Pengenalan Tulisan pada Dokumen Terdegradasi menggunakan Pengambangan Citra dan CNN

Ariel Herfrison 13522002[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522002@std.stei.itb.ac.id, [2]arielherfrison@gmail.com

Abstrak*—Dokumen bersejarah rentan terhadap kerusakan lingkungan dan keausan fisik yang membuatnya tidak dapat dibaca. Ketika dokumen tidak dapat dibaca, pengetahuan, teknologi, dan budaya yang terkandung di dalamnya tidak lagi dapat dipelajari. Makalah ini merumuskan sistem untuk memperbaiki dan mentranskripsi dokumen terdegradasi secara otomatis. Perbaikan dokumen dilakukan menggunakan metode pengambangan Otsu, Sauvola, dan Su. Karakter dalam dokumen yang telah diperbaiki kemudian dideteksi menggunakan *CNN* yang telah dilatih dari data set *EMNIST*. Hasil uji coba menunjukkan bahwa kombinasi pengambangan citra dan *CNN* memiliki potensi dalam memperbaiki dan mengenal tulisan dalam dokumen terdegradasi secara otomatis. Namun, masih terdapat banyak kekurangan, antara lain dalam segmentasi karakter dan prediksi karakter. Metode pengambangan Otsu, Sauvola, dan Su dapat memisahkan tulisan dengan latar belakang dengan cukup baik. Namun, segmentasi karakter cenderung tidak berurut dan tidak dapat memisahkan dua karakter berbeda yang tersambung. Selain itu, *CNN* masih belum mampu memprediksi karakter yang telah disegmentasi dengan benar. Oleh karena itu, perlu dilakukan eksperimen lebih lanjut untuk menemukan metode segmentasi citra dan arsitektur *CNN* yang efektif.

***Kata Kunci*— Citra, *CNN*, Degradasi, Dokumen, Pengambangan**

I. PENDAHULUAN

Tidak jarang bahwa sejarah bermain peran penting dalam perkembangan peradaban. Sejarah mengandung ilmu pengetahuan, teknologi, budaya, maupun filosofi yang dapat kita pelajari di zaman sekarang. Sejarah dapat terkandung dalam buku atau dokumen fisik yang berumur ratusan atau bahkan ribuan tahun. Namun, dokumen tersebut seringkali rentan terhadap kerusakan lingkungan dan keausan fisik, yang menyebabkan degradasi signifikan seiring waktu. Ketika dokumen tersebut tidak dapat dibaca, pengetahuan, teknologi, dan budaya yang terkandung di dalamnya menjadi hilang. Oleh karena itu, dibutuhkan teknik restorasi dokumen yang tidak hanya dapat melestarikan dokumen pada zaman ini, tetapi juga untuk mengembalikan pengetahuan yang terdapat dalam dokumen bersejarah yang rusak.[3]

Terdapat beragam macam tipe degradasi yang dapat ditemukan pada dokumen-dokumen fisik. Degradasi tersebut antara lain cahaya yang tidak merata, tulisan yang pudar, tulisan yang menembus kertas, kotoran atau noda, dan *blur*.[10] Degradrasi-degradasi tersebut dapat mempersulit proses restorasi citra dan membutuhkan teknik yang berbeda sesuai dengan degradasi yang ada. Sebagai contoh, teknik *thresholding* dapat memperbaiki tulisan yang pudar, tetapi tidak dapat menghilangkan *blur*.

Salah satu metode restorasi dokumen adalah Segmentasi Citra. Segmentasi citra berfungsi untuk memisahkan latar belakang dengan objek pada latar depan, salah satunya adalah tulisan.[7] Salah satu metode segmentasi citra adalah pengambangan. Pengambangan dapat memisahkan tulisan dengan latar belakang berdasarkan perbedaan intensitas pixel antara keduanya. Namun, pengambangan buruk dalam memisahkan objek dengan intensitas yang hampir sama seperti noda dengan tulisan.[10]

Setelah dokumen diperbaiki, tulisan pada dokumen dapat ditranskripsi menjadi tulisan digital melalui *optical character recognition* atau *OCR*. OCR dapat mengenali tulisan antara lain dengan membandingkannya dengan suatu citra referensi atau melalui fitur tulisan seperti garis atau lingkaran.[5] Metode OCR akan dibahas lebih lanjut dalam subbab II.C. Dengan menggabungkan *image segmentation* dan OCR, tulisan dalam dokumen terdegradasi dapat dengan cepat dan otomatis diperbaiki dan ditranskripsi ke dalam bentuk digital.

