Jurnal Infortech
Volume 7 No. 2 Desember 2025
E-ISSN: 2715-8160

Pengembangan Aplikasi Check Font Dengan Algoritma Distribusi Font Pada Dokumen PDF Menggunakan Python

Mugi Raharjo[1], Firmansyah[2], Rian Septian Anwar[3], Tommi Alfian Armawan Sandi[4]

[1]Informatika, Universitas Nusa Mandiri
e-mail: [1]mugi.mou@nusamandiri.ac.id

[2,3]Informatika, Universitas Bina Sarana Informatika
e-mail: [2]firmansyah.fmy@bsi.ac.id,[3]rian.ptn@bsi.ac.id

[4]Informatika, Universitas Bina Sarana Informatika PSDKU Kabupaten Karawang
e-mail: [4]tommi.taf@bsi.ac.id

Diterima	Direvisi	Disetujui
24-09-2025	11-10-2025	22-12-2025

Abstrak - Penggunaan font yang konsisten pada dokumen akademik, seperti skripsi, tesis, dan laporan penelitian, merupakan salah satu persyaratan penting dalam standar penulisan ilmiah. Namun, proses pengecekan font secara manual masih memerlukan waktu yang lama dan berpotensi menimbulkan kesalahan. Penelitian ini bertujuan untuk mengimplementasikan algoritma analisis distribusi *font* pada pembuatan aplikasi deteksi font otomatis berbasis *Python* dan *Streamlit*. Metode yang digunakan mencakup ekstraksi teks dan informasi font dari dokumen PDF, normalisasi nama font untuk menyatukan variasi tipografi tertentu seperti *Times New Roman* regular, bold, italic), perhitungan distribusi font menggunakan analisis frekuensi, serta visualisasi hasil dalam bentuk grafik batang. Aplikasi ini juga dilengkapi dengan fitur anotasi otomatis berupa highlight pada teks yang tidak sesuai dengan font standar yang ditentukan. Implementasi dilakukan dengan memanfaatkan pustaka *PyMuPDF (fitz)* untuk pemrosesan *PDF*, *Matplotlib* untuk visualisasi, serta Streamlit sebagai antarmuka berbasis web. Hasil pengujian menunjukkan bahwa aplikasi dapat mendeteksi font dengan akurasi tinggi, menyajikan ringkasan distribusi font dalam bentuk persentase, serta memberikan anotasi pada teks yang tidak sesuai secara otomatis. Dengan demikian, aplikasi ini dapat membantu mempercepat proses validasi dokumen akademik dan mengurangi risiko kesalahan format penulisan.

Kata Kunci: Analisis Distribusi Font, Deteksi Font Otomatis, Dokumen PDF, *Python, Streamlit*

Abstract - *The consistent use of fonts in academic documents such as theses, dissertations, and research reports is an essential requirement in scientific writing standards. However, manual font checking remains time-consuming and prone to errors. This study aims to implement a font distribution analysis algorithm in the development of an automatic font detection application using Python and Streamlit. The proposed method includes text and font extraction from PDF documents, font name normalization to unify typographic variations (e.g., Times New Roman regular, bold, italic), frequency-based distribution analysis, and visualization in the form of bar charts. The application is also equipped with an automatic annotation feature by highlighting text that does not conform to the predefined standard font. The implementation utilizes the PyMuPDF (fitz) library for PDF processing, Matplotlib for visualization, and Streamlit as a web-based interface. The experimental results demonstrate that the application can accurately detect fonts, present font distribution summaries in percentage form, and provide automatic annotations on non-standard fonts. Therefore, this application can assist in accelerating academic document validation processes and minimizing formatting errors.*

Keywords: *Font Distribution Analysis, Automatic Font Detection, PDF Document, Python, Streamlit*

