Jurnal Rekayasa Teknologi dan Komputasi
Volume 1 Nomor 1 Juni 2025
e-ISSN : 2443-2229 (Online)
PROGRAM STUDI TEKNIK INFORMATIKA FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PONOROGO

RANCANG BANGUN MODEL PENGENALAN TULISAN DENGAN *CONVOLUTIONAL NEURAL NETWORK* (CNN)

[1]Abi Kurniawan, [2]Ismail Abdurrozzaq Z., [3]Jamilah Karaman
[1]Program Studi Teknik Informatika, [2]Falkutas Teknik, [3]Universitas Muhammadiyah Ponorogo
Jl. Budi Utomo No.10 Ronowijayan, Ponorogo

kurniawanabi2217@gmail.com[1]

Abstrak

Pengenalan tulisan tangan telah menjadi topik penelitian yang signifikan dalam bidang visi komputer dan pemrosesan bahasa alami. Penelitian ini bertujuan untuk mengembangkan sistem pengenalan tulisan tangan Hijaiyah menggunakan Convolutional Neural Network (CNN) di Taman Pendidikan Al-Qur'an (TPA) Masjid Baitul Hidayah, desa Kepuhrejo, Kabupaten Magetan. Sistem ini diharapkan dapat membantu siswa dalam mengenali dan menghafal Hijaiyah dengan lebih efektif serta mendukung pengajar dalam penilaian kemampuan siswa. Pengembangan aplikasi dilakukan menggunakan CNN dengan antarmuka GUI berbasis Tkinter. Dataset yang digunakan mencakup karakter tulisan tangan yang beragam untuk melatih model CNN agar dapat mengenali berbagai variasi tulisan tangan. Aplikasi ini memungkinkan pengguna untuk menggambar karakter atau angka pada kanvas digital, yang kemudian dikenali oleh model CNN dan hasil pengenalan ditampilkan secara real-time. Implementasi CNN menunjukkan akurasi tinggi sebesar 99,19% dalam klasifikasi gambar tulisan tangan. Hasil pengujian menunjukkan aplikasi ini memiliki tingkat akurasi yang tinggi dan respons yang cepat, menjadikannya solusi efektif untuk pembelajaran Hijaiyah di TPA.

***Kata Kunci**: GUI, CNN, Neural Network*

Abstract

Handwriting recognition has become a significant research topic in the fields of computer vision and natural language processing. This study aims to develop a Hijaiyah handwriting recognition system using Convolutional Neural Networks (CNN) at Taman Pendidikan Al-Qur'an (TPA) Baitul Hidayah Mosque, Kepuhrejo village, Magetan Regency. This system is expected to assist students in recognizing and memorizing Hijaiyah more effectively and support teachers in assessing students' abilities. The application development was carried out using CNN with a Tkinter-based GUI interface. The dataset used includes diverse handwritten characters to train the CNN model to recognize various handwriting styles. This application allows users to draw characters or numbers on a digital canvas, which are then recognized by the CNN model, and the recognition results are displayed in real-time. The implementation of CNN demonstrates a high accuracy of 99.19% in handwriting image classification. The testing results show that this application has a high accuracy rate and quick response time, making it an effective solution for Hijaiyah learning at TPA.

***Keywords**: GUI, CNN, Neural Network*

I. PENDAHULUAN

Pengenalan tulisan tangan adalah topik penting dalam bidang visi komputer dan pemrosesan bahasa alami, yang relevan dalam berbagai aplikasi seperti pengenalan karakter, pengolahan dokumen otomatis, dan interaksi manusia-komputer. Penggunaan *Convolutional Neural Networks* (CNN) menonjol dalam bidang ini, terinspirasi oleh cara manusia memproses gambar, dan telah menunjukkan hasil impresif dalam tugas pengenalan tulisan tangan. CNN memanfaatkan lapisan konvolusi untuk mengekstrak fitur penting dari gambar tulisan tangan dan lapisan berikutnya untuk mengklasifikasikan karakter atau kata (Krizhevsky, Sutskever, & Hinton, 2012).

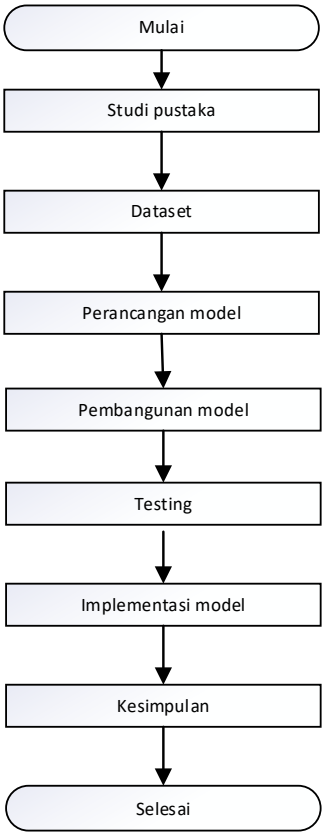
Taman Pendidikan Al-Qur'an (TPA) di Masjid Baitul Hidayah, Desa Kepuhrejo, berfungsi sebagai lembaga pendidikan non-formal yang mengajarkan dasar-dasar ajaran Islam, terutama dalam membaca dan menulis huruf Hijaiyah. Dengan sekitar 20 siswa dan 2 pengajar, TPA ini menghadapi kendala seperti variasi kemampuan siswa dan keterbatasan pengajar dalam penilaian serta jumlah dan kualifikasi pengajar. Teknologi dalam pendidikan dapat meningkatkan efektivitas pembelajaran, dan pengembangan sistem pengenalan tulisan Hijaiyah berbasis *Convolutional Neural Network* (CNN) sangat relevan. CNN, dengan kemampuannya mengenali pola dan bentuk, dapat mengenali huruf Hijaiyah dengan akurasi tinggi (abror & Nopiyanto, 2021).

