Jurnal Teknik Informatika (JUTIF)
Vol. 4, No. 5, October 2023, pp. 1143-1150

DOI: https://doi.org/10.52436/1.jutif.2023.4.5.993
p-ISSN: 2723-3863
e-ISSN: 2723-3871

RECOGNITION OF REAL-TIME HANDWRITTEN CHARACTERS USING CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE

Muhammad Satrio Gumilang[1], Donny Avianto[*2]

[1,2]Informatics, Faculty of Science & Technology, Universitas Teknologi Yogyakarta, Indonesia
Email: [1]muhammad.5200411155@student.uty.ac.id, [2]donny@uty.ac.id

(Article received: April 11, 2023; Revision: May 17, 2023; published: October 15, 2023)

Abstract

Pattern recognition, including handwriting recognition, has become increasingly common in everyday life, as is recognizing important files, agreements or contracts that use handwriting. In handwriting recognition, there are two types of methods commonly used, namely online and offline recognition. In online recognition, handwriting patterns are associated with pattern recognition to generate and select distinctive patterns. In handwritten letter patterns, machine learning (deep learning) is used to classify patterns in a data set. One of the popular and accurate deep learning models in image classification is the convolutional neural network (CNN). In this study, CNN will be implemented together with the OpenCV library to detect and recognize handwritten letters in real-time. Data on handwritten alphabet letters were obtained from the handwriting of 20 students with a total of 1,040 images, consisting of 520 uppercase (A-Z) images and 520 lowercase (a-z) images. The data is divided into 90% for training and 10% for testing. Through experimentation, it was found that the best CNN architecture has 5 layers with features (32, 32, 64, 64, 128), uses the Adam optimizer, and conducts training with a batch size of 20 and 100 epochs. The evaluation results show that the training accuracy is between 85, 90% to 89.83% and testing accuracy between 84.00% to 87.00%, with training and testing losses ranging from 0.322 to 0.499. This research produces the best CNN architecture with training and testing accuracy obtained from testing. The developed CNN model can be used as a reference or basis for the development of more complex handwriting pattern recognition models or for pattern recognition in other domains, such as object recognition in computer vision, facial recognition, and other object detection.

Keywords: *Convolutional Neural Network, Handwriting Patterns, OpenCV.*

PENGENALAN KARAKTER TULISAN TANGAN SECARA *REAL-TIME* MENGGUNAKAN ARSITEKTUR CONVOLUTIONAL NEURAL NETWORK

Abstrak

Pengenalan pola, termasuk pengenalan tulisan tangan telah menjadi semakin umum dalam kehidupan sehari-hari, seperti mengenali berkas penting, perjanjian atau kontrak yang menggunakan tulisan tangan. Dalam pengenalan tulisan tangan, terdapat dua jenis metode yang umum digunakan, yaitu pengenalan secara online dan offline. Dalam pengenalan secara online, pola tulisan tangan dikaitkan dengan pengenalan pola untuk menghasilkan dan memilih pola yang khas. Dalam pola huruf tulisan tangan, pembelajaran mesin (*deep learning*) digunakan untuk mengklasifikasikan pola dalam kumpulan data. Salah satu model *deep learning* yang populer dan akurat dalam klasifikasi citra adalah *convolutional neural network* (CNN). Pada penelitian ini, CNN diimplementasikan bersama dengan *library OpenCV* untuk melakukan pendeteksian dan pengenalan huruf tulisan tangan secara *real-time*. Data huruf alfabet tulisan tangan diperoleh dari tulisan tangan 20 mahasiswa dengan total 1.040 citra, yang terdiri dari 520 citra huruf kapital (A-Z) dan 520 citra huruf kecil (a-z). Data tersebut dibagi menjadi 90% untuk *training* dan 10% untuk *testing*. Melalui eksperimen, ditemukan bahwa arsitektur CNN terbaik memiliki 5 *layer* dengan fitur (32, 32, 64, 64, 128), menggunakan *optimizer* Adam, dan melakukan pelatihan dengan *batch size* sebesar 20 dan *epoch* sebanyak 100. Hasil evaluasi menunjukkan akurasi pelatihan antara 85,90% hingga 89,83% dan akurasi *testing* antara 84,00% hingga 87,00%, dengan *loss* pelatihan dan *testing* yang berada dalam rentang 0,322 hingga 0,499. Penelitian ini menghasilkan arsitektur CNN terbaik dengan akurasi pelatihan dan *testing* yang diperoleh dari pengujian. Model CNN yang dikembangkan dapat digunakan sebagai referensi atau dasar untuk pengembangan model pengenalan pola tulisan tangan yang lebih kompleks atau untuk pengenalan pola pada domain lain, seperti pengenalan objek dalam penglihatan komputer, pengenalan wajah, dan deteksi objek lainnya.

Kata kunci: *Convolutional Neural Network, OpenCV, Pola Tulisan Tangan.*

1. PENDAHULUAN

Pengenalan pola mencakup berbagai bidang seperti, pengenalan wajah, pengenalan sidik jari, pengenalan gambar, pengenalan karakter, pengenalan angka, dan sebagainya [1]. Pola tulisan tangan menjadi semakin umum dalam kehidupan, misalnya untuk mengenali berkas penting yang menggunakan tulisan tangan menjadi suatu tulisan digital yang bisa diakses secara mudah melalui perangkat elektronik. Dalam teknologi pengenalan tulisan tangan, terdapat dua jenis metode pengenalan yang umum digunakan, yaitu pengenalan tulisan tangan secara online dan pengenalan tulisan tangan secara offline [2]. Untuk memahami masalah tulisan tangan secara online erat kaitannya dengan pengenalan pola, yang bertujuan untuk menghasilkan dan memilih pola yang dapat digunakan dengan bukti yang khas.

