Seminar Nasional Sains Data 2025 (SENADA 2025)
UPN "Veteran" Jawa Timur
E-ISSN 2808-5841
P-ISSN 2808-7283

Sistem Pengenalan Kata Isyarat Tangan Berbasis CNN dengan Integrasi *Hadoop Storage* dan Antarmuka GUI Python

Abdurrahman Al-atsary[1], M. Akbar Resdika[2], Putri Olivia Nuraeni[3], Luluk Muthoharoh[4], Ardika Satria[5], Rizty Maulida Badri[6]
[1,2,3,4,5,6,]*Afiliasi pertama (Sains/Sains Data, Institut Teknologi Sumatera)*
[1]abdurrahman.121450128@student.itera.ac.id
[2]makbar.121450066@student.itera.ac.id
[3]putri.121450009@student.itera.ac.id
[4]luluk.muthoharoh@sd.itera.ac.id
[5]ardika.satria@sd.itera.ac.id
[6]riztymaulidabadri@gmail.com
Corresponding author email: luluk.muthoharoh@sd.itera.ac.id

Abstract: *Communication between people with disabilities and the general public is often hampered due to the limited use of sign language. This research develops a Convolutional Neural Network (CNN)-based hand signal word recognition system using TensorFlow, which is integrated with the Hadoop Distributed File System (HDFS) and a Python GUI-based graphical interface. Data in the form of American Sign Language (ASL) images are processed through pre-processing, augmentation, and training stages using CNN architecture. The developed model achieved 99% validation accuracy with a loss value of 0.0196. The system also features an automated pipeline for efficient data upload to Hadoop. The confusion matrix evaluation results achieved accuracy, precision, recall and F1-Score above 98% in most classes, so it has the potential as a system to improve communication accessibility for people with disabilities.*
Keywords: *Apache Hadoop, Disabilities, GUI Python, Hand Sign, Model Integration*

Abstrak: Komunikasi antara penyandang disabilitas dan masyarakat umum seringkali terhambat akibat keterbatasan penggunaan bahasa isyarat. Penelitian ini mengembangkan sistem pengenalan kata isyarat tangan berbasis *Convolutional Neural Network* (CNN) menggunakan TensorFlow, yang terintegrasi dengan *Hadoop Distributed File System* (HDFS) serta antarmuka grafis berbasis Python GUI. Data berupa gambar *American Sign Language* (ASL) diproses melalui tahap pra pengolahan, augmentasi, dan pelatihan menggunakan arsitektur CNN. Model yang dikembangkan mencapai akurasi validasi sebesar 99% dengan nilai loss sebesar 0,0196. Sistem juga dilengkapi pipeline otomatis untuk pengunggahan data ke *Hadoop* secara efisien. Hasil evaluasi *confusion matrix* mencapai akurasi, presisi, recall dan F1-Score di atas 98% pada sebagian besar kelas, sehingga berpotensi sebagai sistem untuk meningkatkan aksesibilitas komunikasi bagi penyandang disabilitas.
Kata kunci: *Apache Hadoop*, Disabilitas, Isyarat Tangan, Integrasi Model, Python GUI

I. PENDAHULUAN

Manusia merupakan makhluk sosial yang dapat berkomunikasi, berinteraksi, serta bersosialisasi dengan sesama manusia. Bahasa menjadi sarana utama komunikasi antarmanusia. Namun, tidak semua individu mampu berkomunikasi secara verbal, seperti penyandang disabilitas tunarungu dan tunawicara [1]. Bahasa isyarat kemudian menjadi media alternatif utama dalam menyampaikan pesan, meskipun pemahaman masyarakat umum terhadap bahasa isyarat masih rendah sehingga menimbulkan hambatan komunikasi [1], [2]. Oleh karena itu, penyandang disabilitas umumnya hanya dapat berkomunikasi menggunakan bahasa isyarat [1], [2].

Bahasa isyarat sendiri menggunakan gerakan tubuh seperti tangan dan wajah untuk menyatakan istilah atau kata, dan setiap negara memiliki standar berbeda. Di tingkat internasional, *American Sign Language* (ASL) menjadi bahasa isyarat yang banyak digunakan, khususnya oleh penyandang tunarungu di Amerika Utara, dan sering menjadi fokus pengembangan sistem pengenalan berbasis komputer karena strukturnya yang visual dan cocok diimplementasikan dengan teknik computer

vision seperti *Convolutional Neural Network* (CNN) [6]–[9]. Sementara itu, di Indonesia terdapat Bahasa Isyarat Indonesia (BISINDO) dan Sistem Isyarat Bahasa Indonesia (SIBI). SIBI merupakan bentuk bahasa isyarat baku yang diseragamkan dan umumnya digunakan di konteks formal [2].

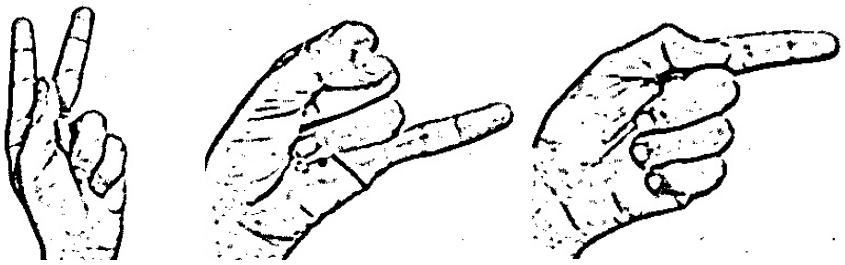
Teknologi pengenalan citra, khususnya CNN, terbukti efektif dalam mendeteksi dan mengenali pola visual karena kemampuannya mengekstraksi fitur spasial dari citra [6]–[9]. Seiring meningkatnya volume data, tantangan pengelolaan big data menjadi lebih kompleks, sehingga dibutuhkan sistem penyimpanan terdistribusi seperti *Hadoop Distributed File System* (HDFS) untuk menangani penyimpanan dan pemrosesan data skala besar secara paralel dan efisien [21], [22].