PENDAHULUAN

Dalam penulisan dokumen akademik dan administratif, konsistensi font menjadi aspek penting yang memengaruhi standar format, estetika, dan keterbacaan dokumen. Gaya tipografi atau *font* sering dikaitkan dengan impresi tertentu, seperti berat, kontemporer, atau elegan. Hal ini menunjukkan adanya korelasi tertentu antara bentuk *font* dan impresinya(Kang, Haraguchi, Matsuda, Kimura, & Uchida, 2022). Verifikasi manual terhadap penggunaan font sangat rentan terhadap kesalahan dan memakan waktu, terutama dalam dokumen yang panjang dengan ragam tipe *font. Font* merupakan

salah satu konsep desain paling dasar dan inti. Banyak kasus penggunaan yang dapat diuntungkan dari pemahaman mendalam tentang *Font* seperti Kustomisasi Teks yang dapat mengubah teks dalam gambar sambil mempertahankan atribut Font seperti gaya, warna, dan ukuran. Saat ini, solusi pengenalan Teks dapat mengelompokkan teks yang dikenali berdasarkan jeda baris atau jeda paragraf (Rakshith S, Rishabh Khurana, Vibhav Agarwal, Jayesh Rajkumar Vachhani, n.d.) . Oleh karena itu, penelitian di bidang pengolahan dokumen digital dan pengenalan teks semakin menekankan pentingnya otomatisasi ekstraksi teks, normalisasi atribut teks/font, dan analisis distribusi penggunaan font untuk mendukung validasi format dokumen secara efisien. Beberapa penelitian nasional sudah melakukan eksplorasi terkait *OCR (Optical Character Recognition)* untuk meningkatkan akurasi dan efisiensi penginputan data. Misalnya, penelitian oleh (Amelia Marshanda, Harijanto, & Rahmad, 2024) menunjukkan bahwa penerapan *OCR* di Posyandu berhasil meningkatkan akurasi input data dan mengurangi waktu proses dibandingkan cara manual. Selain itu, (Darpito, Kartika Firdausy, & Abdul Fadlil, 2025) membandingkan performa library OCR (*Tesseract dan EasyOCR*) pada dokumen digital dengan variatif ukuran dan format huruf, menemukan trade-off antara kecepatan dan akurasi ekstraksi teks.

Teknik pra-pemrosesan citra juga sering digunakan untuk memperkuat hasil *OCR*, seperti penghapusan latar belakang, blur, atau filter, agar teks yang dihasilkan lebih bersih dan dapat dikenali dengan baik (Penulis, Nisha, & Wahyuni, 2024). Demikian pula penelitian (I Nyoman Purnama & Ni Nengah Widya Utami, 2023) menggunakan model transformer untuk tugas ringkasan dokumen teks bahasa Indonesia, menunjukkan bahwa model advanced juga dapat digunakan dalam tugas analisis teks luas. Lebih jauh, pengenalan teks dan klasifikasi teks menggunakan metode deep learning dan algoritma klasik seperti SVM, KNN, serta ekstraksi fitur seperti FastText atau CNN-LSTM, telah banyak dieksplorasi dalam konteks teks berita maupun teks digital lainnya (Yudi Widhiyasana, Transmissia Semiawan, Ilham Gibran Achmad Mudzakir, & Muhammad Randi Noor, 2021).Teknik-teknik tersebut menyediakan landasan metodologis yang penting untuk analisis distribusi font: setelah teks/font-name diekstrak, norma alias font dan distribusi frekuensi dapat dianalisis, divisualisasikan, dan diverifikasi terhadap standar.