Pengembangan sistem pengenalan tulisan Hijaiyah berbasis CNN dapat memberikan solusi inovatif bagi TPA di Masjid Baitul Hidayah. Sistem ini membantu siswa dalam mengenali dan menghafal huruf atau angka Hijaiyah lebih cepat serta membantu pengajar dalam penilaian dan pemahaman kesulitan siswa. Diharapkan, sistem ini akan memberikan dampak positif bagi masyarakat Desa Kepuhrejo dengan meningkatkan akses pendidikan agama dan kualitas kehidupan beragama.

Oleh karena itu, pengembangan sistem pengenalan tulisan Hijaiyah berbasis teknologi di TPA Masjid Baitul Hidayah merupakan langkah strategis untuk meningkatkan kualitas pendidikan agama dan mencetak generasi muda yang lebih cerdas dan berakhlak.

II. METODE PENELITIAN

Penelitian Studi ini menunjukkan bahwa proses penelitian terdiri dari urutan tindakan, atau langkah-langkah, yang dilakukan oleh peneliti mulai dari tahap awal hingga tahap akhir. Dalam kasus ini, tahapan penelitian yang dilakukan termasuk:



Gambar 1. Flowchart Metode Penelitian

2.1 Dataset

Dataset MNIST (*Modified National Institute of Standards and Technology*) adalah koleksi terkenal yang digunakan secara luas dalam penelitian dan pengembangan algoritma pengenalan pola dan pembelajaran mesin. Dataset ini terdiri dari 70.000 gambar tangan dari angka-angka 0 hingga 9 yang ditulis tangan, yang dibagi menjadi dua subset: 60.000 gambar untuk set pelatihan dan 10.000 gambar untuk set pengujian. Setiap gambar memiliki resolusi 28x28 piksel dalam skala abu-abu, yang menghasilkan 784 fitur per gambar.

2.2 Pre-processing

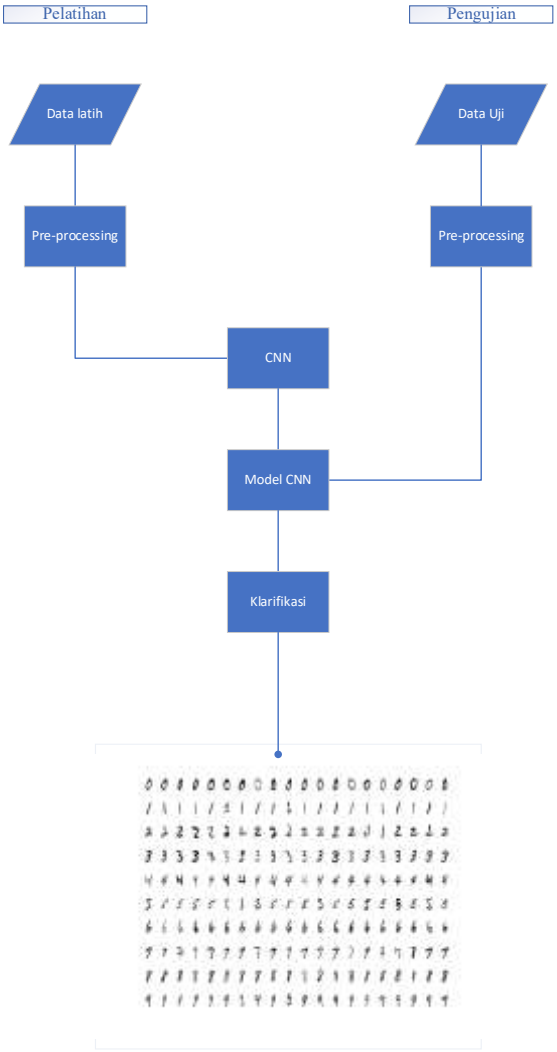
Dalam penelitian ini, *pre-processing* merupakan tahapan krusial yang bertujuan untuk meningkatkan kualitas citra sehingga hasil pengolahan mencapai tingkat optimal. Sebelum dilakukan analisis mendalam, citra dalam dataset disesuaikan ukurannya melalui proses *resizing*. Variasi ukuran citra dalam dataset termasuk 28 x 28, 20 x 20, dan 10 x 10 piksel untuk mengamati pengaruh ukuran terhadap kinerja penelitian. Penggunaan ukuran piksel yang lebih kecil juga meningkatkan efisiensi proses *training* pada tahapan berikutnya. Selanjutnya, citra yang telah di-*resize* diubah dari format RGB ke *grayscale*. Transformasi ini bertujuan untuk menyederhanakan citra menjadi matriks satu dimensi warna, mempermudah proses lanjutan pada *layer* konvolusi dalam studi ini.

Gambar 3. Proses *resizing* citra

2.3 Perancangan Model

Metode *convolutional neural network* digunakan untuk mengidentifikasi pola dalam tulisan tangan. Studi ini terdiri dari dua tahap utama: tahap pelatihan dan tahap pengujian. Tahap pelatihan melibatkan pengolahan gambar dari data yang dilatih untuk membentuk model klasifikasi dengan menggunakan *convolutional neural network*. Di sisi lain, tahap pengujian bertujuan untuk mengevaluasi kinerja model yang telah dilatih dengan melihat gambar dari data uji. Gambar 2 menunjukkan proses klasifikasi.