Untuk mengenali huruf tulisan tangan seseorang, program komputer harus dilatih terlebih dahulu. Sehingga diperoleh informasi yang mewakili data yang diambil dari objek, yang tentunya harus berupa informasi numerik agar komputer dapat mengenalinya. Informasi numerik diperoleh dengan memindai gambar objek [3]. Dari hasil pemindaian diperoleh citra yang kemudian diproses terlebih dahulu untuk membentuk pola berupa data numerik yang dapat dibaca komputer.

Pola dalam tulisan tangan dapat dianalisis dengan menggunakan teknik pembelajaran mesin. Pembelajaran mesin merupakan suatu bidang yang baru digunakan untuk menganalisis pola dalam sekumpulan data dengan tujuan mendapatkan informasi khusus mengenai pola yang telah diklasifikasikan. Dengan semakin mendalamnya *pattern learning*, maka prosesnya menjadi *deep learning* atau sering disebut dengan deep learning [4]. Metode *deep learning* ini melakukan pembelajaran dengan menggunakan beberapa lapis jaringan syaraf tiruan untuk meneliti setiap fungsi, kemudian meneliti model lebih dalam untuk mendapatkan model yang lebih baik, seperti meniru kemampuan otak manusia untuk mengingat sesuatu. Salah satu model deep learning adalah *convolutional neural network*.

Convolutional neural network (CNN) merupakan salah satu model yang banyak digunakan dan dikenal sebagai salah satu model yang akurat dalam melakukan klasifikasi citra [5]. Hal ini dapat ditemukan dalam penelitian sebelumnya tentang pembuatan sistem pengenalan tulisan tangan yang memberikan bukti yang mendukung, seperti; penggunaan algoritma *convolutional neural network* (CNN) mendapatkan akurasi sebesar 92,91% [6], *Gray Level Co-occurrence Matrix* (GLCM) dengan pengklasifikasian *Euclidean Distance* 78% [7], Serta *Freeman Chain Code dan K-Nearest Neighbor* 72% [8].

Berikut hasil-hasil penelitian sebelumnya dengan bidang dan tema yang sama seperti penelitian yang sedang dilakukan.

Menurut penelitian [9] pengenalan tulisan tangan huruf Hiragana dalam bahasa Jepang dengan menggunakan metode *Convolutional Neural Network* (CNN) menggunakan arsitektur VGG-16. Dalam penelitian ini, terdapat total 1380 contoh huruf yang terdiri dari 30 contoh per jenis huruf. Terdapat tiga skenario yang digunakan, yaitu menggunakan *optimizer* Adam dengan rentang *learning rate* antara 0,0001 hingga 0,1, *optimizer* SGD dengan rentang *learning rate* antara 0,0001 hingga 0,1, dan *optimizer* RMSprop dengan rentang *learning rate* antara 0,0001 hingga 0,1. Hasil terbaik ditemukan pada skenario yang menggunakan *optimizer* Adam dengan *learning rate* 0,0001, dengan akurasi sebesar 97,6%

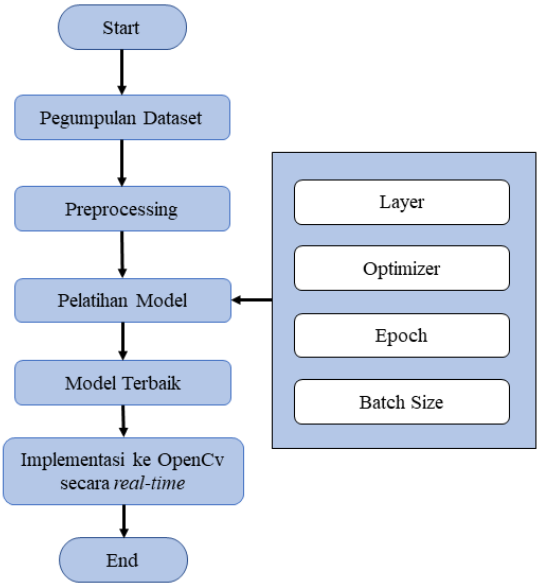
Menurut penelitian [10] menjelaskan pengembangkan model *LeNet-5* berbasis *Convolutional Neural Networks* yang dapat mengidentifikasi karakter tulisan tangan dari data EMNIST dibagi menjadi kumpulan pelatihan dan kumpulan uji, masing-masing berisi 80% dan 20% dari kumpulan data dengan akurasi tinggi. Penggunaan *Python* dengan *library TensorFlow* untuk mengembangkan simulasi algoritma pembelajaran model gabungan dan menggunakan *library OpenCV* untuk melakukan pemrosesan gambar. Penelitian yang dilakukan dapat meningkatkan tingkat pengenalan karakter dengan akurasi 88,70 %.

Menurut penelitian [6] menjelaskan CNN diimplementasikan untuk mengenali karakter dari dataset uji. Pada penelitian ini menggunakan dataset EMNIST untuk mendapatkan akurasi karakter tulisan tangan. Perolehan akurasi yang diperoleh dari 200 gambar pelatihan adalah 65,32% ditingkatkan secara bertahap dengan meningkatnya gambar pelatihan. Akurasinya mencapai 92,91% dengan 1000 gambar pelatihan.