II. LANDASAN TEORI

A. Segmentasi Citra

Segmentasi Citra adalah proses identifikasi *region* dari citra, dimana pixel-pixel pada *region* yang sama terhubung dengan satu sama lain.[7] Segmentasi citra berfungsi untuk memisahkan objek, misalnya tulisan, dengan latar belakang sehingga objek tersebut dapat diproses secara tersendiri tanpa memroses seluruh citra. Segmentasi dilakukan dengan menemukan bagian citra yang seragam berdasarkan properti citra, seperti kecerahan, warna, atau tekstur. Salah satu metode segmentasi citra adalah pengambangan atau *thresholding*.

Pengambangan mempartisi citra berdasarkan nilai intensitas masing-masing pixel relatif terhadap suatu nilai ambang T. Misalkan citra lama, A, dengan pixel-pixel pada posisi (i,j) dengan intensitas $f(i,j)$, pengambangan menghasilkan citra biner, B, berdasarkan fungsi berikut:

$$f_B(i,j) = \begin{cases} 1, & f_g(i,j) \le T \\ 0, & lainnya \end{cases}$$

Pemilihan nilai *T* yang tepat adalah masalah utama dari pengambangan. Nilai *T* dapat ditentukan berdasarkan intensitas pixel seluruh citra (pengambangan global) atau berdasarkan intensitas pixel-pixel tetangga (pengambangan lokal). Pengambangan global dan lokal umumnya perlu dikombinasikan untuk menghasilkan pengambangan yang lebih sempurna.[9] Pada makalah ini, akan dikaji metode kombinasi pengambangan oleh [Su et al, tahun 2011]. Pengambangan akan memanfaatkan pengambangan global dengan metode Otsu[7] dan pengambangan lokal dengan metode Sauvola.[8]

A.1. Pengambangan Otsu

Pengambangan Otsu menentukan nilai ambang dengan memaksimalkan variansi intensitas pixel antara kedua *region* hasil pengambangan (σ_B).[7] Variansi antar-*region*, σ_B, dihitung berdasarkan rumus berikut:

$$\sigma_B(k) = P_1 P_2 (m_1 - m_2)^2$$

dengan *k* adalah nilai ambang, *(1)* adalah himpunan pixel dengan intensitas *[0, k]*, *(2)* adalah himpunan pixel dengan intensitas *[k+1, 255]*, *Px* adalah jumlah pixel dalam himpunan *x*, dan m_i adalah rata-rata intensitas pixel dalam himpunan *x*. Konstanta-konstanta tersebut dihitung berdasarkan rumus berikut:

$$m_1 = \frac{1}{P_1} \sum_{i=0}^{k} i.p_i$$

$$m_2 = \frac{1}{P_2} \sum_{i=k+1}^{255} i.p_i$$

Dengan mencoba semua kemungkinan nilai ambang *k*, dari 1 sampai 254, metode Otsu dapat menemukan intensitas terbaik untuk memisahkan objek latar depan dengan latar belakang. Namun, metode Otsu cenderung memiliki keterbatasan dalam mengambang citra dengan derau tidak merata.[10]

A.2. Pengambangan Sauvola

Pengambangan Sauvola adalah metode pengambangan lokal yang menentukan nilai ambang berdasarkan rata-rata dan standar deviasi pixel lokal.[8] Nilai ambang *T* ditentukan menggunakan rumus berikut:

$$T(x,y) = m(x,y) \cdot \left[1 + k \cdot \left(\frac{s(x,y)}{R} - 1\right)\right],$$

Dengan *(x,y)* adalah koordinat pixel lokal, *m* adalah rata-rata intensitas pixel, *s* adalah standar deviasi intensitas pixel, *k* adalah parameter yang ditentukan pengguna, dan *R* adalah jangkauan dinamis dari standar deviasi. Pada umumnya, *R* bernilai 128 untuk citra 8-bit.