.



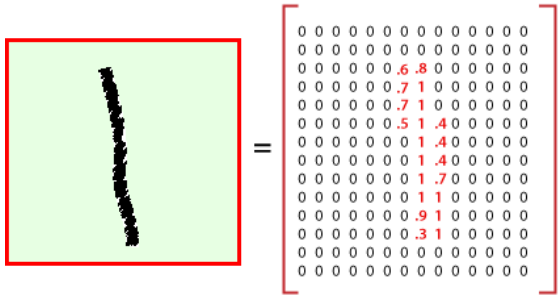
Gambar 2. Proses Klasifikasi *Convolution Neural Network*

2.4 Pembangunan Model Convolutional Neural Network

Penelitian ini menggunakan metode *Convolutional Neural Network* (CNN) untuk klasifikasi. Pelatihan model dimulai dengan gambar 28x28 piksel yang diubah menjadi *grayscale* sebagai data latih. Proses pembelajaran fitur melibatkan layer konvolusi dan *layer pooling*. Selanjutnya, gambar diproses melalui lapisan yang benar-benar terhubung dengan *dropout* untuk melakukan klasifikasi. Jumlah data yang digunakan dalam penelitian ini dibagi menjadi 30 persen untuk data uji dan 70 persen untuk data latih.

1. *Input* citra

   Citra yang telah dimasukkan akan digambarkan dalam bentuk matriks citra dua dimensi untuk memungkinkan gambar melalui proses pembelajaran fitur.

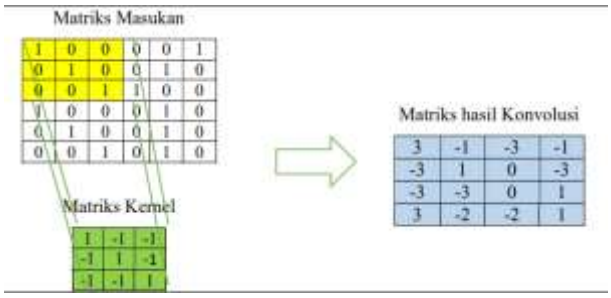


Gambar 4. Representasi citra menjadi matriks

2. *Convolution*

   Lapisan *Convolution Layer* menyaring matriks dari gambar yang dimasukkan. Setelah proses konvolusi, ukuran matriks akan berkurang, sehingga tidak ada *padding* untuk menjaga ukuran matriks. Matriks kernel yang digunakan untuk proses konvolusi adalah 3x3 dan 5x5. Tujuan dari pemilihan matriks kernel yang berbeda ini adalah untuk mengevaluasi dampak ukuran matriks kernel terhadap penelitian ini. *Layer pooling* akan menerima input dari layer konvolusi.

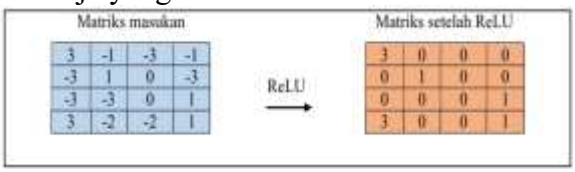
..



Gambar 5. Proses *convolution*

Pada tahap perhitungan di lapisan konvolusi, yang menghasilkan nilai negatif, proses tambahan dilakukan untuk menghilangkan nilai negatif pada matriks citra. Dalam penelitian ini, nilai negatif diubah menjadi 0, dengan menggunakan fungsi aktivasi ReLU. Teknik ini telah digunakan secara luas

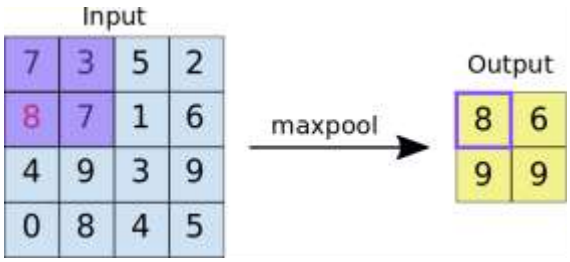
dalam penelitian sebelumnya tentang jaringan saraf konvolusi dan terbukti memiliki kinerja yang baik.



Gambar 6. Penerapan fungsi aktivasi ReLU pada matriks hasil *convolution*

3. *Max Pooling*

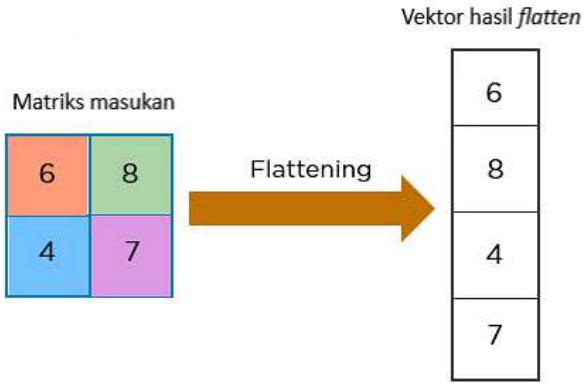
Dalam penelitian ini, penggunaan *layer pooling* bertujuan untuk mengurangi dimensi citra, sehingga meningkatkan efisiensi proses pembentukan peta fitur. Metode *pooling* yang digunakan melibatkan matriks berukuran 2x2 dengan *stride* sebesar 2. Dengan demikian, proses *pooling* ini akan melakukan perpindahan sejauh 2 indeks dan mengekstraksi nilai maksimum dari setiap area *pooling*, yang dikenal sebagai *max pooling*.