Pemanfaatan teknologi ini membuka peluang mempermudah komunikasi antara penyandang disabilitas dan masyarakat umum melalui penerapan *computer vision* untuk pengenalan bahasa isyarat [2],[3],[6]–[9]. Penelitian ini bertujuan membangun sistem deteksi bahasa isyarat tangan berbasis CNN yang terintegrasi dengan Hadoop dan dilengkapi antarmuka *Graphical User Interface* (GUI). Sistem memanfaatkan pipeline CNN menggunakan TensorFlow untuk mengenali kata dan huruf ASL, serta pengelolaan data melalui *Hadoop Distribution* guna mendukung penyimpanan dan pengolahan data berskala besar [21], [22]. Sistem ini juga dirancang dapat diakses melalui laptop atau komputer dengan kamera, sehingga diharapkan mempermudah komunikasi dan menciptakan interaksi yang lebih inklusif.

II. METODE

II.1. Pengambilan dan Pemrosesan Data

Pada penelitian ini digunakan dataset publik yang diperoleh dari website Kaggle[1], berisi citra tangan dalam format berwarna dan kondisi mentah. Sebelum digunakan dalam tahap pemodelan, citra-citra tersebut terlebih dahulu melalui proses *preprocessing* yang mencakup konversi gambar ke format *grayscale* serta *binarization*. Tahapan ini bertujuan untuk menyederhanakan representasi visual citra dengan mengurangi jumlah kanal warna dan memisahkan objek utama dari latar belakang, sehingga dapat memperkecil dimensi data dan kompleksitas komputasi. Dengan demikian, model yang dikembangkan dapat lebih cepat dan efisien dalam mengenali serta memprediksi pola gerakan tangan. Beberapa sampel dari hasil *preprocessing* dapat dilihat dari gambar berikut:



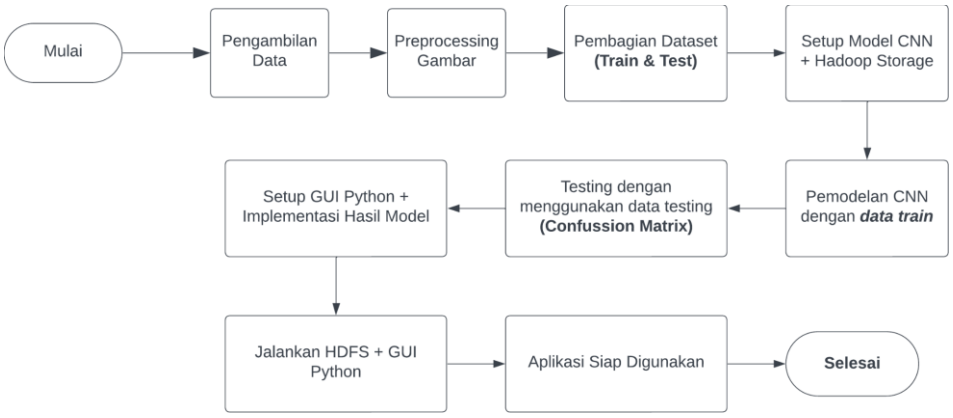
Gambar 1. Hasil *Preprocessing* pada gambar tangan

II.2. Diagram Alur Penelitian

Pada penelitian ini, tahapan metode dirancang secara sistematis dan divisualisasikan melalui sebuah flowchart. Flowchart ini berfungsi untuk menggambarkan alur proses penelitian secara keseluruhan, mulai dari tahap awal pengumpulan data hingga proses evaluasi hasil. Dengan adanya

[1] **Link dataset:** https://www.kaggle.com/datasets/grassknoted/asl-alphabet

flowchart, diharapkan pembaca dapat lebih mudah memahami setiap langkah yang dilakukan dalam penelitian ini serta hubungan antar tahapannya.



Gambar 2. Diagram Alir Penelitian

Diagram alir pada penelitian ini menggambarkan alur kerja mulai dari tahap pengambilan data hingga aplikasi siap digunakan. Proses diawali dengan pengambilan data yang kemudian melalui tahap preprocessing gambar untuk memudahkan pemodelan dengan mengubah menjadi bentuk binerisasi untuk bentuk telapak tangan dari dataset yang dikumpulkan. Langkah selanjutnya, melakukan pembagian menjadi data *train* dan *test* sebelum masuk tahapan pada pemodelan dengan CNN (*Convolutional Neural Network*). Model CNN dibangun dan dikombinasikan dengan Hadoop Storage untuk memaksimalkan pengelolaan data skala besar. Proses pemodelan dilakukan menggunakan data *train* yang mendapatkan augmentasi ringan seperti rotasi sebesar20^{o}, proses setelah melakukan pemodelan, yakni dilakukan evaluasi performa model dilakukan pada data testing dengan menggunakan *confusion matrix*. Setelah itu, hasil model diintegrasikan ke dalam antarmuka berbasis GUI Python agar mudah digunakan oleh pengguna dan siap untuk digunakan dalam lingkup masyarakat. Tahap akhir meliputi pengujian dan menjalankan HDFS bersama GUI Python hingga aplikasi siap digunakan secara utuh.