A.3. Metode Su

[Su et al, tahun 2011] merumuskan metode kombinasi algoritma pengambangan untuk mengurangi keterbatasan masing-masing algoritma. Metode Su membagi pixel menjadi tiga kategori: *foreground* jika pixel berwarna putih di semua metode pengambangan, *background* jika pixel berwarna hitam di semua metode pengambangan, dan *uncertain* jika pixel berbeda warna di antara metode pengambangan. Metode Su mengubah warna pixel *uncertain* menjadi putih (*foreground*) atau hitam (*background*) berdasarkan selisih/jarak intensitas pixel tersebut dengan rata-rata intensitas pixel di sekitarnya. Metode Su dapat dirumuskan sebagai berikut:

$$P(x) = \begin{cases} foreground, & |I_F - I(x)| > |I(x) - I_B| \\ background, & lainnya \end{cases}$$

dengan *P(x)* adalah pixel *uncertain*, I_F adalah rata-rata intensitas pixel *foreground*, I_B adalah rata-rata intensitas pixel *background*, dan *I(x)* adalah intensitas pixel *uncertain*.

B. Optical Character Recognition

Optical Character Recognition adalah proses perubahan karakter atau tulisan dari bentuk citra menjadi teks digital yang dapat dibaca mesin.[5] Salah satu metode pengenalan tulisan adalah menggunakan *pattern matching*. *Pattern Matching* adalah metode perbandingan citra karakter yang telah terisolasi terhadap citra karakter templat. Perbandingan kedua citra dilakukan secara pixel per pixel sehingga identifikasi karakter dilakukan berdasarkan jumlah pixel yang sama.

Metode lain adalah dengan memanfaatkan pembelajaran mendalam *(deep learning)* seperti *CNN* yang telah dilatih untuk mengidentifikasi karakter. CNN dapat mempermudah proses *pattern matching* karena menghilangkan kebutuhan untuk melakukan ekstraksi fitur secara manual.[5] *CNN* juga bersifat adaptif karena dapat mempelajari pola-pola derau tanpa diperlukan perbaikan citra selama data latih cukup besar.

III. IMPLEMENTASI

Makalah ini merumuskan sistem untuk melakukan pengenalan tulisan pada dokumen terdegradasi dengan memanfaatkan pengambangan citra dan *CNN*. Program diimplementasi dalam bahasa *Python* dengan *library* antara lain *opencv, tensorflow, keras,* dan *numpy*. Program antara lain terdiri atas tiga modul, yaitu:

1. *Thresholding*
2. *Segmentation*
3. *CNN*

Berikut adalah kode sumber implementasi sistem pengenalan tulisan pada dokumen terdegradasi:

1. *Thresholding*

```
import cv2
import numpy as np
from scipy.signal import convolve2d
import matplotlib.pyplot as plt

def manual_otsu(image):
    if len(image.shape) == 3:
        image = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)
```

```
    # Calculate Histogram
    hist = cv2.calcHist([image], [0], None, [256], [0, 256]).ravel()

    total = image.shape[0] * image.shape[1]
    max_variance = 0
    best_threshold = 0

    # Iterate through chosen thresholds: iterate all 0-(L-1)
    for k in range(256):
        # Probability) <= k
        w0 = np.sum(hist[:k]) / total
        # Probability >= k
        w1 = np.sum(hist[k:]) / total

        if w0 == 0 or w1 == 0:
            continue

        # Mean intensity <= k
        mu0 = np.sum(np.arange(k) * hist[:k]) / np.sum(hist[:k])

        # Mean intensity >= k
        mu1 = np.sum(np.arange(k, 256) * hist[k:]) / np.sum(hist[k:])

        variance_between = w0 * w1 * ((mu0 - mu1) ** 2)

        if variance_between > max_variance:
            max_variance = variance_between
            best_threshold = k

    # Apply threshold
    binary_img = (image > best_threshold).astype(np.uint8) * 255
    return binary_img, best_threshold # also return threshold for debug

def manual_sauvola(image, window_size=15, k=0.5, R=128):
    if len(image.shape) == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    img_float = image.astype(np.float32)

    # Create Kernel
    kernel = np.ones((window_size, window_size), dtype=np.float32) / (window_size**2)

    # Calculate local mean (mean)
    mean = convolve2d(img_float, kernel, mode='same', boundary='symm')

    # Calculate local standard deviation (std)
    img_sq = img_float**2
    mean_sq = convolve2d(img_sq, kernel, mode='same', boundary='symm')

    variance = mean_sq - mean**2
    std = np.sqrt(np.maximum(variance, 0)) # check edge case

    # Apply Sauvola Formula
    threshold = mean * (1 + k * (std / R - 1))

    # Apply threshold
    binary = (img_float > threshold).astype(np.uint8) * 255
    return binary

def combine_su(gray_img, image1, image2, window_size = 5):
    foreground = image1 & image2
    background = ~(image1 | image2)
    uncertain = image1 ^ image2

    output = foreground.copy()

    pad = window_size // 2
    padded_img = np.pad(gray_img, pad, mode='constant')
    padded_fg = np.pad(foreground, pad, mode='constant')
    padded_bg = np.pad(background, pad, mode='constant')

    rows, cols = gray_img.shape
    for i in range(rows):
        for j in range(cols):
            if uncertain[i, j] == 1:
                # Extract local windows
                win_fg = padded_fg[i:i+window_size, j:j+window_size]
                win_bg = padded_bg[i:i+window_size, j:j+window_size]
                win_gray = padded_img[i:i+window_size, j:j+window_size]

                # Get mean of foreground & background
                fg_pixels = win_gray[win_fg == 1]
                bg_pixels = win_gray[win_bg == 1]

                mu_fg = np.mean(fg_pixels) if fg_pixels.size > 0 else 0
                mu_bg = np.mean(bg_pixels) if bg_pixels.size > 0 else 255

                # Distance-based classification
                pixel_val = gray_img[i, j]
                if abs(pixel_val - mu_fg) < abs(pixel_val - mu_bg):
                    output[i, j] = 1
                else:
                    output[i, j] = 0

    return output

def binarize_image(image):
    gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
    # Process
    otsu_img, _ = manual_otsu(gray_img)
    sauvola_img = manual_sauvola(gray_img)
    final_image = combine_su(gray_img, otsu_img, sauvola_img)

    return final_image
```

Modul *Thresholding* terdiri atas fungsi-fungsi yang mengimplementasikan metode-metode pengambangan yang telah dijabarkan pada subbab II.A. Fungsi-fungsi pada modul *Thresholding,* antara lain:

a) manual_otsu
Fungsi ini merupakan implementasi dari metode Otsu. Fungsi ini menerima masukan berupa citra dan mengeluarkan citra biner yang telah diambangkan. Nilai ambang *k* yang mungkin diambil adalah 1-254.

b) manual_sauvola
Fungsi ini merupakan implementasi dari metode Sauvola. Fungsi ini menerima masukan berupa citra awal, ukuran kernel, serta konstanta *k* dan *R*. Nilai ukuran kernel yang digunakan adalah 15x15, nilai *k* adalah 0.5, dan nilai *R* adalah 128. Keluaran dari fungsi ini adalah citra biner yang telah diambangkan.

c) combine_su
Fungsi ini merupakan implementasi dari metode Su. Fungsi ini menerima masukan berupa citra *grayscale*, citra biner pertama, citra biner kedua, dan ukuran kernel. Ukuran kernel yang digunakan adalah 5x5. Keluaran dari fungsi ini adalah citra biner yang telah diambangkan.

d) binarize_image
Fungsi ini berfungsi menggabungkan semua metode pengambangan. Fungsi ini menerima masukan berupa citra awal. Pertama-tama, fungsi memanggil pengambangan otsu dan sauvola. Kemudian, hasil pengambangan otsu dan sauvola tersebut dimasukkan ke dalam pengambangan Su. Keluaran fungsi adalah citra biner hasil pengambangan Su tersebut.