Dengan demikian, penggunaan *Convolutional Neural Network* dalam pengenalan karakter tulisan tangan dapat menghasilkan akurasi yang tinggi. Oleh karena itu, pada penelitian ini *Algoritma Convolutional Neural Network* (CNN) akan diimplementasikan bersama dengan *library OpenCV* untuk digunakan dalam melakukan pendeteksian dan pengenalan huruf alfabet tulisan tangan secara *real-time*, dimana hasil akhir dari penelitian ini berupa perolehan model terbaik dari uji coba beberapa skenario yang nantinya akan mendapatkan nilai akurasi tertinggi.

2. METODE PENELITIAN

Alur penelitian yang digunakan dalam penelitian ini dapat dilihat pada Gambar 1, menggambarkan metode penelitian yang digunakan.



Gambar 1. Alur Penelitian

Pada Gambar 1 terlihat bahwa terdapat lima alur dalam penelitian ini. Pertama pengumpulan dataset citra tulisan tangan. Data huruf alfabet tulisan tangan pada penelitian ini didapatkan dari tulisan tangan mahasiswa dengan jumlah 20 orang sehingga mendapatkan jumlah hasil citra huruf alfabet tulisan tangan sebanyak 1.040 citra. Citra terdiri dari 520 citra huruf kapital (A-Z) dan 520 citra huruf kecil (a-z). Untuk mengorganisir data, citra-citra huruf tersebut dimasukkan ke dalam 26 folder yang merepresentasikan setiap huruf alfabet dari A sampai Z. Data tersebut kemudian dibagi menjadi 90% untuk folder *training* dan 10% untuk folder *testing*. Sebanyak 936 citra akan digunakan untuk *training* dan 104 citra akan digunakan untuk *testing*.

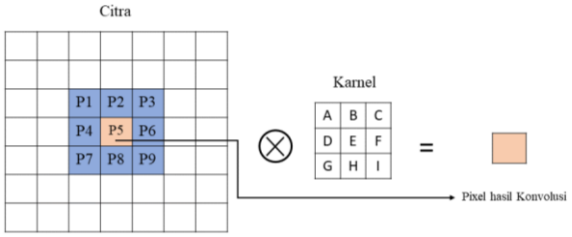
Kedua *preprocessing* citra tulisan tangan. Alur *preprocessing* dilakukan agar kualitas citra meningkat, akurasi lebih optimal, dan mempermudah kinerja sistem dalam mengenali pola [11]. Pada *preprocessing* dataset dilakukan *grayscale* agar dataset citra yang dimiliki menjadi keabu-abuan dengan hanya memiliki satu layer matriks sehingga lebih sederhana. Selanjutnya, dilakukan *gaussian blur* agar citra yang diperoleh lebih bersih dengan mengurangi *noise* pada proses *blurring* citra. Terakhir dilakukan binerisasi agar pola pada tulisan tangan lebih tegas atau tebal. Gambar 2. (a) dan 2(b) terlihat perbedaan dari ketegasan garisnya menunjukkan hasil dari alur *preprocessing* sangat berpengaruh pada proses pelatihan model CNN.

(a) (b)

Gambar 2. (a) Sebelum Preprocessing, (b) Hasil Preprocessing

Ketiga Pelatihan model CNN dalam *deep learning* adalah proses iteratif yang memerlukan eksperimen dan pengujian berulang. Ada beberapa parameter yang perlu disesuaikan dan diuji untuk mendapatkan model terbaik, termasuk *layer*, *optimizer*, *epoch*, dan *batch size*.

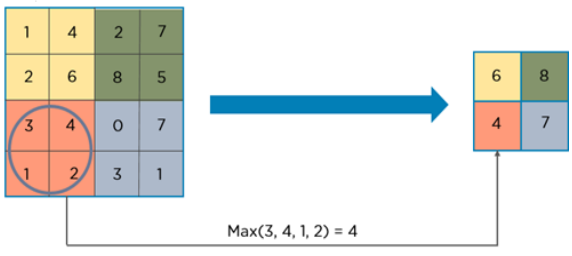
Pengujian *layer* dilakukan dengan mencoba berbagai arsitektur dan kombinasi *layer* untuk menentukan model yang paling cocok untuk tugas tertentu. Beberapa jenis layer yang umum digunakan dalam CNN adalah *convolutional layer, pooling layer*, dan *fully connected layer* [12]. Pada CNN, *convolutional layer* bertanggung jawab untuk melakukan ekstraksi fitur (*feature extraction*) dengan melakukan operasi konvolusi pada input data menggunakan sekumpulan filter yang disebut sebagai kernel. Pada Gambar 3 menunjukkan proses *convolutional layer*.



Gambar 3. Proses Convolutional Layer

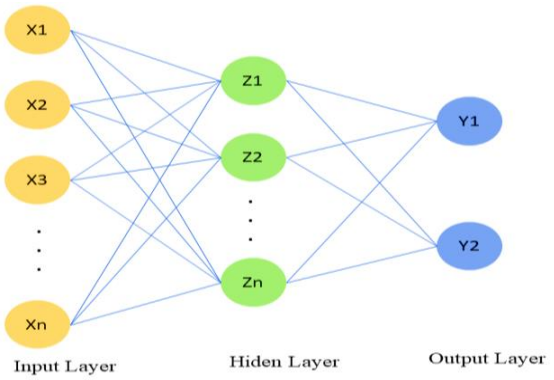
Proses konvolusi pada Gambar 3 dilakukan dengan cara mengalikan setiap nilai pixel pada input data dengan bobot-bobot (*weights*) pada filter kernel dan menjumlahkan hasil perkalian tersebut. Filter kernel ini akan bergerak secara horizontal dan vertikal pada input data dan menghasilkan output berupa matriks yang disebut sebagai *feature map*.