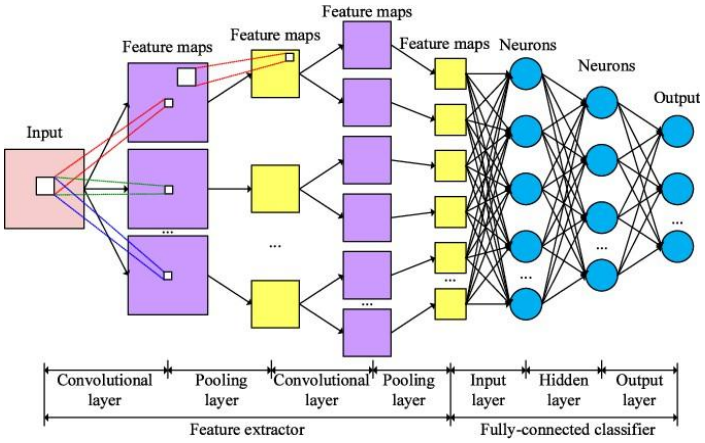
II.3. CNN *(Convolutional Neural Network)*

Convolutional Neural Network (CNN) merupakan salah satu arsitektur deep learning yang dirancang khusus untuk mengolah data berbentuk grid, seperti citra dua dimensi. CNN bekerja dengan mengekstraksi fitur spasial melalui operasi konvolusi, sehingga mampu mengenali pola penting dalam data visual, seperti garis tepi, tekstur, dan bentuk kompleks. Mekanisme kerja CNN dimulai dengan proses konvolusi menggunakan filter (kernel) berukuran tertentu, misalnya 3×3. Filter ini melintasi seluruh area citra input dan menghasilkan feature maps yang memuat ciri-ciri penting. Dua parameter utama pada proses ini adalah *stride* dan *padding*. Stride mengatur jumlah pergeseran filter saat berpindah posisi, sedangkan padding adalah penambahan piksel (biasanya nol) di tepi citra input agar ukuran output tetap sama dengan input, sehingga informasi di pinggir citra tidak hilang[8].

Setelah layer konvolusi, diterapkan fungsi aktivasi seperti ReLU (*Rectified Linear Unit*) yang berfungsi menambah non-linearitas pada model, memungkinkan CNN mempelajari pola yang lebih kompleks[8]. Selanjutnya, *pooling layer* biasanya menggunakan metode max pooling untuk

mereduksi dimensi *spasial feature maps* dengan mengambil nilai maksimum dari *patch* berukuran tertentu, seperti 2×2. Proses ini membantu mengurangi jumlah parameter dan meningkatkan ketahanan model terhadap pergeseran kecil pada citra [7], [8]. Arsitektur CNN umumnya menyusun beberapa layer konvolusi dan *pooling* secara bergantian agar model memiliki pemahaman lebih mendalam terhadap data. Setelah tahap ekstraksi fitur selesai, data hasil *pooling* yang masih berbentuk matriks diubah menjadi vektor satu dimensi melalui flattening layer[10].

Tahap akhir CNN adalah *fully connected layer*, yang menghubungkan seluruh neuron hasil *flattening* ke neuron output. *Fully connected layer* berfungsi melakukan klasifikasi berdasarkan fitur yang telah diekstraksi [7], [8]. Proses pelatihan CNN dilakukan menggunakan algoritma optimasi seperti Adam atau SGD, dengan tujuan memperbarui bobot dan bias agar nilai *loss* minimal dan akurasi meningkat [7], [9],[13] .



Gambar 3. Proses model CNN (*Input–Output*)

II.4. Fungsi Kerugian (*Categorical Cross-Entropy*)

Cross-Entropy Loss merupakan fungsi loss yang paling umum digunakan untuk tugas klasifikasi, karena mampu mengukur perbedaan antara distribusi probabilitas yang diprediksi model dengan distribusi target sebenarnya [7][8]. Fungsi ini menghitung *log loss* dari probabilitas yang diprediksi untuk kelas benar, lalu mengakumulasikannya untuk seluruh kelas. Dengan demikian, *Cross-Entropy Loss* memberikan penalti lebih besar terhadap prediksi yang jauh dari target, sehingga mendorong model untuk meningkatkan akurasi [7][8].

$$\text{CCE Loss} = -\frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \cdot log(p_{ij}) \quad (1)$$

Selain itu, *Cross-Entropy Loss* sering dikombinasikan dengan aktivasi *softmax* agar output dapat diinterpretasi sebagai probabilitas yang sah [7]. Sifat konveks dari fungsi ini membantu proses pelatihan model menjadi lebih stabil dan konvergen lebih cepat dibandingkan fungsi loss lainnya, sehingga menjadi standar utama dalam pelatihan model klasifikasi [7][8].

II.5. Optimizer Adam

Adam (*Adaptive Moment Estimation*) adalah algoritma optimasi yang menggabungkan keunggulan dari Momentum dan RMSProp untuk menghasilkan proses pelatihan yang lebih cepat dan stabil. Adam bekerja dengan menyimpan estimasi rata-rata gradien (momen pertama) dan rata-rata kuadrat gradien (momen kedua), kemudian melakukan koreksi bias agar estimasi tersebut tidak terlalu kecil di awal pelatihan. Pendekatan ini memungkinkan Adam untuk menggunakan *learning rate* yang disesuaikan secara adaptif untuk setiap parameter, sehingga dapat menangani data yang bervariasi skala gradiennya maupun model yang kompleks seperti *deep neural networks*.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\left[\frac{\delta L}{\delta w_t}\right]$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2 \quad (2)$$

Keunggulan utama Adam terletak pada kemampuannya mempercepat konvergensi tanpa memerlukan banyak penyesuaian hyperparameter, serta performa yang robust meskipun data yang digunakan memiliki gradien yang jarang atau fluktuasi besar[11]. Karena itu, Adam menjadi salah satu algoritma optimasi paling populer dalam *deep learning modern*, baik untuk data berukuran besar maupun model dengan kedalaman arsitektur yang tinggi[12].