2. *Segmentation*

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def segment_characters(binary_img):
    # Invert so text is white and background is black (0)
    if np.mean(binary_img) > 127:
        binary_img = cv2.bitwise_not(binary_img)

    contours, _ = cv2.findContours(binary_img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    char_images = []
    # Sort from left to right, top to bottom
    contours = sorted(contours, key=lambda ctr: cv2.boundingRect(ctr)[1])

    for ctr in contours:
        x, y, w, h = cv2.boundingRect(ctr)

        # Filter noise by size
        if w > 5 and h > 10:
            roi = binary_img[y:y+h, x:x+w]

            # Resize to match CNN input
            roi_resized = cv2.resize(roi, (28, 28), interpolation=cv2.INTER_AREA)
            char_images.append(roi_resized)

    return char_images
```

Modul *Segmentation* terdiri atas fungsi untuk mensegmentasi karakter dalam citra menjadi citra-citra kecil yang terisolir. Fungsi ini menerima masukan berupa citra biner yang telah diambangkan. Kemudian, pixel-pixel karakter diidentifikasi dan dipisahkan menggunakan fungsi *findContours* dari *opencv*. *findContours* mensegmentasi citra berdasarkan area pixel-pixel yang memiliki intensitas yang sama. Karakter-karakter yang telah disegmentasi/diisolir kemudian diubah menjadi kotak berukuran 28x28 supaya sesuai dengan arsitektur *CNN*. Keluaran fungsi adalah kumpulan citra karakter yang telah terisolir tersebut.

3. *CNN*

```
import numpy as np
import tensorflow as tf
from keras import layers, models
from emnist import extract_training_samples, extract_test_samples
from .segmentation import segment_characters, display_segments
import os
import gzip
import idx2numpy

def train_emnist_model():
    # balanced set -> 47 classes: 0-9, A-Z, some lowercase
    x_train, y_train = get_emnist_dataset()

    # Normalize and Reshape
    x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0

    num_classes = 47

    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
```

```
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

    print("Training model...")
    model.fit(x_train, y_train, epochs=10,
batch_size=128, validation_split=0.1)

    model.save('emnist_model.keras')
    return model

def predict_text(final_image, model):
    print("Predicting text...")
    EMNIST_MAP =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXY
Zabdefghnqrt"

    def get_label(index):
        return EMNIST_MAP[index]

    chars = segment_characters(final_image)
    # display_segments(chars)

    detected_text = []

    for c in chars:
        # Prepare for CNN (1, 28, 28, 1)
        img_input = c.reshape(1, 28, 28, 1) / 255.0

        prediction = model.predict(img_input, verbose=0)
        label_idx = np.argmax(prediction)
        detected_text.append(get_label(label_idx))

    return "".join(detected_text)
```

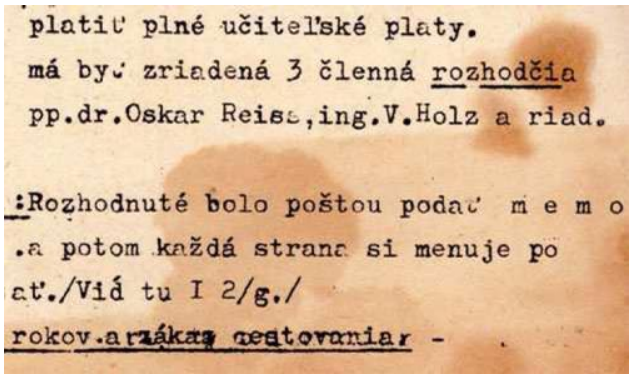
Modul *CNN* terdiri atas fungsi *train_emnist_model* untuk melatih model *CNN* dan fungsi *predict_text* melakukan prediksi menggunakan model yang telah dilatih. Model dilatih dari data set *EMNIST* yang berisi citra-citra karakter tulisan tangan. Arsitektur *CNN* terdiri dari *convolution layer* berisi 32 filter, *convolution layer* berisi 64 filter, *fully connected layer* berisi 256 filter, diakhiri dengan *output layer* yang menggunakan fungsi *softmax*.

4. Fungsi Utama