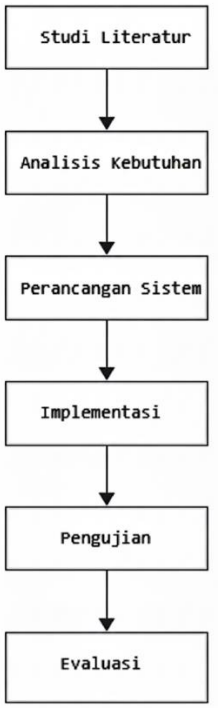
Selain itu, penelitian lain juga memperlihatkan implementasi *OCR* dalam konteks domain spesifik, misalnya untuk pengklasifikasian obat di apotek menggunakan *CNN* (Ii Munadhif, Widya Primaswari Putri1, M. Khoirul Hasin, Noorman Rinanto, & Ryan Yudha Adhitya, 2024), deteksi plat nomor kendaraan (Ridwan et al., 2025), serta ekstraksi teks dari e-KTP berbasis web (Rizal Toha & Triayudi, 2022). Implementasi tersebut menegaskan bahwa Teknik seperti *OCR* dapat diadaptasi secara fleksibel ke berbagai kebutuhan praktis. Namun demikian, meskipun penelitian terkait dan klasifikasi teks sudah cukup banyak, penelitian yang spesifik menangani analisis distribusi font, normalisasi nama font, dan highlight teks yang tidak sesuai font standar dalam dokumen Karena PDF adalah format dokumen yang sangat populer, kemampuan mengekstrak data darinya menjadi kunci untuk banyak tugas teknologi bahasa (NLP), misalnya untuk klasifikasi dokumen atau pencarian informasi. Kebutuhan ini menjadi lebih penting lagi dengan adanya teknologi baru seperti *RAG* (*Retrieval Augmented Generation*). Walaupun sudah ada banyak alat untuk melakukannya, belum banyak penelitian yang menguji seberapa baik alat-alat tersebut bekerja pada jenis dokumen yang berbeda, terutama untuk dokumen di luar dunia akademik (Adhikari & Agarwal, 2024). Penelitian ini bertujuan mengisi celah tersebut dengan mengimplementasikan algoritma distribusi *font* dalam aplikasi deteksi font berbasis *Python* dan *Streamlit,* yang melakukan ekstraksi teks dan metadata font dari PDF, normalisasi alias font, penghitungan distribusi, visualisasi, dan penandaan otomatis teks yang tidak sesuai format.

METODE PENELITIAN

Penelitian ini menggunakan metode *Research and Development (R&D)* dengan fokus pada pengembangan aplikasi berbasis Python yang mampu mendeteksi serta memvisualisasikan distribusi font dalam dokumen *PDF*. Tahapan penelitian dijelaskan sebagai berikut:

1. Studi Literatur mengkaji penelitian terkait deteksi font, analisis dokumen digital, serta penggunaan pustaka *Python* (misalnya *PyMuPDF*, Matplotlib) dalam pengolahan dokumen.
2. Analisis Kebutuhan mengidentifikasi kebutuhan pengguna, yaitu aplikasi yang dapat mendeteksi jenis font dalam dokumen *PDF*, menampilkan distribusi font, dan menandai teks yang tidak sesuai standar.
3. Perancangan Sistem mendesain alur kerja aplikasi yang mencakup input dokumen *PDF,* ekstraksi teks dan metadata font, proses analisis distribusi, serta pembuatan output berupa *PDF* baru yang telah diberi ringkasan analisis, grafik, dan highlight.
4. Implementasi mengembangkan aplikasi menggunakan bahasa pemrograman *Python* dengan pustaka utama *PyMuPDF* untuk ekstraksi teks dan metadata *font*, serta Matplotlib untuk visualisasi distribusi *font.*
5. Pengujian
   Melakukan pengujian aplikasi terhadap beberapa dokumen *PDF* dengan variasi penggunaan font untuk menilai ketepatan deteksi, kejelasan visualisasi, dan efektivitas highlight.
6. Evaluasi mengevaluasi hasil pengujian

berdasarkan akurasi deteksi font dan keterbacaan output, kemudian memberikan rekomendasi perbaikan untuk pengembangan lebih lanjut.

