian ini. Berikut ini hasil dari pengujian jumlah layer dapat dilihat pada Tabel 2.

Tabel 2 Hasil pengujian jumlah layer pada CNN

Banyaknya Layer *Conv.* dan *Pooling*	***Validation Accuracy***	***Validation Loss***
3	0.91	0.56
4	0.85	0.56

Berdasarkan Tabel 2 menunjukkan bahwa jumlah layer dengan jumlah 3 lebih baik dengan mendapatkan akurasi sebesar 91%. Jumlah layer terbaik akan dipakai untuk pengujian selanjutnya.

3.1.3. Pengujian Jumlah Neuron pada *Hidden Layer*

Pengujian jumlah neuron pada *hidden layer* dilakukan untuk mendapatkan jumlah neuron yang sesuai untuk kasus penelitian ini. Berikut ini hasil dari pengujian jumlah neuron dapat dilihat pada Tabel 3.

Tabel 3 Hasil pengujian jumlah neuron pada *hidden layer*

Jumlah Neuron	***Validation Accuracy***	***Validation Loss***
5120	0.87	0.66
7168	0.91	0.56
9216	0.89	0.57

Berdasarkan Tabel 3 menunjukkan bahwa jumlah neuron dengan jumlah 7168 lebih baik dengan mendapatkan akurasi sebesar 91% dan *loss* yang paling kecil sebesar 0.56. Jumlah neuron terbaik akan dipakai untuk pengujian selanjutnya.

3.1.4. Pengujian *Learning Rate* dan *Epoch*

Pengujian *learning rate* dan *epoch* dilakukan untuk mendapatkan nilai *learning rate* yang terbaik untuk kasus penelitian ini. Berikut ini hasil dari pengujian *learning rate* dan *epoch* dapat dilihat pada Tabel 4.

Tabel 4 Hasil pengujian *learning rate* dan *epoch*

No.	***Lr***	***Jumlah Epoch***	**Berhenti *Epoch***	***Validation Accurcy***	***Validation Loss***
1	0.0005	30	15	0.88	0.57
2	0.0006	30	15	0.89	0.56
3	0.0007	30	17	0.91	0.47
4	0.0008	30	15	0.9	0.51
5	0.0009	30	15	0.89	0.51
6	0.001	30	15	0.9	0.53
7	0.0011	30	14	0.89	0.6
8	0.0012	30	11	0.89	0.56
9	0.0013	30	11	0.89	0.55
10	0.0014	30	11	0.87	0.59
11	0.0015	30	16	0.87	0.65

Berdasarkan Tabel 4 menunjukkan bahwa *learning rate* dengan sebesar 0.0007 mendapatkan akurasi lebih baik dari yang lain sebesar 91% dan *loss* yang paling kecil dari yang lain sebesar 0.47. *Learning rate* terbaik akan dipakai untuk pengujian prediksi data baru.

3.1.5. Pengujian Prediksi Data Baru

Pengujian prediksi data baru dilakukan untuk mendapatkan seberapa baik akurasi arsitektur yang telah dibuat dengan melakukan prediksi data baru. Dari jumlah gambar yang diujikan sebanyak 180 gambar didapatkan sebanyak 172 gambar yang dapat diprediksi dengan benar. Dengan menggunakan persamaan (25) akan didapatkan akurasi pada penelitian ini.

$$\frac{Jumlah\ data\ benar}{Jumlah\ data\ test} * 100\% \tag{25}$$

$$\frac{172}{180} * 100\% = 95.56\%$$

3.1.6. Pengujian pada Baris Notasi Musik

Pengujian baris notasi musik dilakukan untuk mendapatkan seberapa baik akurasi arsitektur yang telah dibuat terhadap pengujian baris notasi musik. Dari 6 baris notasi musik yang diujikan dengan jumlah notasi yang akan diprediksi sebanyak 84 notasi, didapatkan jumlah prediksi benar sebanyak 22 notasi. Dengan menggunakan persamaan (25) maka didapatkan akurasi pada penelitian ini sebesar 26.19%.

4. KESIMPULAN

Berdasarkan dari penelitian yang telah dilakukan bahwa *optical music recognition* menggunakan *convolutional neural network* dengan melakukan pengujian prediksi data baru mendapatkan hasil akurasi sebesar 95.56% dan pengujian terhadap baris notasi musik dengan akurasi sebesar 26.19%. Dengan akurasi yang didapatkan cukup tinggi maka dapat disimpulkan metode *convolutional neural network* berhasil diimplementasikan dengan baik untuk *optical music recognition* namun masih kurang baik pada pengenalan baris notasi musik.

5. SARAN

Penelitian *optical music recognition* pada citra notasi musik menggunakan *convolutional neural network* masih jauh dari kata sempurna, untuk itu peneliti memberikan saran dari penelitian ini diantaranya adalah :

- Menambah kelas baru, agar sistem dapat mengenal lebih banyak objek.
- Membandingkan dengan menggunakan arsitektur *convolutional neural network* lainnya untuk dapat mengetahui akurasi pada *optical music recognition* pada citra lembaran musik.
- Memperbaiki *preprocessing* pada pemotongan baris notasi musik dan dapat berurutan.

DAFTAR PUSTAKA

[1] Attwenger, P., 2015, Recognizing Musical Notation Using Artificial Neural Networks, *Thesis*, Computer Science, University of Vienna, Vienna.
[2] Samuel, S., 2017, Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN), *https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94*, diakses tgl 16 November 2018.
[3] Ramadhan, I., 2018, Pengenalan Pola Citra Tulisan Tangan Aksara Sunda Dengan Metode Convolutional Neural Network, *Skripsi*, Teknik dan Ilmu Komputer, Universitas Komputer Indonesia, Bandung.
[4] Shafira, T., 2018, Implementasi Convolutional Neural Networks Untuk Klasifikasi Citra Tomat Menggunakan Keras, *Skripsi*, Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia, Yogyakarta.
[5] Nurhikmat, T., 2018, Implementasi Deep Learning Untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) Pada Citra Wayang Golek, *Skripsi*, Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia, Yogyakarta.
[6] Blanes, A. R. dan Bisquerra, A. F., 2017, Camera-based Optical Music Recognition using a Convolutional Neural Network, *2017 14th IAPR Int. Conf. Doc. Anal. Recognit.*, vol. 2, hal. 27–28.
[7] Pacha, A. and Calvo-Zaragoza, J., 2018, Optical Music Recognition in Mensural Notation with Region-Based Convolutional Neural Networks, *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, September 23.
[8] Kadir, A. dan Susanto, A., 2013, *Teori dan Aplikasi Pengolahan Citra*, ANDI, Yogyakarta.
[9] Putra, D., 2010, *Pengolahan Citra Digital*, ANDI, Yogyakarta.
[10] Kingma, D. P. dan Ba, J. L., 2017, ADAM: A Method For Stochastic Optimization, *arXiv preprint arXiv:1412.6980*, vol.9, hal.1–15.