```
    def main_func():
        isLoad = input("Load model?[Y/N] ")

        if (isLoad.upper() == "Y"):
            model_path = select_model()
            if model_path:
                model = cnn.load_cnn_model(model_path)

            if not model_path or not model:
                model = cnn.train_emnist_model()
        else:
            model = cnn.train_emnist_model()

        path = select_image()

        if not path:
            print("No file selected...")
            return

        # Load as grayscale directly
        original_img = cv2.imread(path)
        if original_img is None:
            print("Error: Could not decode image.")
            return

        # Process
        final_image =
thresholding.binarize_image(original_img)

        chars =
segmentation.segment_characters(final_image)

        text = cnn.predict_text(final_image, model)

        app = OutputGUI(original_img, final_image,
chars, text)
        app.run()

    if __name__ == "__main__":
    main_func()
```

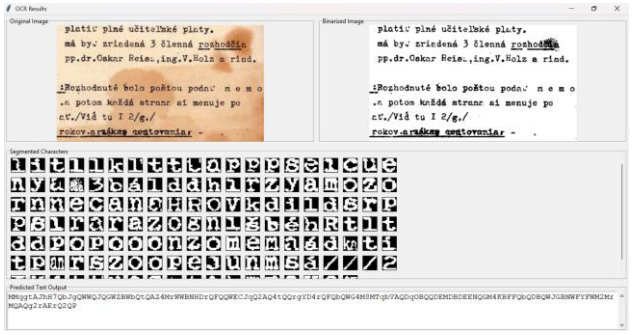
Fungsi Utama menyatukan fungsi pengambangan, segmentasi, dan pengenalan teks menggunakan *CNN*. Pertama-tama, model *CNN* dilatih menggunakan fungsi *train_emnist_model* atau dimuat menggunakan fungsi *load_cnn_model*. Kemudian, citra dimuat dan diambangkan menggunakan fungsi *binarize_image* yang menerapkan metode Su berbasis metode Otsu dan Sauvola. Hasil pengambangan lalu disegmentasi menggunakan fungsi *segment_characters* untuk memperoleh citra-citra karakter yang terisolasi. Setiap karakter kemudian diidentifikasi dan digabungkan menjadi satu rangkaian karakter menggunakan fungsi *predict_text* dari *CNN*. Terakhir, hasil pengenalan tulisan ditampilkan dalam *GUI* melalui fungsi *OutputGUI*.

IV. Uji Coba dan Analisis



Gambar IV.1. Citra Dokumen Terdegradasi
Sumber: https://spie.org/news/0681-degraded-document-image-enhancement

Program yang telah diimplementasi kemudian diuji menggunakan citra dokumen terdegradasi pada Gambar IV.1. Berikut adalah contoh tampilan hasil pengenalan tulisan.



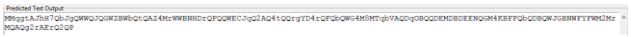
Gambar IV.2. Hasil Pengenalan Tulisan
Sumber: Dokumentasi pribadi

Gambar IV.3. Hasil Pengambangan
Sumber: Dokumentasi pribadi

Gambar IV.3 menunjukkan bahwa kombinasi metode pengambangan Otsu, Sauvola, dan Su cukup baik dalam memisahkan tulisan dengan latar belakang. Namun, masih terdapat derau noda yang tidak dapat dihilangkan melalui pengambangan.

Gambar IV.4. Hasil Segmentasi Karakter
Sumber: Dokumentasi pribadi

Gambar IV.4 menunjukkan bahwa fungsi segmentasi secara mayoritas sudah dapat mengisolir citra-citra karakter. Namun, masih terdapat karakter tersambung yang tidak dapat dipisahkan. Selain itu, fungsi segmentasi tidak dapat mengidentifikasi tata letak tulisan. Karakter cenderung belum terurut dari atas ke bawah dan kiri ke kanan. Dengan demikian, dibutuhkan tahap *preprocessing* maupun *post-processing* untuk mengidentifikasi tata letak tulisan.



Gambar IV.5. Hasil Prediksi *CNN*
Sumber: Dokumentasi pribadi

Gambar IV.5 menunjukkan bahwa *CNN* belum cukup baik dalam mengenali karakter. Terlihat bahwa masih terdapat sangat banyak kesalahan prediksi karakter. Hal ini dapat disebabkan perbedaan bentuk karakter pada citra uji dengan data set latihan. Kesalahan prediksi juga dapat disebabkan timbulnya *overfitting* saat pelatihan atau arsitektur *CNN* yang tidak sesuai.