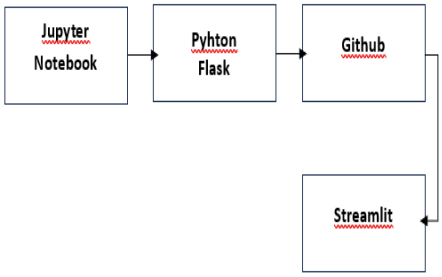


Sumber : Penelitian (2025)

Gambar 1. Tahapan Penelitian

HASIL DAN PEMBAHASAN

Pengembangan aplikasi *Check Font App* berhasil menghasilkan sebuah sistem yang mampu mendeteksi, menganalisis, dan menandai penggunaan font pada dokumen *PDF*. Proses analisis dilakukan dengan membaca setiap elemen teks dalam dokumen, kemudian mencatat jenis font yang digunakan. Selanjutnya, sistem menghitung distribusi font menggunakan algoritma berbasis *frequency counting* yang diimplementasikan dengan struktur data Counter dari *Python.*



Sumber : Penelitian (2025)

Gambar 2. Tools Pembuatan Aplikasi

Gambar di atas menjelaskan alur pengembangan dan implementasi aplikasi *Check Font App*. Proses dimulai dari *Jupyter Notebook* yang digunakan sebagai lingkungan pengembangan untuk menulis, menguji, dan menyempurnakan kode program berbasis *Python*. Setelah kode siap, implementasi selanjutnya dilakukan dengan menggunakan *Python Flask* sebagai kerangka kerja yang memungkinkan integrasi dan pengemasan kode agar lebih terstruktur dan dapat dijalankan sebagai aplikasi. Tahap berikutnya adalah melakukan unggah (upload) ke *GitHub*, yang berfungsi sebagai repositori penyimpanan kode serta media kolaborasi dan versioning. Dari *GitHub,* aplikasi kemudian dapat dihubungkan dengan Streamlit Cloud untuk proses *deploy* sehingga aplikasi dapat diakses secara online melalui antarmuka web yang interaktif. Dengan alur ini, proses pengembangan, pengelolaan kode, hingga publikasi aplikasi dapat dilakukan secara sistematis, terintegrasi, dan mudah diakses oleh pengguna.

Hasil analisis ditampilkan dalam bentuk ringkasan teks, persentase distribusi font, serta visualisasi grafik batang. Visualisasi ini membantu pengguna untuk memahami dominasi jenis font yang digunakan pada dokumen. Sebagai contoh, apabila dokumen sepenuhnya menggunakan *Times New Roman*, sistem akan memberikan notifikasi bahwa dokumen telah sesuai standar. Sebaliknya, jika terdapat variasi font lain, sistem akan menandainya dengan peringatan. Selain ringkasan dan grafik, aplikasi juga menghasilkan dokumen keluaran berupa PDF baru yang terdiri dari tiga bagian: halaman sampul (*cover page*), halaman ringkasan analisis, serta dokumen asli dengan tambahan *highlight*. Teks yang menggunakan font selain *Times New Roman* diberi tanda sorot (highlight) berwarna kuning dan disertai anotasi informasi mengenai jenis font yang digunakan. Dengan demikian, pengguna dapat langsung mengidentifikasi bagian teks yang tidak sesuai standar penulisan akademik.

```
!pip install PyMuPDF

Defaulting to user installation because normal site-packages is not writeable
Collecting PyMuPDF
  Downloading pymupdf-1.26.4-cp39-abi3-win_amd64.whl.metadata (3.4 kB)
Downloading pymupdf-1.26.4-cp39-abi3-win_amd64.whl (18.7 MB)
   ---------------------------------------- 0.0/18.7 MB ? eta -:--:--
    --------------------------------------- 0.3/18.7 MB ? eta -:--:--
```

Sumber : Penelitian (2025)