Gambar 7. Proses *max pooling*

4. *Flatten*

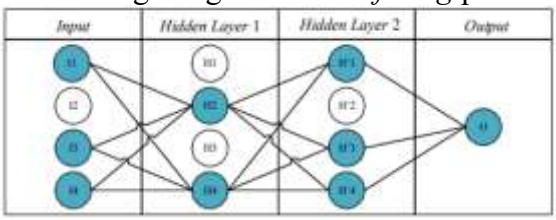
Dalam penelitian ini, hasil dari tahap konvolusi dan *max pooling* yang telah dilakukan sebelumnya akan diolah melalui proses *flattening*. *Flattening* merupakan transformasi yang mengubah struktur matriks menjadi *vektor*, sehingga format masukan dapat sesuai dengan kebutuhan jaringan saraf tiruan. *Output* dari tahap flattening ini kemudian diteruskan ke lapisan *fully connected* untuk menjalani proses klasifikasi.



Gambar 8. Proses *flatten*

5. *Fully Connected Layer dengan dropout*

Pada struktur model yang dibangun, terdapat dua lapisan tersembunyi dengan masing-masing 64 *neuron*. Meskipun demikian, penggunaan *fully connected layer* berpotensi menyebabkan overfitting. Oleh karena itu, metode *dropout* diterapkan untuk mengatasi potensi *overfitting* tersebut. *Dropout* secara acak menonaktifkan beberapa *neuron* pada setiap lapisan, mengurangi jumlah *weigh*t dan *neuron* yang terlibat dalam perhitungan. Tindakan ini bertujuan untuk mengurangi risiko *overfitting* pada model.



Gambar 9. *Fully connected layer* dengan *dropout*

2.5 Testing

Tahapan ini menerapkan metode pengujian *whitebox* untuk menganalisis fungsi internal algoritma atau objek dalam aplikasi, khususnya pada antarmuka pengguna (GUI). Analisis meliputi langkah-langkah fungsional seperti mulai, *preprocessing*, Integrasi CNN, memasukkan gambar tulisan tangan, penggambaran tulisan tangan, prediksi, menghapus, hasil, dan selesai. Tujuannya adalah untuk memverifikasi operasi fungsi-fungsi GUI tulisan tangan.

2.6 Implementasi Model

- Penelitian ini menggunakan metode *Convolutional Neural Network* (CNN) untuk pengenalan tulisan tangan, di mana keberhasilan proses pelatihan sangat dipengaruhi oleh arsitektur yang digunakan. Diperlukan serangkaian uji coba untuk menentukan arsitektur jaringan terbaik. Parameter yang diatur pada sistem CNN meliputi dimensi input citra, jumlah *filter kernel*, ukuran *kernel*, *hidden layer*, jumlah *neuron* pada setiap *hidden layer*, dropout, dan *learning rate*. Evaluasi model dilakukan melalui pengujian dengan menggunakan dataset dan parameter eksperimen seperti dimensi *kernel* konvolusi, ukuran lapisan, jumlah *epoch*, dan skenario pelatihan. Kinerja model diukur dengan metrik akurasi, presisi, dan *recall* menggunakan teknik *confusion matrix*.
- Perancangan sistem mempertimbangkan tujuan fungsionalitas aplikasi untuk mengidentifikasi tulisan tangan, dengan menggunakan antarmuka pengguna grafis (GUI). Setelah model CNN selesai dilatih dan dievaluasi, model disimpan dalam format yang sesuai dan dimuat kembali dalam aplikasi. Di mana pengguna dapat menggambar angka atau karakter pada area putih di sebelah kiri dan sistem akan menampilkan prediksi pada kotak biru di sebelah kanan. Terdapat tombol "Prediksi" dan "Hapus" untuk memprediksi input tulisan tangan atau menghapusnya.

III. HASIL DAN PEMBAHASAN

3.1 Implementasi CNN

1. *Load* Dataset

```
Downloading data from https://storage.googleapis.com/tensorflow/tf
11490434/11490434 [==============================] - 0s 0us/step
(60000, 28, 28) (60000,)
```

Gambar 10. *Load Dataset*

Output tersebut merupakan komponen dari proyek yang menggunakan dataset MNIST, yang berisi gambar-gambar digit tulisan tangan. Pada tahap awal, program melakukan inisialisasi terhadap dataset MNIST dengan mengimpor data latihan dan data uji. Dataset latihan terdiri dari 60.000 gambar dengan resolusi 28x28 piksel, dan terdapat juga 60.000 label yang sesuai. Selain itu, program ini mengunduh dataset MNIST dari URL yang tersedia, dengan ukuran file sekitar 11.490.434 *byte*. Informasi ini memberikan detail tentang jumlah sampel, dimensi setiap gambar, serta jumlah label yang terkait.