II.6. *Hadoop Distributed File System (HDFS)*

Hadoop Distributed File System (HDFS) adalah sistem penyimpanan terdistribusi yang menjadi inti dari ekosistem *Hadoop*, dirancang khusus untuk menyimpan dan mengelola data berukuran besar secara efisien. HDFS membagi file besar menjadi blok-blok data yang tersebar ke berbagai node (*datanode*) dengan mekanisme replikasi otomatis, sehingga data tetap aman dan tersedia meskipun terjadi kegagalan pada sebagian node[20][4]. Dengan pendekatan ini, HDFS mampu mendukung pemrosesan data secara paralel dan menjaga keandalan sistem dalam skala besar.

Dalam penelitian ini, HDFS dimanfaatkan sebagai media penyimpanan utama untuk dataset yang akan diperoleh dari hasil deteksi gerakan tangan oleh pengguna. Untuk memudahkan pengguna mengunggah file ke HDFS, dibangun dua jenis *pipeline*: *pipeline* otomatis yang menangani unggahan secara terprogram, serta pipeline berbasis GUI yang memungkinkan pengguna memilih file dari lokal dan langsung mengunggahnya ke HDFS melalui antarmuka grafis [21]. Pendekatan ini tidak hanya membuat sistem lebih mudah digunakan oleh pengguna awam, tetapi juga memastikan bahwa data besar dapat dikelola secara terstruktur dan terdistribusi.

II.7. Spesifikasi Device

Perangkat yang digunakan dalam penelitian ini memiliki spesifikasi untuk mendukung proses pengembangan dan pengujian sistem. Pada proses pelatihan model, device yang digunakan berasal dari komputasi awan (*cloud computing*) melalui platform Kaggle, dengan memanfaatkan GPU NVIDIA Tesla P100 yang memungkinkan percepatan proses pelatihan selama kurang lebih 2 jam. Sementara itu, *Hadoop Storage* (HDFS) diinisialisasi pada VirtualBox yang dijalankan secara lokal untuk menyediakan server yang dapat diakses melalui *localhost* maupun dikoneksikan ke server berskala lebih besar sesuai kebutuhan.

Untuk keperluan pengembangan aplikasi GUI berbasis Python serta uji coba pipeline, digunakan spesifikasi prosesor Intel Core i7 generasi ke-9, RAM sebesar 8 GiB, serta GPU NVIDIA GTX 1650 dengan memori VRAM 4 GiB. Spesifikasi ini dinilai cukup untuk menangani visualisasi data, menjalankan skrip Python, serta mendukung komunikasi dengan *Hadoop* melalui pustaka python seperti *InsecureClient*. Dengan konfigurasi perangkat keras tersebut, proses pengujian *pipeline* otomatis dan *pipeline* berbasis GUI dapat berjalan lancar tanpa kendala berarti.

III. HASIL DAN PEMBAHASAN

III.1. Membangun Model CNN

Arsitektur model CNN dirancang dengan melakukan konfigurasi terhadap parameter- parameter yang digunakan. Parameter yang digunakan pada model CNN meliputi 2 lapisan konvolusi (Conv2D) sebagai lapisan dasar ekstraksi fitur, 2 lapisan *Max Pooling* 2D sebagai bentuk representasi dari pengurangan dimensi dan seleksi fitur dominan dari pemetaan fitur , lapisan *flatten* sebagai perataan input layer sebelum masuk ke lapisan *Fully Connected*, Lapisan *Fully Connected* sebagai pelatihan untuk fitur yang akan diklasifikasikan dan terakhir lapisan dropout sebagai regularisasi terhadap *Fully Connected Layer* yang berguna sebagai mekanisme *switch neuron* agar hasil lebih presisi dan tidak didominasi pelatihan pada salah satu jalur neuron saja[5]. Fungsi aktivasi yang digunakan pada model CNN dalam penelitian ini meliputi *Rectified Linear Unit* (ReLu) dan fungsi aktivasi *softmax*. Parameter-parameter yang digunakan secara spesifik dapat ditinjau pada Tabel 1 sebagai berikut,

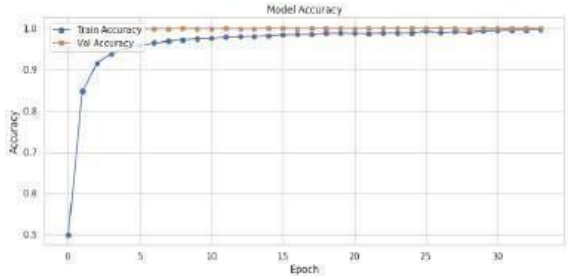
Tabel 1. Parameter dan Arsitektur Model *Deep Neural Network*

Layer	Output Shape	Params (#)
Conv2D	(None, 128, 128, 32)	320
MaxPool2D	(None, 64, 64, 32)	0
Conv2D	(None, 64, 64, 32)	9,248
MaxPool2D	(None, 32, 32, 32)	0
Flatten	(None, 30752)	0
Dense	(None, 128)	3,936,384
Dropout	(None, 128)	0
Dense	(None, 96)	12,384
Dropout	(None, 96)	0
Dense	(None, 64)	6,208
Dense	(None, 27)	1,755

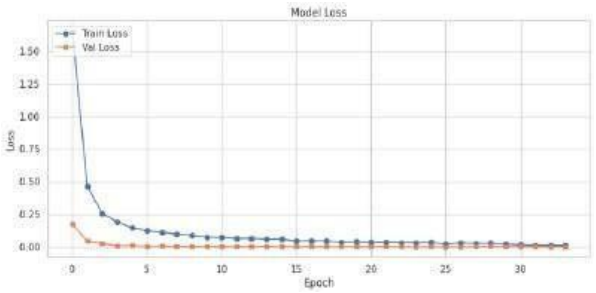
III.2. Pelatihan Model

Pelatihan model dilakukan dengan menggunakan server kaggle dengan ketentuan dari hyperparameter *batch size* yang berjumlah 32 dengan tujuan untuk mempercepat inisialisasi model yang akan dilatih kemudian dari segi pembagian dataset data dibagi menjadi 80% untuk data latih (*train*) dan 20% sebagai data validasi (*testing*). Kemudian, dilakukan pelatihan model menggunakan optimizer Adam dengan *learning rate* default, yakni 10^{-5} dan dilakukan pelatihan model selama 100 epoch dengan ketentuan adanya mekanisme *early stopping* dan *Checkpoint Model* yang memiliki tujuan secara berurutan mencegah overfit dari model dan menyimpan bobot terbaik selama pelatihan yang menjadi *base* terhadap penilaian bobot dipertahankan.