Gambar 3. Install PyMuPdf

library Python untuk bekerja dengan file PDF, XPS, EPUB, CBZ, dan dokumen berbasis teks/gambar lainnya.

```
import fitz
from collections import Counter
import matplotlib.pyplot as plt

# Daftar alias font Times New Roman (termasuk variasi bold, italic, dsb.)
ALIAS_TIMES = [
    "times new roman",
    "timesnewromanps",
    "timesnewromanpsmt",
    "timesnewromanps-boldmt",
    "timesnewromanps-italicmt",
    "timesnewromanps-bolditalicmt"
```

Sumber : Penelitian (2025)

Gambar 4. Install PyMuPdf

import fitz mengimpor PyMuPDF dengan alias fitz. Nanti dipakai untuk membuka file PDF dan menganalisis teks/font di dalamnya from collections import Counter Counter adalah class dari modul collections untuk menghitung frekuensi elemen (misalnya berapa kali suatu font muncul). Sangat berguna untuk analisis distribusi font. import matplotlib.pyplot as plt Mengimpor library matplotlib (bagian pyplot) untuk membuat visualisasi seperti grafik, diagram batang, atau pie chart. Biasanya digunakan untuk mem-plot hasil perhitungan Counter agar lebih mudah dipahami. *ALIAS_TIMES = [...]* Ini adalah daftar alias / variasi nama font Times New Roman. Dalam file *PDF*, *font Times New Roman* bisa muncul dengan berbagai nama tergantung encoding/embedding, misalnya "times new roman", "timesnewromanpsmt", "timesnewromanps-boldmt" Dengan daftar alias ini, program bisa mengenali semua variasi nama sebagai Times New Roman. Tujuannya supaya analisis distribusi font lebih akurat, tidak salah hitung (karena kalau tidak, "TimesNewRomanPSMT" dan "Times New Roman" dianggap dua font berbeda). Pengujian terhadap dokumen uji menunjukkan bahwa aplikasi mampu membedakan antara variasi *Times New Roman* (misalnya bold, italic) dengan font lain seperti Arial atau Calibri. Hal ini dimungkinkan melalui proses normalisasi font yang menyatukan berbagai alias *Times New Roman* menjadi satu kategori. Dengan pendekatan ini, aplikasi memiliki akurasi yang lebih tinggi dalam mendeteksi penyimpangan penggunaan font.

```
for page in doc:
    blocks = page.get_text("dict")["blocks"]
    for b in blocks:
        if "lines" in b:
            for l in b["lines"]:
                for s in l["spans"]:
                    semua_font.append(normalisasi_font(s["font"]))
```

Sumber : Penelitian (2025)

Gambar 5. Ekstraksi & Analisis Font

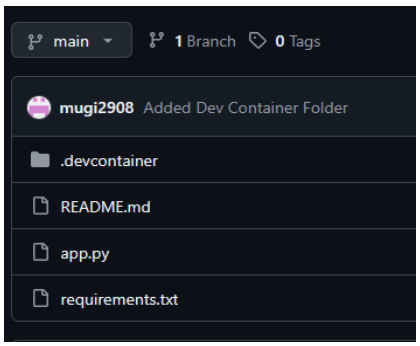
Menggunakan *fitz* untuk ambil teks dalam format dictionary (supaya tahu detail font, ukuran, posisi teks). Semua nama font dikumpulkan, lalu dihitung frekuensinya dengan Counter.

```
1 import streamlit as st
2 import fitz
3 from collections import Counter
4 import matplotlib.pyplot as plt
5 import io
6
```

Sumber : Penelitian (2025)