2. *Reshape* Data

Proses pengolahan data untuk model pembelajaran mesin yang menggunakan citra mencakup pengubahan dimensi data latih (x_train) dan data uji (x_test) menjadi 28x28 piksel dengan 1 channel warna (*grayscale*) menggunakan metode reshape. Langkah ini memastikan bahwa setiap citra memiliki dimensi yang konsisten tanpa mengubah data aslinya. Selanjutnya, input_shape diinisialisasi dengan nilai (28, 28, 1) untuk merepresentasikan dimensi citra, di mana ukuran citra adalah 28x28 piksel dan jumlah channel warna adalah 1. Inisialisasi input_shape penting dalam konfigurasi model karena menyediakan informasi tentang bentuk *input* yang akan diterima oleh model.

3. *One hot encoding*

```
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Gambar 11. *One hot encoding*

Kode ini menggunakan fungsi `to_categorical` dari *library Keras* untuk mengonversi label kelas pada data pelatihan (`y_train`) dan data pengujian (`y_test`) menjadi bentuk *one-hot encoding*. Label awalnya berbentuk *integer* yang mewakili kelas tertentu. Misalnya, dalam kasus klasifikasi 10 kelas (seperti pengenalan angka), setiap label adalah angka dari 0 hingga 9. Fungsi `to_categorical` mengubah label ini menjadi *vektor biner*, di mana hanya satu elemen yang bernilai 1 (menunjukkan kelas tersebut), sementara elemen lainnya bernilai 0. Parameter kedua pada fungsi tersebut, yaitu 10, menunjukkan jumlah total kelas yang ada. Hasilnya, setiap label *integer* diubah menjadi *vektor* panjang 10, yang mempermudah algoritma pembelajaran mesin dalam memproses dan belajar dari data tersebut.

4. *Normalize* Data

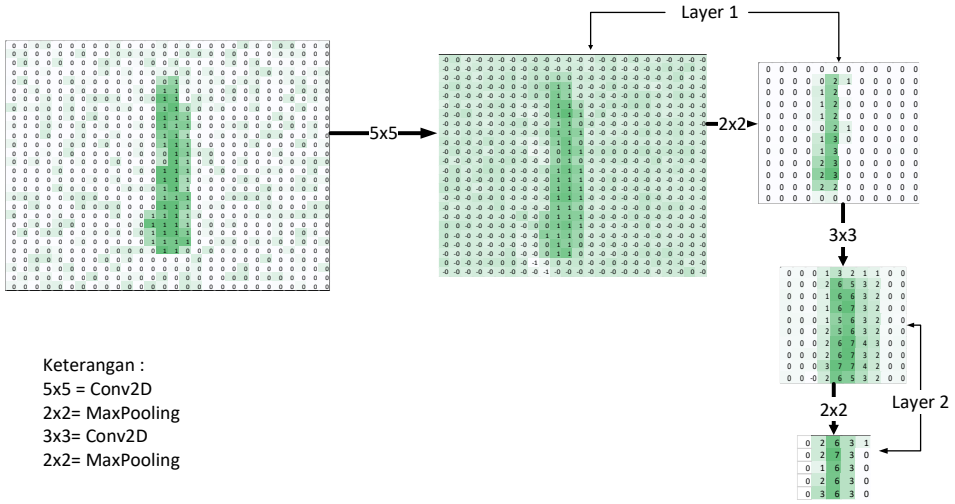
```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

Gambar 12. *Normalize* Data

Dalam *Normalize* Data, data gambar awalnya diubah menjadi tipe data float32, yang memungkinkan perhitungan yang lebih presisi. Selanjutnya, data gambar dibagi dengan nilai 255 untuk menskalakan nilai piksel dari rentang 0-255 menjadi rentang 0-1, yang merupakan praktik umum dalam normalisasi data gambar agar jaringan saraf lebih mudah dan lebih cepat untuk dilatih. Setelah proses normalisasi, bentuk (*shape*) dari data pelatihan (x_train) dan data pengujian (x_test) diperiksa dan ditampilkan. Dari hasil keluaran, terlihat bahwa terdapat 60.000 sampel pelatihan dan 10.000 sampel pengujian, masing-masing dengan dimensi 28x28 piksel dan 1 saluran (*grayscale*).

5. CNN Model

Model ini dimulai dengan inisialisasi objek *Sequential*, yang memungkinkan penambahan lapisan *neural network* secara bertahap. Pertama, lapisan konvolusi (Conv2D) dengan 32 filter berukuran 5x5 dan fungsi aktivasi ReLU ditambahkan, dengan *input* dalam bentuk tertentu (input_shape). Kemudian, lapisan *pooling* maksimum (MaxPooling2D) berukuran 2x2 diterapkan untuk mengurangi dimensi fitur. Proses ini diulangi dengan lapisan konvolusi kedua yang memiliki 64 filter berukuran 3x3, diikuti dengan lapisan pooling maksimum. Selanjutnya, lapisan *Flatten* digunakan untuk mengubah data 2D menjadi vektor 1D, yang kemudian diteruskan ke lapisan *dense* (Dense) dengan 128 *neuron* dan fungsi aktivasi ReLU. Untuk mencegah *overfitting*, lapisan *dropout* dengan rasio 0.3 ditambahkan. Setelah itu, lapisan *dense* lain dengan 64 *neuron* dan fungsi aktivasi ReLU diikuti dengan lapisan *dropout* dengan rasio 0.5 ditambahkan. Akhirnya, lapisan *output* (Dense) dengan jumlah *neuron* sesuai dengan jumlah kelas (num_classes) dan fungsi aktivasi *softmax* ditambahkan, yang menghasilkan probabilitas untuk setiap kelas dalam tugas klasifikasi.



