PROPOSAL PROYEK AKHIR

Sistem Deteksi Citra Aksara Jawa Berdasarkan Analisis Kemiripan Pixel
(*Pixel Difference Analysis*)

Oleh :
Silvina Hariati
225210734

PROGRAM PASCASARJANA
MAGISTER TEKNOLOGI INFORMASI
INSTITUT SAINS DAN TEKNOLOGI TERPADU
SURABAYA
2025

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Aksara Jawa (Hanacaraka) merupakan warisan budaya yang memiliki struktur visual yang unik. Dalam bidang *Computer Vision*, pengenalan karakter biasanya menggunakan metode kompleks seperti *Convolutional Neural Network* (CNN). Namun, untuk kebutuhan pembelajaran dasar dan sistem dengan komputasi ringan, diperlukan pendekatan alternatif yang lebih sederhana tanpa melibatkan arsitektur *Deep Learning*. Penelitian ini berfokus pada metode *Pixel Difference Analysis* (Analisis Perbedaan Pixel) untuk mencocokkan pola input dengan database referensi.

1.2 Rumusan Masalah

Bagaimana membangun sistem aplikasi desktop sederhana yang mampu mendeteksi tulisan tangan Aksara Jawa menggunakan metode pencocokan citra berbasis nilai error pixel terendah?

1.3 Tujuan Penelitian

1. Mengimplementasikan algoritma pengolahan citra digital dasar antara lain : Proses Grayscale (merubah RGB menjadi BW), Thresholding, Resizing.
2. Membangun antarmuka pengguna (GUI) menggunakan PyQt5 untuk memvisualisasikan proses deteksi dan tingkat kemiripan data.

BAB 2

METODOLOGI DAN PERSIAPAN

2.1 Lingkungan Pengembangan

Sistem dibangun menggunakan bahasa pemrograman Python. Sebelum memulai pengembangan, pustaka (*library*) pendukung diinstal melalui terminal dengan perintah berikut:

codeBash

```
pip install PyQt5 opencv-python numpy
```

Penjelasannya :

- **PyQt5:** Digunakan untuk membangun antarmuka grafis (GUI) yang interaktif (tombol, tabel, display gambar).
- **OpenCV (opencv-python):** Digunakan untuk manipulasi citra digital (membaca, mengubah warna, mengubah ukuran).
- **NumPy:** Digunakan untuk operasi aritmatika matriks saat menghitung selisih pixel.

2.2 Sumber Data (*Dataset*)

Penelitian ini menggunakan dua jenis sumber data:

1. **Data Training (Referensi):** Dataset digital huruf Aksara Jawa dasar yang diperoleh dari repositori publik **Kaggle**. Data ini digunakan sebagai "cetakan" atau pola baku yang bersih.
2. **Data Testing (Pengujian):** Citra tulisan tangan asli yang ditulis di atas kertas putih biasa menggunakan spidol/pulpen, kemudian difoto menggunakan kamera **Handphone**. Hal ini bertujuan untuk menguji ketahanan sistem terhadap input dunia nyata.

BAB 3

PERANCANGAN DAN IMPLEMENTASI SISTEM

3.1 Alur Proses Algoritma (*Algorithm Workflow*)

Sistem bekerja dengan prinsip membandingkan matriks gambar input melawan matriks gambar referensi. Berikut adalah tahapan prosesnya:

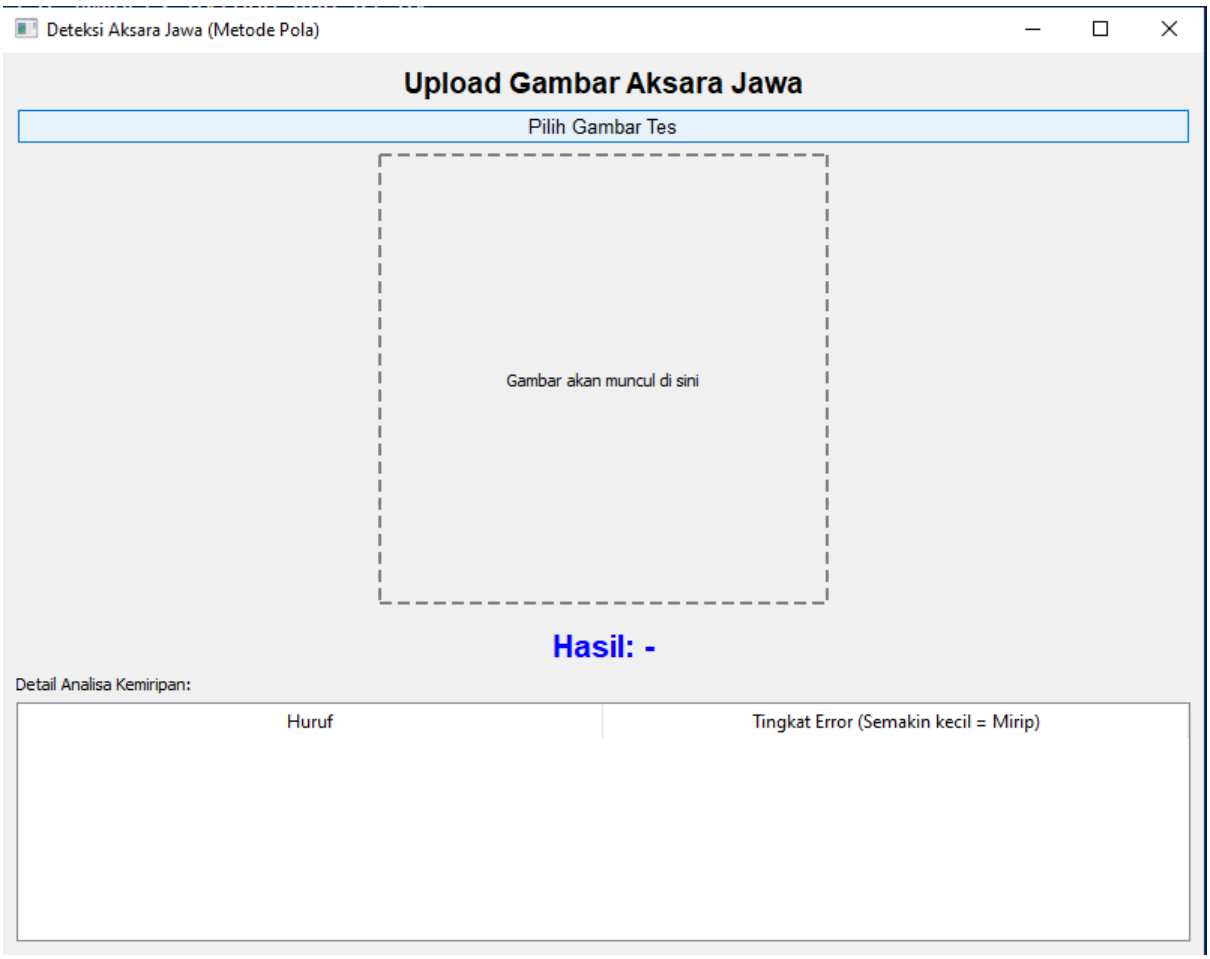
1. **Preprocessing (Pra-pemrosesan):**
   Setiap gambar (baik training maupun testing) harus melalui tahapan standarisasi agar bisa dibandingkan ("Apple to Apple").
   - **Grayscale:** Mengubah citra berwarna (RGB) menjadi citra kelabu (1 channel) untuk menghilangkan bias warna.
   - **Thresholding (Binarisasi):** Ini adalah tahap krusial untuk menemukan "data unik" atau bentuk tegas huruf. Pixel diubah menjadi hitam mutlak (0) atau putih mutlak (255).
     - *Tujuan:* Memisahkan objek tulisan (foreground) dari latar belakang kertas (background) dan menghilangkan *noise* bayangan dari foto kamera HP.
   - **Resizing:** Mengubah ukuran citra menjadi dimensi tetap (64x64 pixel) agar matriks dapat dioperasikan secara matematika.
2. **Pembentukan Model Referensi (train.py):**
   Sistem membaca folder dataset Kaggle, mengambil satu sampel terbaik untuk setiap huruf (Ha, Na, Ca, dst), memprosesnya, dan menyimpannya sebagai *dictionary* pola dalam file database_pola.pkl.
3. **Proses Matching / Prediksi (app_ui.py):**
   Saat pengguna mengunggah gambar tes:
   - Gambar tes diproses (Grayscale -> Threshold -> Resize).
   - Sistem melakukan operasi pengurangan matriks absolut (*Absolute Difference*) antara gambar tes dengan setiap gambar di database.
   - **Rumus:** Error = Rata-rata (|Pixel_Input - Pixel_Database|)