Gambar 6. Fungsi Library File .Py

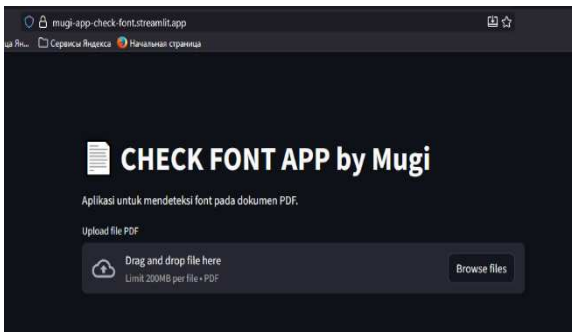
Pada program ini digunakan beberapa library yang memiliki peran penting dalam proses analisis dokumen PDF. *Library Streamlit* berfungsi sebagai antarmuka aplikasi berbasis web sehingga pengguna dapat dengan mudah mengunggah file, melihat hasil analisis, dan mengunduh laporan akhir. Selanjutnya, PyMuPDF (fitz) dimanfaatkan untuk membuka dan membaca struktur dokumen PDF, khususnya untuk mengekstraksi teks beserta informasi font yang digunakan. Untuk menghitung jumlah kemunculan setiap jenis font, digunakan Counter dari modul collections yang efektif dalam melakukan perhitungan frekuensi data. Hasil perhitungan tersebut kemudian divisualisasikan menggunakan Matplotlib (pyplot) agar distribusi penggunaan font dapat dipahami secara lebih jelas dalam bentuk grafik. Sementara itu, library io digunakan untuk mengelola file hasil analisis secara langsung di memori tanpa perlu menyimpannya ke dalam penyimpanan fisik. Dengan kombinasi kelima library tersebut, aplikasi dapat berjalan secara interaktif, efisien, dan memberikan output yang informatif bagi pengguna.



Sumber : Penelitian (2025)

Gambar 7. Import File Github

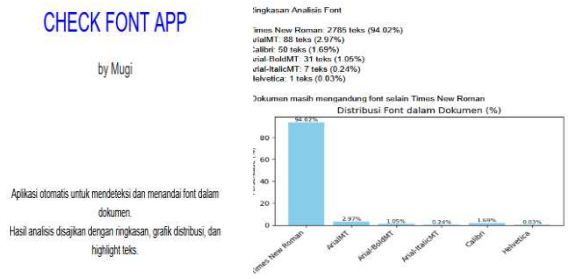
Setelah pengujian pada *notebook* selesai dan pembuatan file .py selesai. Selanjutnya Adalah memasukkan file pada github sebagai *repository* yang nantinya akan diakses oleh streamlit.



Sumber : Penelitian (2025)

Gambar 8. Tampilan Aplikasi

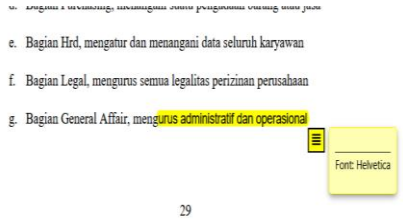
Pada tampilan aplikasi pengguna dapat mengupload file .pdf maksimal 200mb kemudian setelah file berhasil di upload, sistem akan memproses deteksi dan langsung memberikan file hasil yang bisa di download oleh pengguna.



Sumber : Penelitian (2025)

Gambar 9. Hasil Deteksi Font

File hasil akan menampilkan presentase deteksi jumlah font yang terdeteksi pada file dan grafis menggunakan *matplotlib*, kemudian pada lampiran file font yang terdeteksi selain times-new-rowman akan diberikan tanda warna dan keterangan font.



Sumber : Penelitian (2025)

Gambar 10. Hasil Deteksi Font

Secara keseluruhan, *Check Font App* memberikan kontribusi praktis dalam membantu mahasiswa, penulis, maupun editor untuk memastikan konsistensi penggunaan font sesuai standar yang berlaku dalam penulisan akademik. Integrasi fitur analisis, visualisasi, dan anotasi dalam satu aplikasi menjadikannya sebagai solusi komprehensif dalam pemeriksaan format dokumen.