Berdasarkan Gambar 3a dan Gambar 3b, terlihat bahwa model *Convolutional Neural Network* (CNN) yang dibangun menunjukkan performa yang sangat baik selama proses pelatihan dan validasi. Gambar 3a menampilkan grafik akurasi yang menunjukkan bahwa akurasi data pelatihan mengalami peningkatan tajam pada epoch awal dan kemudian stabil mendekati 100%. Akurasi data validasi juga mengikuti pola serupa, dengan peningkatan yang konsisten dan mendekati nilai akurasi pelatihan. Hal ini menunjukkan bahwa model memiliki kemampuan generalisasi yang tinggi dan tidak mengalami *overfitting*, karena akurasi pada data validasi tidak tertinggal jauh dari data pelatihan. Sementara itu, Gambar 3b memperlihatkan grafik *loss*, di mana baik nilai *loss* pada data pelatihan maupun validasi menurun drastis di awal pelatihan dan kemudian terus menyusut hingga mendekati nol. Tidak terdapat lonjakan atau peningkatan nilai *loss* pada data validasi, yang umumnya merupakan indikator *overfitting*. Keseluruhan hasil ini mengindikasikan bahwa proses pelatihan berjalan stabil, dan model mampu belajar serta menggeneralisasi pola dengan sangat baik. Hal ini sesuai dengan teori yang dikemukakan oleh Salma et al. yang menyatakan bahwa model *deep learning*, khususnya CNN, dapat mencapai akurasi tinggi dalam klasifikasi data visual apabila arsitektur dan parameter yang digunakan sesuai[14]. Dengan demikian, grafik akurasi dan *loss* ini memberikan bukti kuat bahwa model yang dikembangkan cukup optimal dan siap untuk digunakan dalam prediksi data baru dengan akurasi tinggi[15].



Gambar 4a. Akurasi Pelatihan



Gambar 4b. *Loss* Pelatihan

Gambar 4. Grafik hasil data pelatihan dan data validasi

III.3. Evaluasi Model

Berdasarkan hasil evaluasi terhadap model pada Tabel 2, model menunjukkan performa yang sangat baik dengan seluruh matrik evaluasi melebihi 98% pada keseluruhan kelas karakter. Model yang dilatih dengan menggunakan Optimizer Adam dengan *learning rate* sebesar 10^{-5} menunjukkan hasil dari model mencapai akurasi validasi sebesar 94.24% dan *loss* sebesar 0.3096. Hasil ini

menunjukkan bahwa arsitektur yang dibangun mampu belajar dengan baik dari data. Evaluasi terhadap masing-masing kelas menunjukkan akurasi di atas 98%, namun perlu disertai dengan visualisasi *confusion matrix* untuk memastikan distribusi kesalahan antar kelas. Nilai akurasi tertinggi dicapai oleh kelas A dengan akurasi sebesar 98.76%, sedangkan akurasi terendah pada kelas D yaitu sebesar 98.45%. Hasil evaluasi ini menunjukkan bahwa model mampu mengenali pola isyarat dengan baik dan tingkat kesalahan yang sangat rendah. Integrasi dengan GUI Python dan pipeline *Hadoop* menunjukkan fleksibilitas sistem untuk digunakan baik oleh pengguna teknis maupun non-teknis. Pengujian sistem pipeline menunjukkan waktu unggah data yang efisien dengan latensi rendah (<1s untuk file <1MB), namun belum diuji untuk skenario streaming data.

Tabel 2. Evaluasi Model dengan *Confusion Matrix*

Kelas	Akurasi (%)	Presisi (%)	Recall (%)	F1-score (%)
A	98.76	98.92	98.65	98.78
B	98.53	98.71	98.44	98.57
C	98.62	98.80	98.51	98.65
D	98.45	98.61	98.33	98.47
E	98.71	98.89	98.60	98.74
F	98.59	98.78	98.47	98.62
G	98.60	98.75	98.48	98.61
H	98.68	98.84	98.55	98.69
I	98.50	98.66	98.39	98.52
J	98.55	98.73	98.44	98.58
K	98.63	98.81	98.52	98.66
L	98.72	98.90	98.62	98.76

E-ISSN 2808-5841
P-ISSN 2808-7283

M	98.58	98.74	98.45	98.59
N	98.64	98.83	98.53	98.67
O	98.49	98.67	98.36	98.51
P	98.61	98.78	98.49	98.63
Q	98.57	98.76	98.46	98.60
R	98.66	98.85	98.55	98.69
S	98.70	98.87	98.59	98.72
T	98.60	98.77	98.48	98.61
U	98.65	98.83	98.53	98.66
V	98.53	98.70	98.42	98.55
W	98.62	98.79	98.50	98.63
X	98.56	98.74	98.45	98.58
Y	98.69	98.86	98.58	98.70
Z	98.60	98.78	98.47	98.61
0	98.67	98.84	98.55	98.68

III.4. Pembuatan Pipeline GUI Python dan HDFS

Data *pipeline* merupakan alur kerja yang dimulai dari proses pengumpulan, pengolahan, hingga penyimpanan data ke dalam sistem tertentu, baik itu penyimpanan lokal maupun sistem terdistribusi seperti *Hadoop*. Pipeline ini dirancang agar data yang telah diproses dapat disimpan dan digunakan

kembali dalam proses analisis kata yang sering muncul untuk meningkatkan efektivitas dari sistem atau dipergunakan untuk pengembangan model kedepannya. Dalam penelitian ini, dibangun dua jenis *pipeline*, yaitu pipeline berbasis otomatisasi dan *pipeline* berbasis antarmuka grafis pengguna (GUI) menggunakan pustaka *Tkinter*. Tujuan utama dari pipeline ini adalah untuk mengirimkan data hasil deteksi isyarat tangan ke sistem *Hadoop* secara efisien.