V. Kesimpulan

Berdasarkan hasil eksperimen, ditemukan bahwa sistem pengenalan tulisan yang telah dirancang belum sempurna. Meskipun program sudah dapat memisahkan antara tulisan dan latar belakang dengan baik, program gagal dalam mendeteksi tata letak tulisan dan memprediksi karakter. Masih terdapat banyak perbaikan yang dapat dilakukan, terutama pada metode segmentasi citra dan pelatihan model *CNN*. Program dapat menambahkan tahap *preprocessing* seperti *layout analysis* untuk mendeteksi tata letak tulisan dan melakukan prediksi berdasarkan *region* tulisan.[4] Selain itu, model *CNN* dapat dikembangkan dengan mengubah arsitektur atau menambah data pelatihan

VI. Lampiran

Pranala Github kode sumber: github.com/Ariel-HS/DegradedDocumentOCR

Ucapan Terima Kasih

Penulis mengucapkan rasa syukur kepada Tuhan Yang Maha Esa oleh karena hikmat dan anugerahnya penulis dapat menyelesaikan makalah ini. Penulis juga berterima kasih kepada orang tua yang menyemangati dan mendukung penulis selama penulisan makalah ini. Penulis juga menyampaikan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen mata kuliah IF4073 Pemrosesan Citra Digital yang telah memberi ilmu yang menjadi landasan penulisan makalah ini. Terakhir, penulis berterimakasih kepada teman-teman penulis yang menjadi penyemangat sekaligus rekan diskusi selama penyelesaian makalah ini.

Referensi

[1] Agam, G., Frieder, O., & Frieder, G. (2007, Mei 16). Degraded document image enhancement. SPIE, the international society for

optics and photonics. https://spie.org/news/0681-degraded-document-image-enhancement, diakses pada 24 Desember 2025
[2] Contours : Getting Started. OpenCV. (n.d.). https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html, diakses pada 24 Desember 2025
[3] Conway, P., & Conway, M. O. H. (2018). Flood in Florence, 1966: A Fifty-Year Retrospective. *Maize Books*.
[4] Ha, J., Haralick, R. M., & Phillips, I. T. (1995, Agustus). Recursive XY cut using bounding boxes of connected components. In *Proceedings of 3rd International Conference on Document Analysis and Recognition* (Vol. 2, pp. 952-955). IEEE.
[5] Konovalchuk, N. (n.d.). OCR Algorithms: Types, Use Cases and Best Solutions. Itransition. https://www.itransition.com/computer-vision/ocr-algorithm, diakses pada 24 Desember 2025
[6] Munir, R. (2025). Convolutional Neural Network. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2025-2026/22-Segmentasi-Citra-Bagian1-2025.pdf, diakses pada 24 Desember 2025
[7] Munir, R. (2025). Segmentasi Citra (Bagian 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2025-2026/21-CNN-2025.pdf, diakses pada 24 Desember 2025
[8] Sauvola, J., & Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern recognition*, *33*(2), 225-236.
[9] Su, B., Lu, S., & Tan, C. L. (2011, September). Combination of document image binarization techniques. In *2011 International Conference on Document Analysis and Recognition* (pp. 22-26). IEEE.
[10] Sulaiman, A., Omar, K., & Nasrudin, M. F. (2019). Degraded historical document binarization: A review on issues, challenges, techniques, and future directions. *Journal of imaging*, *5*(4), 48.
[11] The EMNIST dataset. NIST. (2024, December 2). https://www.nist.gov/itl/products-and-services/emnist-dataset, diakses pada 24 Desember 2025

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

Ariel Herfrison 13522002