Convolutional layer diikuti dengan *pooling layer* yang berfungsi untuk mengurangi dimensi dari *feature map* dan mengekstraksi informasi yang lebih penting. Ada dua jenis *pooling layer* yang umum digunakan dalam CNN, yaitu *Max Pooling* dan *Average Pooling* [13].



Gambar 4. Max Pooling Layer

Max Pooling mengambil nilai maksimum dari setiap jendela pooling, sedangkan *Average Pooling* mengambil nilai rata-rata dari setiap jendela *pooling* Pada Gambar 4 menunjukkan proses *max- pooling layer*. *Layer* terakhir pada arsitektur CNN, yaitu *fully connected layer*. Pada layer ini, input dari layer sebelumnya dikonversi menjadi sebuah vektor dan dihubungkan ke setiap neuron pada *layer Dense*.



Gambar 5. fully connected layer

Pada Gambar 5 menunjukkan *fully connected layer*. Tujuan dari *layer Dense* ini adalah untuk melakukan klasifikasi atau regresi pada data yang telah melalui proses ekstraksi fitur pada layer sebelumnya. Pada *layer Dense*, *output* dari setiap neuron dihitung dengan melakukan operasi perkalian matriks antara input dari *layer* sebelumnya dan bobot (*weight*) pada *layer Dense*, kemudian ditambah dengan bias (bias) pada setiap neuron. Hasil dari operasi ini kemudian diteruskan ke fungsi aktivasi *softmax* untuk menghasilkan *output* dengan kelas yang lebih dari dua.

Pengujian *Optimizer* digunakan untuk memperbarui nilai bobot dalam model selama pelatihan untuk mendapatkan hasil yang optimal [14]. Terdapat beberapa algoritma optimisasi yang umum digunakan dalam Convolutional Neural Networks (CNN), antara lain Stochastic Gradient Descent (SGD), Adam, dan RMSprop. Pengujian *optimizer* dilakukan dengan mencoba beberapa *optimizer* dan parameter yang berbeda untuk menentukan *optimizer* yang paling cocok untuk dataset dan arsitektur model yang digunakan.

Pengujian *Epoch* digunakan untuk mengamati dampak hasil pelatihan set data dengan variasi jumlah langkah pelatihan (Epoch) [15], sehingga membantu model dalam mempelajari pola yang lebih kompleks dan perlu dilakukan eksperimen dengan jumlah *epoch* yang berbeda untuk menemukan jumlah yang optimal.

Pengujian *Batch size* digunakan untuk menghitung gradien selama pelatihan dengan mengindikasikan jumlah sampel yang diambil dari dataset pelatihan pada setiap iterasi [16]. *Batch size* yang lebih kecil dapat mempercepat pelatihan dan mengurangi penggunaan memori, tetapi juga dapat membuat pelatihan menjadi lebih tidak stabil. Pengujian *batch size* dilakukan dengan mencoba beberapa nilai untuk menentukan nilai optimal untuk dataset dan model yang digunakan.

Dengan melakukan pengujian terhadap parameter-parameter tersebut, dapat membantu dalam membangun model yang lebih optimal dan mengetahui seberapa baik model tersebut dalam memprediksi hasil yang diinginkan sehingga mendapatkan model terbaik sesuai alur penelitian keempat pada Gambar 1.

Terakhir mengimplementasikan model terbaik menggunakan *OpenCV*. Dengan cara ini, pengguna dapat menggunakan *whiteboard* untuk menulis dan menggambar, kemudian gambar tersebut diolah menjadi bentuk input yang sesuai untuk model CNN sehingga sistem dapat langsung mengenali gambar tersebut dan menampilkan hasil prediksi pada layar sebagai output secara *real-time*.

3. HASIL DAN PEMBAHASAN

3.1. Skenario Pengujian *Layer*

Mendapatkan hasil pengujian 3 *layer* dengan fitur (32, 64, 128) memperoleh akurasi pelatihan sebesar 68,53%, akurasi test sebesar 71,15%, nilai loss pelatihan 0,985, dan loss test 0,894. Untuk 3 *layer* dengan fitur (32, 64, 64) menghasilkan akurasi pelatihan 65,62%, akurasi test 63,46%, loss pelatihan 1,149, dan loss test 1,272. Untuk 4 *layer* dengan fitur (32, 64, 64, 128) menghasilkan akurasi pelatihan 79,91%, akurasi test 82,00%, loss pelatihan 0,665, dan loss test 0,549. Untuk 4 *layer* dengan fitur (32, 64, 128, 128) menghasilkan akurasi pelatihan 81,28%, akurasi test 80,04%, loss pelatihan 0,573, dan loss test 0,605. Sedangkan untuk 5 *layer* dengan fitur (32, 32, 64, 64, 128) menghasilkan akurasi pelatihan 85,90%, akurasi test 84,00%, loss pelatihan 0,392, dan loss test 0,485 Dapat dilihat pada Tabel 1.