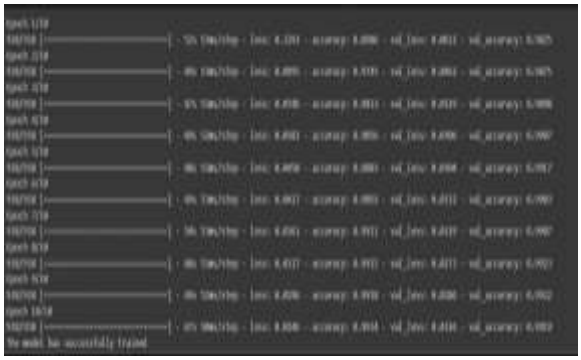
Gambar 13. Matriks CNN

6. *Compile* Model

Melatih model pembelajaran mesin, khususnya model jaringan saraf tiruan, dengan menggunakan pustaka Keras dalam bahasa pemrograman Python. Perintah `model.compile` digunakan untuk mengonfigurasi proses pelatihan model. Parameter `loss='categorical_crossentropy'` menentukan fungsi kerugian yang digunakan untuk mengevaluasi seberapa akurat prediksi model dibandingkan dengan label sebenarnya, yang sesuai untuk masalah klasifikasi dengan banyak kelas. Parameter `optimizer='Adam'` menetapkan algoritma optimasi yang digunakan untuk memperbarui bobot model berdasarkan gradien dari fungsi kerugian, di mana Adam merupakan salah satu algoritma yang efisien dan umum digunakan. Terakhir, parameter `metrics=['accuracy']` menunjukkan metrik yang digunakan untuk menilai kinerja model selama pelatihan dan pengujian, dalam hal ini akurasi, yang mengukur persentase prediksi yang benar dari total prediksi.

7. *Train* Model

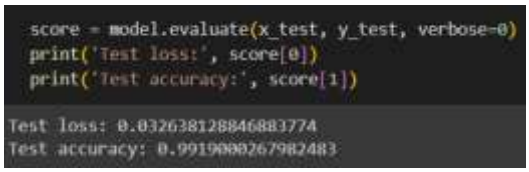
Terlihat pada gambar 14 selama pelatihan, model melalui 10 *epoch*, dan pada setiap *epoch* ditampilkan metrik kinerja seperti `loss` (kerugian), `accuracy` (akurasi pelatihan), `val_loss` (kerugian validasi), dan `val_accuracy` (akurasi validasi). Dari output, terlihat bahwa seiring bertambahnya *epoch*, nilai kerugian menurun dan akurasi meningkat, menunjukkan bahwa model semakin baik dalam melakukan prediksi. Setelah selesai, pesan "*The model has successfully trained*" ditampilkan sebagai indikasi bahwa proses pelatihan telah selesai.



Gambar 14. *Train* Model

8. Evaluasi model

Program yang ditampilkan dalam gambar 15 menunjukkan proses evaluasi model *neural network* yang telah dilatih sebelumnya menggunakan pustaka Keras. Fungsi `model.evaluate` digunakan untuk mengukur kinerja model pada dataset uji (`x_test` dan `y_test`). Parameter `verbose=0` mengindikasikan bahwa proses evaluasi tidak akan menampilkan output progres di konsol.

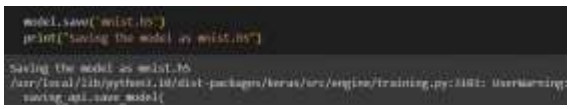


Gambar 15. Evaluasi Model

Hasil evaluasi pada gambar 15 disimpan dalam variabel `*score*`, yang berisi dua nilai: `score[0]` untuk nilai kerugian (*loss*) pada data uji dan `score[1]` untuk akurasi pada data uji. Program kemudian mencetak kedua nilai ini dengan perintah `*print*`. Dari output yang ditampilkan, terlihat bahwa nilai kerugian (*test loss*) adalah 0.0326 dan akurasi pada data uji (*test accuracy*) adalah 0.9910, yang menunjukkan bahwa model memiliki kinerja yang baik dan mampu membuat prediksi yang akurat pada data uji

9. *Save* model

Pada gambar 16 tersebut, terlihat proses penyimpanan model jaringan *neural* yang telah terlatih menggunakan pustaka Keras. Tindakan model.save('mnist.h5') dilakukan untuk menyimpan keseluruhan arsitektur, bobot, dan konfigurasi pelatihan model ke dalam satu file dengan format HDF5 (.h5). Langkah ini memungkinkan untuk memuat kembali model tanpa perlu melakukan pelatihan ulang dari awal. Pesan *print*("Saving the model as mnist.h5") diprint ke konsol untuk memberitahukan pengguna bahwa model telah berhasil disimpan dengan nama file mnist.h5.



Gambar 16. *Save* Model

3.2 GUI Aplikasi Klasifikasi Pengenalan Tulisan

Program dibawah adalah aplikasi GUI (*Graphical User Interface*) berbasis *Python* menggunakan pustaka *tkinter* untuk mengenali tulisan tangan, khususnya angka ke hijaiyah. Aplikasi ini memuat model pembelajaran mesin yang telah dilatih (mnist.h5) untuk mengenali digit tulisan tangan.