III.4.1 Pipeline Otomatis

Pipeline otomatis ini dibuat untuk memudahkan pengiriman data dari komputer lokal ke server Hadoop, tanpa memerlukan interaksi manual dari pengguna. Pipeline dibangun dalam bentuk kelas Python yang memiliki dua fungsi utama.

Pertama, fungsi inisialisasi yang mengatur koneksi ke *Hadoop* dengan menggunakan informasi seperti alamat *server, port,* dan nama pengguna. Kedua, fungsi untuk mengunggah file, yang akan secara otomatis memindahkan data hasil deteksi dari penyimpanan lokal ke direktori yang dituju di *Hadoop* (HDFS). Dengan pendekatan ini, proses transfer data dapat berjalan lancar melalui port lokal (8080), sehingga pengguna tidak perlu melakukan langkah tambahan saat mengirimkan data ke Hadoop.

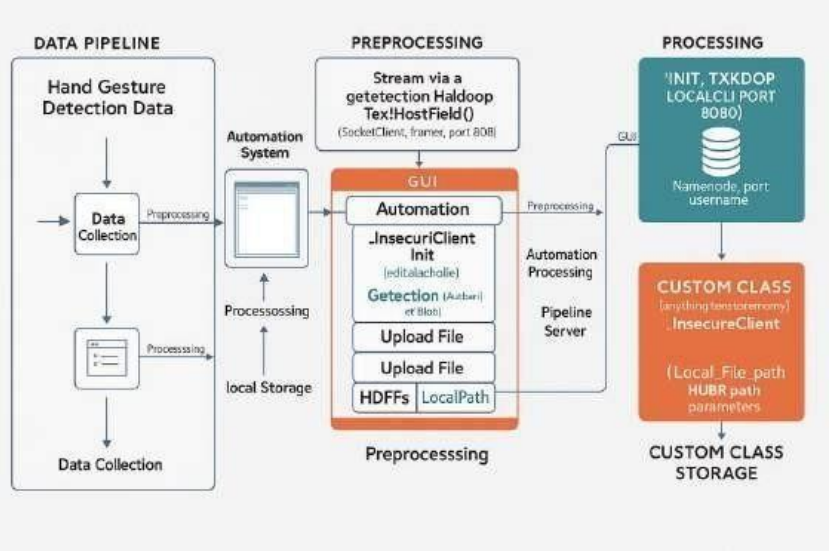
III.4.2 Pipeline Manual (AntarMuka GUI Python)

Selain pipeline otomatis, juga dikembangkan pipeline berbasis antarmuka grafis (GUI) untuk memudahkan pengguna dalam memilih dan mengunggah file ke *Hadoop*. Antarmuka ini dibuat menggunakan pustaka Tkinter dan tetap terhubung ke Hadoop melalui *InsecureClient*, sehingga pengguna dapat berinteraksi dengan sistem secara lebih intuitif.

Dalam sistem GUI ini terdapat dua fungsi utama. Pertama, fungsi untuk menampilkan data, yang memungkinkan pengguna memilih file CSV dari komputer lokal dan langsung melihat isinya di layar. Kedua, fungsi untuk mengunggah file terpilih ke server *Hadoop*, lengkap dengan pengaturan lokasi penyimpanan di HDFS. Tampilan GUI dilengkapi label sebagai keterangan, area teks untuk menampilkan isi file, serta tombol yang dapat diklik untuk memulai proses unggah. Jendela aplikasi ini berukuran 800 x 400 piksel dan diberi judul "*Hadoop File Upload*". Pendekatan gabungan antara otomatisasi dan GUI ini membantu mempercepat integrasi data hasil deteksi ke *Hadoop*, sehingga proses pengolahan data menjadi lebih mudah, efisien, dan dapat digunakan baik oleh pengguna teknis maupun non-teknis

III.4.3 Fungsional GUI Python dengan Pipeline

Pemrosesan dari data dapat diilustrasikan pada Gambar 4, pada awalnya model akan berjalan dengan GUI Python dan akan menerima data input secara langsung oleh pengguna, dari record pengguna kemudian dilanjutkan pada tahapan streaming datanya ke server dari *Hadoop* pada localhost atau server yang tersedia, data dapat di automasi dari masuknya secara otomatis atau manual yang nantinya data akan mengisi *node storage* pada HDFS yang kemudian datanya dapat dipergunakan untuk pemrosesan selanjutnya sebagai bahan analisis, pemodelan dengan struktur kata maupun tujuan lainnya