Tabel 1. Pengujian *Layer*

Layer	Akurasi		Loss	
	training (%)	testing (%)	training	testing
3 (32, 64, 128)	68,53	71,15	0,985	0,894
3 (32, 64, 64)	65,62	63,46	1,149	1.272
4 (32, 64, 64, 128)	79,91	82,00	0,665	0,549
4 (32, 64, 128, 128)	81,28	80,04	0,573	0,605
5 (32, 32, 64, 64, 128)	85,90	84,00	0,392	0,485

Berdasarkan Tabel 1 menunjukkan bahwa dari pengujian layer, hasil terbaik adalah menggunakan 5 layer konvolusi dengan jumlah filter sebanyak 32, 32, 64, 64, dan 128. Layer konvolusi ini diikuti oleh layer *MaxPooling* dan kemudian diakhiri dengan 2 layer *fully connected*. Dari hasil pengujian ini, dapat dilihat bahwa model dengan layer konvolusi yang lebih dalam dan kompleks dapat menghasilkan akurasi yang lebih tinggi pada data *training* dan *testing*, meskipun ada sedikit *overfitting* yang terjadi. Namun, hal ini masih dapat diterima karena akurasinya masih cukup tinggi.

3.2. Skenario Pengujian *Optimizer*

Hasil pengujian menggunakan *optimizer* SGD menunjukkan akurasi pelatihan sebesar 71,83% dan akurasi test sebesar 71,80%, dengan nilai loss pelatihan sebesar 0,926 dan loss test sebesar 0,882. Sementara itu, penggunaan *optimizer* RMSprop

menghasilkan akurasi pelatihan sebesar 87,45%, akurasi test sebesar 82,90%, loss pelatihan sebesar 0,399, dan loss test sebesar 0,886. Selanjutnya, penggunaan *optimizer* Adam menghasilkan akurasi pelatihan sebesar 88,76%, akurasi test sebesar 88,00%, loss pelatihan sebesar 0,323, dan loss test sebesar 0,419. Dapat dilihat pada Tabel 2.

Tabel 2. Pengujian *Optimizer*

Optimizer	Akurasi		Loss	
	training (%)	testing (%)	training	testing
SGD	71,83	71,80	0,926	0,882
RMSprop	87,45	82,90	0,399	0,886
Adam	88,76	88,00	0,323	0,419

Berdasarkan Tabel 2 menunjukkan bahwa pengujian dari semua jenis optimizer, *optimizer* adam yang menghasilkan nilai akurasi terbaik dan memiliki nilai loss yang rendah. Sedangkan untuk *optimizer* RMSprop memiliki rentang nilai loss *training* dan *testing* yang sangat jauh sehingga bisa dikatakan overfitting. Sedangkan untuk *optimizer* SGD peformanya masih dibawah *optimizer* lainnya yang diujicobakan. Sehingga dapat diambil kesimpulan pada skenario pengaruh *optimizer* ini bahwa *optimizer* terbaik, yaitu adam.

3.3. Skenario Pengujian *Epoch*

Mendapatkan hasil pengujian *epoch* sebanyak 10 dengan akurasi pelatihan sebesar 37,77%, akurasi test sebesar 31,99%, nilai loss pelatihan 2,016, dan loss test 2,240. Untuk 50 *epoch* menghasilkan akurasi pelatihan 50,55%, akurasi test 47,00%, loss pelatihan 1,633, dan loss validasi 1,727. Untuk 80 menghasilkan akurasi pelatihan 73,58%, akurasi test 72,63%, loss pelatihan 0,842, dan loss test 0,920. Untuk 100 menghasilkan akurasi pelatihan 88,90%, akurasi test 87,00%, loss pelatihan 0,393, dan loss test 0,499. Dapat dilihat pada Tabel 3.

Tabel 3. Pengujian Epoch

Epoch	Akurasi		Loss	
	training (%)	testing (%)	training	testing
10	37,77	31,99	2,016	2,240
50	50,55	47,00	1,633	1,727
80	73,58	72,63	0,842	0,920
100	88,90	87,00	0,393	0,499

Berdasarkan pada Tabel 3 menunjukkan bahwa Semakin meningkatnya jumlah *epoch* yang digunakan, menghasilkan peningkatan akurasi dan penurunan loss pada pelatihan model. Dalam penelitian ini, penggunaan 100 *epoch* menunjukkan kinerja terbaik dengan menghindari *overfitting* dan *underfitting*, sebagaimana dibuktikan oleh nilai akurasi data *training* dan data *testing* yang tidak terlalu berbeda. Selain itu, nilai akurasi pada data *testing* mencapai 87%, sedikit lebih tinggi daripada penggunaan jumlah *epoch* lainnya. Oleh karena itu, dapat disimpulkan bahwa jumlah *epoch* optimal dalam pengenalan tulisan tangan dalam skenario ini adalah 100.

3.4. Skenario Pengujian *Batch Size*

Mendapatkan hasil pengujian *batch size* sebesar 10 dengan akurasi pelatihan sebesar 77,54%, akurasi test sebesar 71,70%, nilai loss pelatihan 0,983, dan loss test 1,146. Untuk *batch size* 15 menghasilkan akurasi pelatihan 81,10%, akurasi test 80%, loss pelatihan 0,534, dan loss validasi 0,697. Untuk 20 menghasilkan akurasi pelatihan 89,83%, akurasi test 87,00%, loss pelatihan 0,322, dan loss test 0,343. Untuk 30 menghasilkan akurasi pelatihan 90,74%, akurasi test 83,33%, loss pelatihan 0,263, dan loss test 0,554. Dapat dilihat pada Tabel 3.