BAB 4

HASIL DAN PEMBAHASAN

4.1 Tampilan Antarmuka Sistem

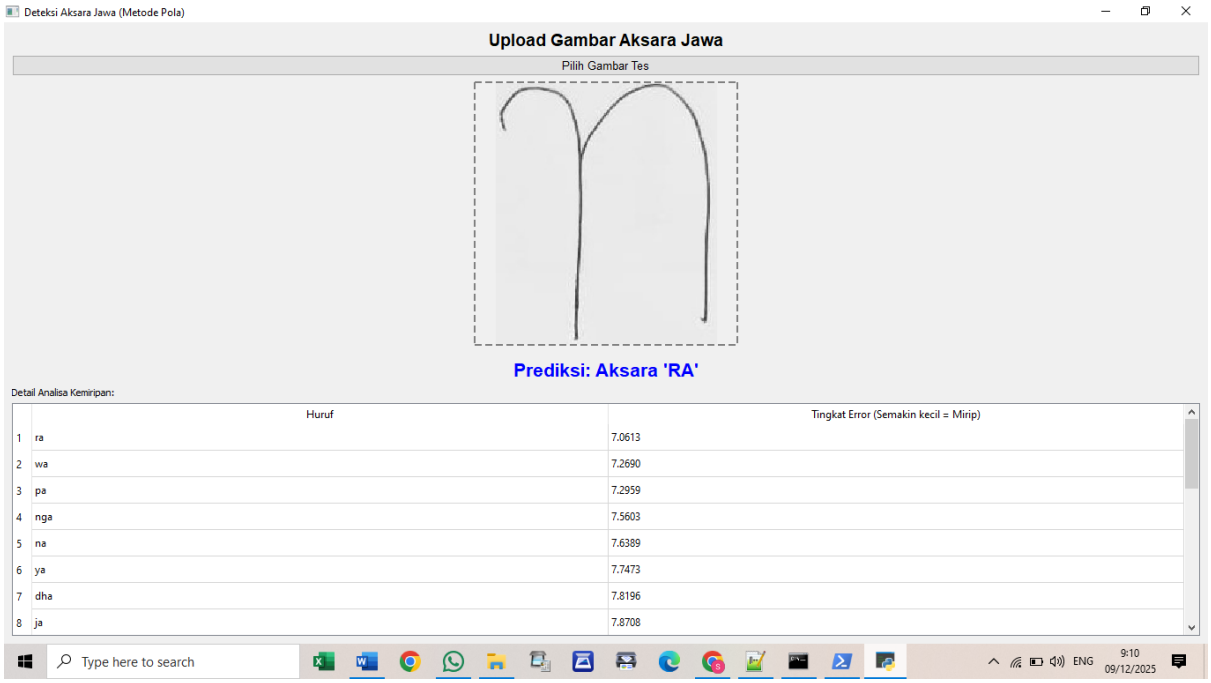
Sistem berhasil dibangun dengan antarmuka GUI berbasis PyQt5. Pengguna dapat memilih file gambar, melihat visualisasi gambar input, dan melihat hasil prediksi beserta tabel analisis data.



Gambar 4.1 Tampilan GUI Python PyQt5

4.2 Analisis Pengujian (Studi Kasus Huruf 'Ra')

Berdasarkan pengujian menggunakan foto tulisan tangan huruf "Ra" dari kamera HP, diperoleh hasil sebagai berikut (merujuk pada tangkapan layar sistem):



Gambar 4.2 Gambar Tangkapan Layar Foto data tes “Ra” (Aksara Jawa)

Penjelasan Gambar :

- **Input:** Foto tulisan tangan huruf "Ra".
- **Preprocessing:** Gambar berhasil diubah menjadi hitam-putih, sehingga bentuk lengkungan kaki huruf "Ra" terlihat jelas (fitur unik teridentifikasi).
- **Hasil Prediksi:** Sistem menampilkan **Prediksi: Aksara “RA”**.

4.3 Analisis Tabel Kemiripan (Tingkat Error)

Pada bagian bawah UI, terdapat tabel "Detail Analisa Kemiripan" yang mengurutkan huruf berdasarkan nilai error terendah (semakin kecil, semakin mirip).

Peringkat	Huruf	Tingkat Error (Skor)	Analisis
1	ra	7.0613	**Nilai Error Terkecil.** Pola input paling cocok dengan pola 'ra' di database.

2	wa	7.2690	Error sedikit lebih tinggi. Huruf 'wa' memang memiliki bentuk lengkungan yang mirip dengan 'ra'.
3	pa	7.2959	Memiliki kemiripan struktural sebagian.
...	...	...	...
8	ja	7.8708	Error semakin besar, menandakan bentuk sangat berbeda.