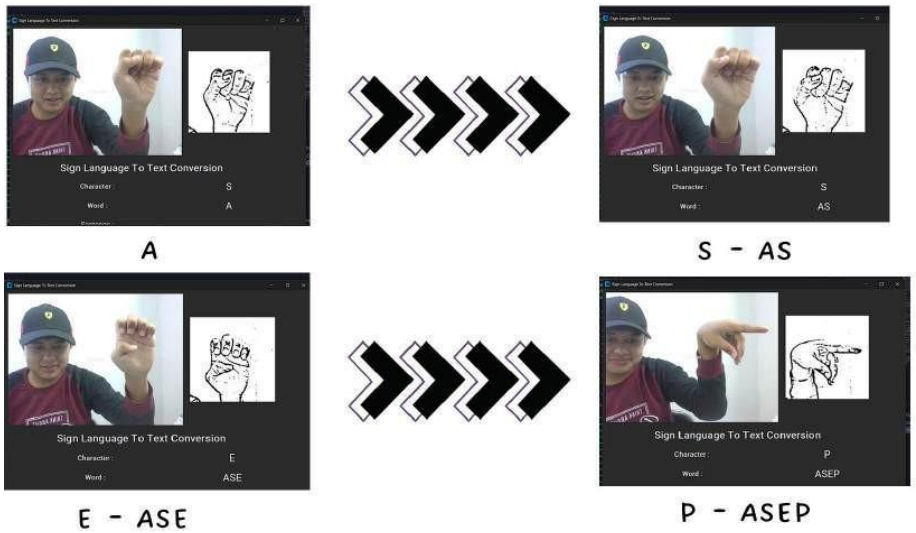
Gambar 5. Fungsional GUI Python (Model dengan *Hadoop Storage*)

III.5. Penerapan Model Secara Real-Time

Model deteksi isyarat tangan dikembangkan dengan pendekatan berbasis video yang terintegrasi dalam antarmuka grafis (GUI). Proses diawali dengan menginisialisasi model CNN yang telah dirancang untuk mengonversi gerakan isyarat tangan menjadi teks. Selanjutnya, dilakukan pengaturan tampilan antarmuka seperti judul, ukuran jendela, serta tata letak tombol dan area tampilan video dan luas dari area deteksi yang berada di atas kiri dari layar dengan ukuran 50x50 px. Sistem kemudian memproses video secara real time dengan membaca setiap frame, mengubah format warna agar sesuai dengan kebutuhan tampilan, dan menghasilkan prediksi teks dari input isyarat tangan. Apabila input tidak sesuai atau tidak terdeteksi dalam data latih, maka sistem akan menampilkan hasil kosong. Untuk meningkatkan akurasi hasil prediksi, diterapkan tahap pemeriksaan ejaan sehingga mendekati kata yang paling relevan. Sebelum aplikasi ditutup, dilakukan pembersihan data sementara dan penyimpanan hasil proses yang telah dijalankan. Proses penutupan antarmuka dilakukan secara terstruktur untuk memastikan aplikasi berhenti dengan aman dan data yang diperlukan tetap tersimpan.

Gambar 5. memperlihatkan hasil implementasi sistem pengenalan kata isyarat tangan secara berurutan dalam menyusun satu kata utuh, yaitu "ASEP". Setiap gambar menunjukkan proses klasifikasi karakter individu—dimulai dari huruf "A", "S", "E", hingga "P"—yang masing-masing dikenali oleh model CNN dan ditampilkan dalam antarmuka GUI. Dengan setiap input gestur baru, sistem secara otomatis memperbarui susunan huruf yang membentuk kata, yang menandakan keberhasilan integrasi antara deteksi karakter dan penyusunan kata secara real-time. Fenomena ini menunjukkan bahwa sistem memiliki kemampuan untuk mempertahankan konteks input sebelumnya, sebuah karakteristik penting dalam interaksi manusia-komputer yang berkelanjutan. Berdasarkan teori *Recurrent Inference Patterns*[15], model yang berinteraksi mendukung kontinuitas input-output akan lebih meningkatkan pengalaman pengguna dan efektivitas komunikasi, terutama dalam aplikasi yang bersifat *sequential* seperti bahasa isyarat.

Selain itu, keberhasilan antarmuka GUI dalam menampilkan hasil klasifikasi secara bertahap memperkuat prinsip *incremental feedback* yang menjadi bagian penting dalam sistem AI yang berorientasi pada manusia [16][17]. Dalam konteks ini, pengguna tidak hanya melihat hasil akhir, tetapi juga memahami progres klasifikasi karakter, sehingga memberikan transparansi dan kemudahan pemahaman bagi pengguna umum. Kombinasi antara kemampuan teknis model CNN dalam mengekstraksi fitur visual dari citra dan desain antarmuka yang ramah pengguna menciptakan sistem yang tidak hanya efisien, tetapi juga inklusif dan aplikatif dalam konteks edukasi dan interaksi sosial penyandang disabilitas[18][19].



Gambar 6. Penerapan model dalam pengenalan kata isyarat tangan

IV. KESIMPULAN

Penelitian ini berhasil mengembangkan sistem pengenalan kata isyarat tangan menggunakan model CNN yang terintegrasi dengan *GUI Python* dan penyimpanan *Hadoop*. Model menunjukkan performa tinggi dalam mengenali gestur tangan berdasarkan data ASL, dengan akurasi diatas 98% pada sebagian besar kelas. Pipeline otomatis dan antarmuka GUI mendukung efisiensi penyimpanan dan aksesibilitas pengguna. Pengembangan aplikasi ini kedepannya dapat menjadi sistem untuk mendukung deteksi *real-time* berbasis server dan dapat diintegrasikan pada perangkat *mobile* agar lebih inklusif dan aplikatif untuk masyarakat umum.

UCAPAN TERIMA KASIH

Terima kasih disampaikan kepada Tim SENADA yang telah berkontribusi dalam penyusunan dan penyediaan template ini.