Tabel 4. Pengujian Batch Size

Batch Size	Akurasi		Loss	
	training (%)	testing (%)	training	testing
10	77,54	71,70	0,983	1,146
15	81,10	80,00	0,534	0,697
20	89,83	87,00	0,322	0,343
30	90,74	83,33	0,263	0,554

Berdasarkan Tabel 4 menunjukkan bahwa pengujian dari semua nilai *batch size*, nilai *batch size* sebesar 20 yang menghasilkan nilai akurasi terbaik dan memiliki nilai *loss* yang rendah. Sedangkan untuk nilai *batch size* 10, 15 dan 30 masih memiliki nilai loss yang tinggi dibandingkan dengan nilai *batch size* 20. *Batch size* dengan nilai 30 memiliki nilai akurasi *training* yang tinggi dari nilai *batch size* lainnya, tetapi memiliki akurasi *testing* yang rendah dari nilai *batch size* 20.

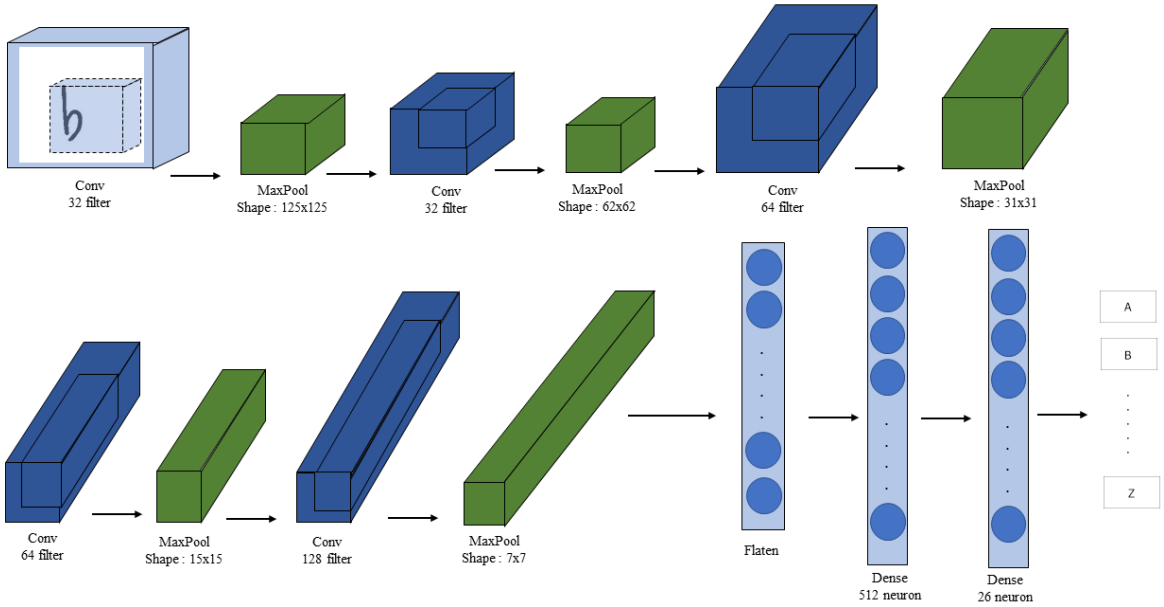
3.5. Model Terbaik

Dari kelima pengujian didapatlah arsitektur model CNN yang terbaik berdasarkan perolehan akurasi tertinggi dan nilai loss terendah. Perolehan arsitektur CNN terbaik menggunakan 5 *layer* dengan fitur (32, 32, 64, 64, 128), *optimizer* Adam, *epoch* sebanyak 100, dan menggunakan *batch size* sebesar 20, Arsitektur model dapat dilihat pada Gambar 6.

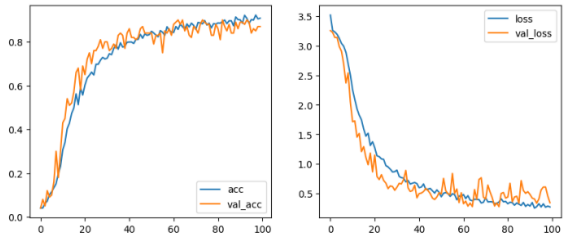
Arsitektur pada Gambar 6 terdiri dari beberapa jenis layer yaitu Conv2D (konvolusi 2 dimensi) dengan kernel 3×3, MaxPooling2D (maksimum *pooling* 2 dimensi) dengan kernel 2×2, Flatten (meluruskan data), dan Dense (*layer fully connected*).

Input dari *layer* pertama adalah gambar dengan ukuran 250×250 *pixel* dengan 3 *channel* (RGB). Kemudian dilakukan konvolusi dengan 32 filter dan dilakukan *max pooling*. Kemudian dilakukan lagi konvolusi dengan 32 filter dan dilakukan *max pooling*. Proses ini diulang dengan 2 *layer* konvolusi dan *max pooling* lagi dengan jumlah filter yang lebih besar, yaitu 64, dan 128. Di setiap layer konvolusi dikuti dengan aktivasi *relu* yang memiliki peran dalam mengaktifkan atau menonaktifkan neuron dalam model, serta dapat mempercepat proses pelatihan model. Penggunaan ReLU dapat

meningkatkan kecepatan pelatihan model, sehingga cocok untuk mencegah masalah *overfitting* pada data pelatihan [17] *connected*. Terdapat 2 *layer Dense* yang digunakan untuk melakukan klasifikasi dengan jumlah *neuron output* sebanyak 26 menggunakan fungsi aktivasi *softmax*, sesuai dengan jumlah kelas yang akan diklasifikasikan.