Pembahasan:

Meskipun input berasal dari foto kamera HP yang mungkin memiliki sedikit distorsi perspektif, proses *Thresholding* berhasil mempertahankan struktur utama huruf. Selisih nilai error antara **'ra' (7.06) dan 'wa' (7.26)** cukup **tipis**, namun algoritma tetap konsisten memilih nilai terendah sebagai pemenang.

BAB 5
KESIMPULAN

Berdasarkan implementasi sistem deteksi Aksara Jawa ini, dapat disimpulkan bahwa:

1. Metode sederhana **Analisis Kemiripan Pixel (*Pixel Difference Analysis*)** efektif digunakan untuk mendeteksi pola huruf Aksara Jawa tanpa menggunakan *Neural Network* atau AI yang rumit, asalkan citra input memiliki kontras yang cukup baik.
2. Penggunaan **PyQt5** sangat membantu dalam memvisualisasikan data numerik (tingkat error) menjadi informasi yang mudah dipahami oleh pengguna melalui tabel peringkat.
3. Proses preprocessing berupa **Grayscale** dan **Thresholding** adalah kunci utama keberhasilan sistem dalam mencocokkan data *training* (Kaggle) dengan data *testing* (Foto HP).

LAMPIRAN 1. ***SOURCE CODE : train.py***

```
import cv2
import os
import pickle
import numpy as np

# KONFIGURASI
DATA_SOURCE = 'dataset_aksara'
IMG_SIZE = (64, 64) # Ukuran baku

database_pola = {}

print("--- Mulai Membuat Database Pola ---")

if not os.path.exists(DATA_SOURCE):
    print(f"Error: Folder '{DATA_SOURCE}' tidak ditemukan!")
    exit()

list_huruf = sorted(os.listdir(DATA_SOURCE))

for huruf in list_huruf:
    folder_path = os.path.join(DATA_SOURCE, huruf)

    if os.path.isdir(folder_path):
        files = os.listdir(folder_path)
        if len(files) > 0:
            # Ambil 1 gambar pertama sebagai referensi pola utama
            path_gambar = os.path.join(folder_path, files[0])

            # PROSES GAMBAR (Preprocessing)
            img = cv2.imread(path_gambar)
            if img is not None:
                # 1. Grayscale
                img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                # 2. Threshold (Hitam Putih Tegas)
                _, img_thresh = cv2.threshold(img_gray, 127, 255,
cv2.THRESH_BINARY_INV)
                # 3. Resize
```

```
                img_final = cv2.resize(img_thresh, IMG_SIZE)

                # Simpan ke memori
                database_pola[huruf] = img_final
                print(f"✅ Pola tersimpan: {huruf}")

# Simpan ke file
with open('database_pola.pkl', 'wb') as f:
    pickle.dump(database_pola, f)

print("--- Selesai! File 'database_pola.pkl' siap digunakan. ---")
```