Pada gambar 17 GUI, pengguna dapat menggambar digit di kanvas putih. Setelah menggambar, pengguna dapat menekan tombol "Pengenalan" untuk memprediksi digit yang digambar. Hasil prediksi akan ditampilkan di sebelah kanan kanvas dalam label dengan teks "Tulisan tersebut menghasil hijaiyah: [prediksi] dengan akurasi". Dengan ini, aplikasi menyediakan cara interaktif untuk mengenali tulisan tangan menggunakan model pembelajaran mesin yang telah dilatih.

Gambar 17. *Output* GUI

3.3 Pengujian Whitebox

a. Fungsi GUI

Pada tabel 1, dilaksanakan pengujian terhadap pernyataan GUI dengan menitikberatkan pada efektivitas skrip yang diterapkan. Hasil pengujian tersebut adalah sebagai berikut:

Tabel 1. Pengujian *Whitebox* GUI

No.	Pengujian	Test Case benar	Test Case salah	Kesimpulan
1.	Load dataset mnist.h5	Menghasilkan model dengan nilai bobot yang telah dioptimasi sesuai parameternya yang di load dari pengujian CNN	Verifikasi bahwa model dimuat tanpa error	Berhasil
2.	handle dari jendela Tkinter	Verifikasi bahwa handle yang dikembalikan adalah untuk kanvas Tkinter yang benar. Uji fungsi dengan berbagai skenario untuk memastikan handle yang tepat diperoleh.	Memastikan bahwa program tidak mengembalikan *handle* yang salah	Berhasil

3.	*Preprocessing* Gambar	fungsi *preprocessing_image* berperilaku dengan benar dalam kasus-kasus yang diharapkan dan tidak memberikan hasil yang tidak diinginkan.	Bahwa yang dimuat tidak terjadi error	Berhasil
4.	Prediksi Tulisan	respons fungsi terhadap gambar yang berisi digit yang jelas, seperti gambar mnist dari dataset. fungsi dapat memprediksi digit dengan tepat dan mengembalikan label yang sesuai dengan digit yang terdapat dalam gambar tersebut.	Memastikan program tidak terjadi seperti masukan gambar kosong, gambar dengan format tidak valid, ukuran gambar tidak valid, Model tidak terinisialisasi	Berhasil

5.	Antarmuka Pengguna	Semua elemen UI dibuat dan diatur dengan benar. Verifikasi bahwa kanvas, label, dan tombol berada di posisi yang benar.	Memastikan ketika menggambar di luar kanvas, tidak ada garis yang muncul di luar batas kanvas. Menekan tombol pengenalan tanpa adanya gambar tidak menghasilkan input atau pesan kesalahan. Menekan tombol hapus tanpa adanya gambar tidak menyebabkan perubahan apa pun pada kanvas.	Berhasil
6.	Menghapus semua gambar pada kanvas	Membuat instance dari kelas dengan canvas sebagai objek yang valid dan kemudian memanggil metode clear_all. Setelah pemanggilan metode, harus memverifikasi bahwa metode delete pada canvas benar-benar dipanggil dengan argumen "all". Menggunakan unittest dan mock untuk memastikan bahwa	Jika canvas adalah None, harus memastikan bahwa pemanggilan metode clear_all menghasilkan pengecualian AttributeError. jika canvas adalah objek yang tidak memiliki metode delete, harus memastikan bahwa pemanggilan metode clear_all menghasilkan pengecualian AttributeError.	Berhasil

		pemanggilan metode terjadi sebagaimana mestinya.		
7.	Klasifikasi tulisan	Verifikasi bahwa gambar diambil dari kanvas dengan koordinat yang benar. Dan gambar diprediksi dengan benar dan hasilnya ditampilkan di label.	Tidak terjadi ID canvas yang tidak valid menyebabkan kegagalan fungsi pada win32gui.GetWindowRect(HWND) atau menghasilkan prediksi yang tidak valid.	Berhasil
8.	Pengguna menggambar angka pada kanvas	Garis digambar dengan benar saat mouse bergerak di kanvas. fungsi draw_lines bekerja sesuai dengan ekspektasi dalam berbagai kondisi, baik dengan input yang valid maupun yang tidak valid.	Tidak terjadi menghasilkan AttributeError karena event tidak memiliki atribut x dan y atau TypeError atau ValueError karena x dan y bukan tipe data numerik	Berhasil

Berdasarkan tabel 4.1, dapat disimpulkan bahwa hasil pengujian *Whitebox* GUI berhasil dengan sukses, dimana seluruh pengujian berjalan sesuai dengan *test case* yang telah ditetapkan.

IV. KESIMPULAN DAN SARAN

4.1 Kesimpulan

Dalam penelitian ini, telah dikembangkan dan diuji sistem pengenalan tulisan tangan menggunakan *Convolutional Neural Network* (CNN) dengan antarmuka grafis pengguna (GUI) berbasis Tkinter. Hasil pengujian menunjukkan bahwa sistem mampu mengenali tulisan tangan dengan akurasi yang memadai. CNN terbukti efektif untuk klasifikasi gambar tulisan tangan, dengan langkah-langkah *pre-processing* yang meliputi konversi citra ke skala abu-abu, *thresholding*, dan ekstraksi kontur yang meningkatkan akurasi model. Pengujian *whitebox* terhadap beberapa fungsi kunci pada GUI menunjukkan bahwa implementasi berjalan sesuai harapan tanpa *error* yang berarti. Fungsi-fungsi utama seperti pemuatan model, pengambilan handle kanvas Tkinter, *pre-processing* gambar, dan prediksi digit berhasil diuji dengan hasil yang memuaskan.