Gambar 6. Arsitektur Model Terbaik



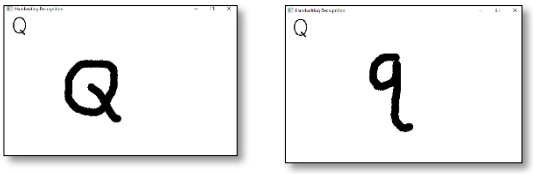
Gambar 7. Grafik Hasil Pelatihan Model

Berdasarkan Gambar 7 loss yang diperoleh dari pelatihan model terbaik sebesar 0,322 untuk data *training* dan 0,343 untuk data *testing*, hal ini bisa dikatakan cukup rendah dan cukup bagus dari model yang telah didapat dengan didukung juga oleh tingginya nilai akurasi dari beberapa pengujian lainnya, yaitu mencapai 87%. Ini menunjukkan bahwa model yang diperoleh sudah cukup baik dikarenakan dari semua percobaan memiliki nilai akurasi tertinggi.

3.6. Implementasi Model CNN ke OpenCV

Pengimplementasian model CNN yang diperoleh dilakukan pada *OpenCV* agar dapat melakukan pengenalan tulisan tangan secara *real time*. Untuk menuliskan huruf pada *OpenCV* dibuatlah kanvas putih yang akan sebagai tampilan awal dari sistem, lalu melakukan inferensi menggunakan model CNN untuk mengenali huruf pada kanvas yang diproses dengan mengubah ukuran, format warna, dan normalisasi intensitas piksel sesuai model CNN.

Pada sistem dibuat tombol fungsi, seperti ketika menekan tombol ‘r’ pada *keyboard* maka sistem akan memunculkan hasil prediksi di pojok kiri atas, ketika menekan ‘c’ maka sistem akan menghapus kanvas, dan ketika menekan ‘q’ maka akan keluar dari sistem. Tampilan sistem pengenalan tulisan tangan secara *real time* dapat dilihat pada Gambar 8.



(a) (b)

Gambar 8. (a) Pengenalan tulisan tangan huruf kapital (b) Pengenalan tulisan tangan huruf kecil

Dapat dilihat pada Gambar 8.(a) sistem dapat mengenali huruf kapital dan pada Gambar 8.(b) sistem dapat mengenali huruf kecil. Ini dikarenakan model CNN yang digunakan pada dataset uji nya yang memiliki variasi huruf kapitan dan huruf kecil, namun pada pelabelannya tetap menggunakan huruf kapital sebanyak 26 label. Oleh karena itulah sistem dapat mengenali huruf kapital dan kecil secara *real time*.

4. DISKUSI

Penggunaan *Convolutional Neural Network* untuk pengenalan pola tulisan tangan memiliki tingkat akurasi yang sangat tinggi. Hasil ini dibuktikan dengan penelitian terdahulu yang membahas identifikasi huruf Hiragana Jepang

menggunakan CNN Arsitektur VGG-16 oleh [9] dan [6]. Tingkat akurasi dari kedua penelitian tersebut mencapai lebih dari 90%. Namun pada penelitian tersebut belum mengimplementasikan model ke dalam sistem untuk mengklasifikasi huruf.

Selain itu, terdapat penelitian yang telah melakukan pengembangan untuk membuat sistem pengklasifikasian tulisan tangan menggunakan *library OpenCV* [10]. Namun penelitian tersebut mengadopsi arsitektur *LeNet-5* tanpa melakukan modifikasi atau pengujian lebih lanjut pada layer-layernya. Keterbaruan pada penelitian ini dibandingkan dengan penelitian sebelumnya, yaitu penelitian menggunakan pengujian pada layer-layernya dan mengimplementasikan pada *library OpenCV* untuk melakukan klasifikasi secara realtime.

Hasil yang diperoleh dari penelitian ini berupa arsitektur model CNN yang diimplementasikan pada *OpenCV*. Adapun arsitektur yang diperoleh, yaitu menggunakan 5 layer dengan fitur (32, 32, 64, 64, 128), *optimizer* Adam, *epoch* sebanyak 100, dan menggunakan batch size sebesar 20. Menghasilkan akurasi pelatihan 89,83%, akurasi test 87,00%, loss pelatihan 0,322, dan loss test 0,343. Grafik pelatihannya dapat dilihat pada Gambar 7. Namun akurasi tersebut masih tergolong tidak terlalu tinggi karena dataset uji merupakan salah satu faktor pengaruh tingkat akurasi yang dibuktikan dari penelitian-penelitian terdahulu. Sehingga untuk mendapatkan akurasi yang lebih tinggi diperlukan penambahan dataset uji dengan menggunakan arsitektur yang sama.

5. KESIMPULAN

Berdasarkan penelitian yang dilakukan penggunaan *convolutional neural network* (CNN) dalam pengenalan huruf tulisan tangan secara real-time dapat menjadi pendekatan yang akurat dan efektif. Implementasi CNN bersama dengan *library OpenCV* dalam penelitian ini menghasilkan model yang dapat mengenali huruf tulisan tangan dengan akurasi yang lumayan tinggi, yaitu akurasi pelatihan 89,83%, akurasi test 87,00%. Namun, penelitian ini mungkin memiliki batasan, seperti ukuran dataset yang digunakan, variasi tulisan tangan atau lingkungan pengujian yang berbeda, dan dapat menjadi sumber ketidakakuratan atau pengurangan kinerja model. Dalam hal ini diharapkan penelitian selanjutnya meningkatkan keberagaman dan *representativitas* data yang dapat meningkatkan akurasi dan generalisasi model yang dikembangkan sehingga dapat digunakan dalam berbagai aplikasi praktis, seperti pengenalan wajah, deteksi objek, atau analisis medis dalam waktu nyata.