LAMPIRAN *2. SOURCE CODE : app_ui.py*

```
import sys
import cv2
import pickle
import numpy as np
import os
from PyQt5.QtWidgets import (QApplication, QMainWindow, QLabel,
QPushButton,
                             QFileDialog, QVBoxLayout, QWidget,
QTableWidget,
                             QTableWidgetItem, QHeaderView, QMessageBox)
from PyQt5.QtGui import QPixmap, QImage, QFont
from PyQt5.QtCore import Qt

class AksaraApp(QMainWindow):
    def __init__(self):
        super().__init__()

        # Konfigurasi Awal
        self.IMG_SIZE = (64, 64)
        self.database_pola = {}
        self.load_database()

        self.initUI()

    def load_database(self):
        # Cek apakah file database ada
        if os.path.exists('database_pola.pkl'):
            with open('database_pola.pkl', 'rb') as f:
                self.database_pola = pickle.load(f)
        else:
            QMessageBox.critical(self, "Error", "File 'database_pola.pkl'
belum ada!₩nSilakan jalankan train.py dulu.")

    def initUI(self):
        self.setWindowTitle("Deteksi Aksara Jawa (Metode Pola)")
        self.setGeometry(100, 100, 800, 600) # Ukuran jendela x, y, width,
height

        # --- Widget Utama ---

        # 1. Label Judul
        self.label_judul = QLabel("Upload Gambar Aksara Jawa", self)
        self.label_judul.setAlignment(Qt.AlignCenter)
```

```
        self.label_judul.setFont(QFont('Arial', 14, QFont.Bold))

        # 2. Area Tampil Gambar
        self.image_label = QLabel(self)
        self.image_label.setAlignment(Qt.AlignCenter)
        self.image_label.setText("Gambar akan muncul di sini")
        self.image_label.setStyleSheet("border: 2px dashed gray;
background-color: #f0f0f0;")
        self.image_label.setFixedSize(300, 300)

        # 3. Tombol Pilih Gambar
        self.btn_load = QPushButton("Pilih Gambar Tes", self)
        self.btn_load.setFont(QFont('Arial', 10))
        self.btn_load.clicked.connect(self.browse_image)

        # 4. Label Hasil Prediksi Utama
        self.label_hasil = QLabel("Hasil: -", self)
        self.label_hasil.setFont(QFont('Arial', 16, QFont.Bold))
        self.label_hasil.setAlignment(Qt.AlignCenter)
        self.label_hasil.setStyleSheet("color: blue; margin-top: 10px;")

        # 5. Tabel Detail Kemiripan
        self.table_result = QTableWidget()
        self.table_result.setColumnCount(2)
        self.table_result.setHorizontalHeaderLabels(["Huruf", "Tingkat
Error (Semakin kecil = Mirip)"])

self.table_result.horizontalHeader().setSectionResizeMode(QHeaderView.Stret
ch)

        # --- Layout (Tata Letak) ---
        layout = QVBoxLayout()
        layout.addWidget(self.label_judul)
        layout.addWidget(self.btn_load)

        # Layout tengah (Gambar)
        layout.addWidget(self.image_label, alignment=Qt.AlignCenter)
        layout.addWidget(self.label_hasil)

        # Label kecil untuk tabel
        lbl_detail = QLabel("Detail Analisa Kemiripan:")
        layout.addWidget(lbl_detail)
        layout.addWidget(self.table_result)
```

```
        # Set Layout ke Window
        container = QWidget()
        container.setLayout(layout)
        self.setCentralWidget(container)

    def browse_image(self):
        # Buka dialog pilih file
        fname, _ = QFileDialog.getOpenFileName(self, 'Pilih Gambar', '.',
'Image files (*.jpg *.png *.jpeg)')

        if fname:
            self.process_and_predict(fname)

    def process_and_predict(self, file_path):
        # 1. Tampilkan Gambar di UI
        pixmap = QPixmap(file_path)
        self.image_label.setPixmap(pixmap.scaled(300, 300,
Qt.KeepAspectRatio))

        # 2. Baca & Proses Gambar dengan OpenCV
        img = cv2.imread(file_path)
        if img is None:
            return

        # --- PREPROCESSING (Wajib sama dengan train.py) ---
        img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        _, img_thresh = cv2.threshold(img_gray, 127, 255,
cv2.THRESH_BINARY_INV)
        img_input = cv2.resize(img_thresh, self.IMG_SIZE)
        # ------------------------------------------------

        # 3. Hitung Kemiripan dengan Database
        hasil_analisa = [] # List untuk menyimpan (nama_huruf, nilai_error)

        skor_terbaik = float('inf')
        huruf_terdeteksi = "Tidak Diketahui"

        for huruf, pola_ref in self.database_pola.items():
            # Hitung selisih pixel (Mean Squared Error sederhana)
            # absdiff menghitung beda warna pixel. Jika sama persis
hasilnya 0.
            selisih = cv2.absdiff(img_input, pola_ref)
            skor_error = np.mean(selisih)
```

```
            hasil_analisa.append((huruf, skor_error))

            # Update jika menemukan error yang lebih kecil
            if skor_error < skor_terbaik:
                skor_terbaik = skor_error
                huruf_terdeteksi = huruf

        # 4. Update UI Hasil
        self.label_hasil.setText(f"Prediksi: Aksara
'{huruf_terdeteksi.upper()}'")

        # 5. Isi Tabel Detail (Diurutkan dari error terkecil)
        hasil_analisa.sort(key=lambda x: x[1]) # Sort by skor error

        self.table_result.setRowCount(len(hasil_analisa))
        for row, (huruf, skor) in enumerate(hasil_analisa):
            self.table_result.setItem(row, 0, QTableWidgetItem(huruf))
            self.table_result.setItem(row, 1,
QTableWidgetItem(f"{skor:.4f}"))

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = AksaraApp()
    ex.show()
    sys.exit(app.exec_())
```

Lampiran 3 . Tampilan Program