4.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan dan diuji, peneliti menyimpulkan bahwa penelitian ini telah berlangsung sesuai dengan perencanaan. Namun, terdapat beberapa kelemahan dan aspek yang masih dapat dikembangkan. Oleh karena itu, peneliti memberikan beberapa saran yang mungkin bermanfaat bagi pembaca. Pertama, implementasi mekanisme penanganan *error* yang lebih komprehensif dalam kode akan membantu dalam identifikasi dan perbaikan masalah yang mungkin terjadi selama penggunaan sistem. Kedua, eksperimen lebih lanjut dengan berbagai arsitektur CNN dan *hyperparameter tuning* dapat dilakukan untuk menemukan konfigurasi yang lebih optimal. Pendekatan seperti data augmentation juga dapat diterapkan untuk meningkatkan performa model. Terakhir, untuk meningkatkan akurasi model lebih lanjut, disarankan untuk memperluas dataset dengan pengembangan pada huruf lain atau simbol yang lebih banyak variasi tulisan tangan. Hal ini akan membuat model lebih *robust* terhadap variasi gaya penulisan.

V. DAFTAR PUSTAKA

[1] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278-2324.

[2] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." *Advances in neural information processing*

systems, vol. 25, 2012.

[3] Graves, Alex, Abdel-rahman Mohamed, Geoffrey E. Hinton, and Navdeep Jaitly. "A novel connectionist system for unconstrained handwriting recognition." *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, 2009, pp. 855-868.

[4] Simard, Patrice Y., David Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." *Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, 2003, pp. 958-962.

[5] Masrani, dkk. "Aplikasi Pengenalan Pola pada Huruf Tulisan Tangan Menggunakan Jaringan Saraf Tiruan dengan Metode Ekstraksi Fitur Geometri." *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 4, no. 2, 2017, pp. 97-104.

[6] Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, vol. 1, MIT Press, 2016.

[7] E. Dina Juliani U M, I. G. P. S. Wijaya, and F. Bimantoro, "Pengenalan Pola Tulisan Tangan Suku Kata Aksara Sasak Menggunakan Metode Integral Projection dan Neural Network," J. Comput. Sci. Informatics Eng., vol. 3, no. 1, p. 19, 2019.

[8] D. C. K. Putu Aryasuta Wicaksana, I Made Sudarma, "Pengenalan Pola Motif Kain Tenun Gringsing Menggunakan Metode Convolutional Neural Network Dengan Model Arsitektur," *J. SPEKTRUM*, vol. 6, no. 3, pp. 159–168, 2019.

[9] A. N. Lorentius, Christopher Albert; Gunadi, Kartika; Tjondrowiguno, "Pengenalan Aksara Jawa dengan Menggunakan Metode Convolutional Neural Network," *J. Infra Petra*, vol. 7, no. 1, pp. 221–227, 2019.

[10] R. Yulianti, I. G. P. S. Wijaya, and F. Bimantoro, "Pengenalan Pola Tulisan Tangan Suku Kata Aksara Sasak Menggunakan Metode Moment Invariant dan Support Vector Machine," *J. Comput. Sci. Informatics Eng*., vol. 3, no. 2, 2019.

[11] Mufassiril abror and N. Nopiyanto, "Pattern Recognition Tulisan Tangan Huruf Hijaiyah Menggunakan Metode Convolutional Neural Network (CNN)", *Jurnal Informasi dan Komputer*, vol. 9, no. 2, pp. 174-178, Oct. 2021.

[12] E. Gunawan, Verrell; Gunadi, Kartika; Setyati, "Identifikasi Buah-buahan Menggunakan Metode Convolutional Neural Network," *J. Infra Petra*, vol. 7, no. 1, pp. 33–38, 2019.

[13] Herlina, "Pengenalan Tulisan Tangan pada Lembar Ujian Pilihan Ganda Menggunakan Metode Convolutional Neural Network," *Jurnal Ilmiah Pendidikan Fisika Al-Biruni*, vol. 8, no. 2, pp. 189-198, 2019.

[14] Wijaya, E. T., & Farqi, I. W. Al. Aplikasi Pengenalan Aksara Carakan Madura Dengan Menggunakan Metode Backpropagation. Jurnal Ilmiah Teknologi Informasi Asia, 9(1), 18–34. 2015.

[15] Agustini and W. J. Kurniawan, "Sistem E-Learning Do'a dan Iqro' dalam Peningkatan Proses Pembelajaran pada TK Amal Ikhlas," *J. Mhs. Apl. Teknol. Komput. dan Inf.*, vol. 1, no. 3, pp. 154–159, 2019, [Online]. Available: http://www.ejournal.pelitaindonesia.ac.id/JMApTeKsi/index.php/JOM/article/view/526

[16] L. Perkovic, Introduction to Computing Using Python: An Application Development Focus. Hoboken, NJ: John Wiley & Sons, 2012.

[17] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2008.

[18] F. M. Ardi, K. N. Ramadhani, dan A. Arifianto, "Pengenalan Angka Tulisan Tangan Menggunakan Diagonal Feature Extraction dan Klasifikasi Artificial Neural Network Multilayer Perceptron", Fakultas Informatika. Universitas Telkom, Vol.3, 2018.

[19] R. Messina, Ronaldo, dan Jerome Louradour, "*Segmentation-free Handwritten Chinese Text Recognition with* LSTM-RNN", *International Conference on Document Analysis and Recognition* (ICDAR), 2015.