DAFTAR PUSTAKA

[1] A. Raup, W. Ridwan, Y. Khoeriyah, Q. Yuliati Zaqiah, dan U. Islam Negeri Sunan Gunung Djati Bandung, "Deep learning dan penerapannya dalam pembelajaran," *JIIP (Jurnal Ilmiah Ilmu Pendidikan)* , vol. 5, pp. 3258–3267, 2022, [Daring]. Tersedia pada: http://Jiip.stkipyapisdompu.ac.id

[2] G. F. Fitriana, "Pengenalan tulisan tangan angka menggunakan Self Organizing Maps (SOM)," *Technology and Science (BITS)*, vol. 3, no. 1, pp. 31–42, 2021, doi: 10.47065/bits.v3i1.1002.

[3] N. Saqib, K. F. Haque, V. P. Yanambaka, dan A. Abdelgawad, "Convolutional-neural-network-based handwritten character recognition: an approach with massive multisource data," *Algorithms*, vol. 15, no. 4, Apr 2022, doi: 10.3390/a15040129.

[4] O. Sudana, I. W. Gunaya, dan I. K. G. D. Putra, "Handwriting identification using deep convolutional neural network method," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 18, no. 4, pp. 1934–1941, 2020, doi: 10.12928/TELKOMNIKA.V18I4.14864.

[5] R. Fadiyah Alya dan M. Wibowo, "Classification of batik motif using transfer learning on convolutional neural network (CNN)," vol. 4, no. 1, pp. 161–170, 2023, doi: 10.20884/1.jutif.2023.4.1.564.

[6] I. Khandokar, M. Hasan, F. Ernawan, S. Islam, dan M. N. Kabir, "Handwritten character recognition using convolutional neural network," dalam *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jun 2021. doi: 10.1088/1742-6596/1918/4/042152.

[7] I. Riadi, A. Fadlil, P. Annisa, A. Dahlan, dan J. Soepomo Sh, "Identifikasi Tulisan Tangan Huruf Katakana Jepang Dengan Metode Euclidean," *Jurnal Sains Komputer & Informatika (J-SAKTI*, vol. 4, pp. 29–37, 2020, [Daring]. Tersedia pada: http://tunasbangsa.ac.id/ejurnal/index.php/jsakti

[8] T. Matius Surya Mulyana, "Implementasi algoritma freeman chain code dan algoritma k-nearest neighbor dalam pengenalan huruf mandarin," *Jurnal Riset Komputer)*, vol. 9, no. 4, pp. 2407–389, 2022, doi: 10.30865/jurikom.v9i4.4532.

[9] A. Willyanto, D. Alamsyah, dan H. Irsyad, "Identifikasi tulisan tangan aksara jepang hiragana menggunakan metode CNN arsitektur VGG-16," *Jurnal Algoritme*, vol. 2, no. 1, pp. 1–11, 2021.

[10] C. S. Ley, A. Syammi, dan B. Ab Ghafar, "Handwritten character recognition using convolutional neural network," *Progress in Engineering Application and Technology*, vol. 2, no. 1, pp. 593–611, 2021, doi:

10.30880/peat.2021.02.01.058.

[11] N. D. Miranda, L. Novamizanti, dan S. Rizal, "Convolutional neural network pada klasifikasi sidik jari menggunakan resnet-50," *Jurnal Teknik Informatika (Jutif)*, vol. 1, no. 2, pp. 61–68, Des 2020, doi: 10.20884/1.jutif.2020.1.2.18.

[12] E. Tanuwijaya, R. L. Lordianto, dan R. A. Jasin, "Pengenalan wajah manusia pada aplikasi video conference menggunakan metode pipeline cnn," *Jurnal Teknik Informatika (JUTIF)*, vol. 3, no. 2, pp. 421–427, 2022, doi: 10.20884/1.jutif.2022.3.2.219.

[13] R. Mehindra Prasmatio, B. Rahmat, dan I. Yuniar, "Deteksi dan pengenalan ikan menggunakan algoritma convolutional neural network," *Jurnal Informatika dan Sistem Informasi (JIFoSI)*, vol. 1, no. 2, pp. 510–521, 2020.

[14] D. Irfan, R. Rosnelly, M. Wahyuni, J. T. Samudra, dan A. Rangga, "Perbandingan optimasi sgd, adadelta, dan adam dalam klasifikasi hydrangea menggunakan cnn," 2022. [Daring]. Tersedia pada: http://jurnal.goretanpena.com/index.php/JSSR

[15] R. Haris Alfikri *dkk.*, "Pembangunan aplikasi penerjemah bahasa isyarat dengan metode cnn berbasis android," *TEKNOINFO*, vol. 16, no. 2, pp. 183–197, 2022, [Daring]. Tersedia pada: https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

[16] Y. N. Fuadah, I. D. Ubaidullah, N. Ibrahim, F. F. Taliningsing, N. K. Sy, dan M. A. Pramuditho, "Optimasi convolutional neural network dan k-fold cross validation pada sistem klasifikasi glaukoma," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 10, no. 3, pp. 728, Jul 2022, doi: 10.26760/elkomika.v10i3.728.

[17] A. D. Aryanto, J. Santoso, dan D. D. Purwanto, "Sistem rekomendasi obat pengganti menggunakan metode cnn," *Surabaya Jurnal Sistem Cerdas dan Rekayasa (JSCR)*, vol. 3, no. 1, pp. 2656–7504, 2021.