REFERENSI

[1] Y. Octaviani and Y. Yuningsih, "Kemampuan Interaksi Sosial Tunarungu di Kelurahan

Batununggal Kota Bandung," *J. Ilmu Kesejahteraan Sosial*, vol. 1, no. 2, pp. 115–134, 2020, [Online]. Available: http://www.journal.unpas.ac.id/index.php/humanitas/article/view/1919

[2] P. W. Aditama, I. Gede, A. S. Anggara, and N. Jayanegara, "Implementation of Sibi And Bisindo Letters Recognition Using Augmented Reality During Pandemic," *Int. J. Inf. Syst. Technol. Akreditasi*, vol. 5, no. 158, pp. 602–611, 2022

[3] B. R. PANDAPOTAN, *Perancangan Sistem Penerjemah Sign Language to Text Berbasis Image Processing*. Bandung: Universitas Telkom, D3 Teknologi Telekomunikasi, 2022.

[4] Y. Surahman and H. Saptono, "Jurnal Informatika Terpadu EVALUASI KINERJA HDFS SEBAGAI INFRASTRUKTUR PEMBANGUNAN BIG DATA," *J. Inform. Terpadu*, vol. 4, no. 2, pp. 63–70, 2018, [Online]. Available: https://journal.nurulfikri.ac.id/index.php/JIT

[5] T. Susim and C. Darujati, "Pengolahan Citra untuk Pengenalan Wajah (Face Recognition) Menggunakan OpenCV," *J. Syntax Admiration*, vol. 2, no. 3, pp. 534–545, 2021, doi: 10.46799/jsa.v2i3.202.

[6] M. B. Subkhi and A. B. S. M. Y. A. Candra, "Klasifikasi Gambar : Membedakan Lukisan Buatan Manusia dan AI dengan CNN," *Paradig. J. Filsafat, Sains, Teknol. dan Sos. Budaya*, vol. 29, no. 4, pp. 149–155, 2023, [Online]. Available: http://www.journal.unpas.ac.id/index.php/humanitas/article/view/1919

[7] M. Muslihati, S. Sahibu, and I. Taufik, "Implementasi Algoritma Convolutional Neural Network untuk Klasifikasi Jenis Sampah Organik dan Non Organik," *MALCOLM Indones. J. Mach. Learn. Comput. Sci.*, vol. 4, no. 3, pp. 840–852, 2024, doi: 10.57152/malcolm.v4i3.1346.

[8] S. R. Suartika E. P, I Wayan, Wijaya Arya Yudhi, "Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) Pada Caltech 101," *J. Tek. ITS*, vol. 5, no. 1, p. 76, 2016, [Online]. Available: http://repository.its.ac.id/48842/

[9] F. M. Qotrunnada and P. H. Utomo, "Metode *Convolutional Neural Network* untuk Klasifikasi Wajah Bermasker," *Prisma*, vol. 5, pp. 799–807, 2022.

[10] D. W. Puteri, P. W. Buana, and I. M. Sukarsa, "Komparasi Metode *Decision Tree* dan *Deep Learning* dalam Meramalkan Jumlah Mahasiswa Drop Out Berdasarkan Nilai Akademik," *J. Internet Softw. Eng.*, vol. 1, no. 2, p. 12, 2024, doi: 10.47134/pjise.v1i2.2327.

[11] S. Lorent, "Implementasi Algoritma *Convolutional Neural Network U*ntuk Klasifikasi Ekspresi Wajah Neural Network Untuk Klasifikasi," *Skripsi , UMN*, vol. 19, no. 2, pp. 1–7, 2021, [Online]. Available: http://kc.umn.ac.id/id/eprint/17494

[12] M. M. A. Wona *et al.*, "Klasifikasi dan deteksi penyakit kulit melalui pengolahan citra dengan metode cnn," *J. Ris. dan Apl. Mhs. Inform.*, vol. 06, no. 01, pp. 43–51, 2025.

[13] M. R. Alwanda, R. P. K. Ramadhan, and D. Alamsyah, "Implementasi Metode Convolutional Neural Network Menggunakan Arsitektur LeNet-5 untuk Pengenalan Doodle," *J. Algorithm.*, vol. 1, no. 1, pp. 45–56, 2020, doi: 10.35957/algoritme.v1i1.434.

[14] K. Salma and S. Hidayat, "Deteksi Antusiasme Siswa dengan Algoritma Yolo V8 pada Proses Pembelajaran Daring," Universitas Islam Indonesia, 2024. doi: 10.35870/jimik.v5i2.716.

[15] R. D. Saputra, "Pengembangan Sistem Deteksi Objek pada Produk Retail dengan Arsitektur YOLO v4-tiny," UNIVERSITAS ISLAM INDONESIA, 2023. [Online]. Available:https://dspace.uii.ac.id/handle/123456789/45878%0Ahttps://dspace.uii.ac.id/bitstream/ handle/123456789/45878/19523235.pdf?sequence=1

[16] A. A. Fauzi, B. Harto, Mulyanto, M. Dulame, and P. Pramuditha, *Pemanfaatan Teknologi Informasi Di berbagai Sektor Pada Masa Society 5.0. PT. Sonpedia Publishing Indonesia*, no. January. 2023.

[17] H. S. Setiawan, B. Haqi, and Y. Haryanto, "Perancangan Komputerisasi Pengolahan Data Kearsipan Pada CV. JB Kreasi Mandiri," *Pros. Semin. Nas. Ris. dan Inov. Teknol.*, pp. 105–110, 2018.

[18] S. Rifky, *Artificial Intelligence : Teori dan Penerapan AI di Berbagai Bidang*, no. June. 2024.

[19] I. R. Mukhlis, *Artificial intelligence*, no. May. 2025.

[20] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, Incline Village, NV, USA, 2010, pp. 1–10. doi: 10.1109/MSST.2010.5496972.

[21] P. Abhishek, B. R. S. Kumar, and A. Govardhan, "*Customized Web User Interface for Hadoop Distributed File System*," ResearchGate, 2015. [Online]. Available: https://www.researchgate.net/publication/284898248_Customized_Web_User_Interface_for_Hadoop_Distributed_File_System