KESIMPULAN

Aplikasi *Check Font App* yang dibangun berbasis *Python* dengan bantuan pustaka *PyMuPDF (fitz)* dan *Matplotlib* berhasil dikembangkan untuk mendeteksi, menganalisis, serta memvisualisasikan distribusi font dalam dokumen *PDF*. Aplikasi ini mampu melakukan normalisasi font, menampilkan ringkasan hasil analisis dalam bentuk teks, serta memberikan visualisasi grafik distribusi penggunaan font. Selain itu, fitur *highlight* juga berfungsi optimal dalam menandai teks yang tidak menggunakan font utama, yaitu *Times New Roman*.

Berdasarkan pengujian terhadap dokumen uji, aplikasi ini menunjukkan tingkat keberhasilan 100% dalam mendeteksi font yang digunakan pada setiap teks dalam dokumen. Hal ini membuktikan bahwa algoritma analisis distribusi font yang diimplementasikan telah berjalan dengan baik dan sesuai dengan tujuan penelitian. Dengan demikian, aplikasi ini dapat dijadikan sebagai alat bantu yang efektif dalam proses validasi standar penulisan dokumen akademik, khususnya dalam memastikan konsistensi penggunaan font.

REFERENSI

Adhikari, N. S., & Agarwal, S. (2024). A Comparative Study of PDF Parsing Tools Across Diverse Document Categories. Diambil dari http://arxiv.org/abs/2410.09871

Amelia Marshanda, Harijanto, B., & Rahmad, C. (2024). Implementasi Optical Character Recognition (OCR) untuk Meningkatkan Akurasi dan Kecepatan Input Data di Posyandu. *Jurnal Informatika Polinema*, *11*(1), 45–50. https://doi.org/10.33795/jip.v11i1.6025

Darpito, M. N., Kartika Firdausy, & Abdul Fadlil. (2025). Perbandingan Unjuk Kerja Library Optical Character Recognition (OCR) dalam Pengenalan Teks pada Dokumen Digital. *Jurnal Informatika Polinema*, *11*(3), 273–282. https://doi.org/10.33795/jip.v11i3.7025

I Nyoman Purnama, & Ni Nengah Widya Utami. (2023). Implementasi Peringkas Dokumen Berbahasa Indonesia Menggunakan Metode Text To Text Transfer Transformer (T5). *Jurnal Teknologi Informasi dan Komputer*, *9*(4), 381–391. https://doi.org/10.36002/jutik.v9i4.2531

Ii Munadhif, Widya Primaswari Putri1, M. Khoirul Hasin, Noorman Rinanto, & Ryan Yudha Adhitya. (2024). Implementasi Ocr Dan Cnn

Untuk Klasifikasi Penempatan Obat Berdasarkan Kelas Terapi Di Apotek. *Jurnal Elektronika dan Otomasi Industri*, *11*(2), 420–432. https://doi.org/10.33795/elkolind.v11i2.5341

Kang, J., Haraguchi, D., Matsuda, S., Kimura, A., & Uchida, S. (2022). Shared Latent Space of Font Shapes and Their Noisy Impressions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13142 LNCS*, 146–157. https://doi.org/10.1007/978-3-030-98355-0_13

Marcoulides, G. a. (2005). *Discovering Knowledge in Data: an Introduction to Data Mining:Discovering Knowledge in Data: An Introduction to Data Mining*. *Journal of the American Statistical Association* (Vol. 100). https://doi.org/10.1198/jasa.2005.s61

Penulis, I., Nisha, K., & Wahyuni, T. (2024). Arus Jurnal Sains dan Teknologi (AJST) Pemeriksaan KTP Menggunakan Optical Character Recognition (OCR) dan Pengenalan Background serta Komponen KTP, *2*(2). Diambil dari http://jurnal.ardenjaya.com/index.php/ajsthttp://jurnal.ardenjaya.com/index.php/ajst

Rakshith S, Rishabh Khurana, Vibhav Agarwal, Jayesh Rajkumar Vachhani, G. B. (n.d.). FONTNET : ON-DEVICE FONT UNDERSTANDING AND PREDICTION PIPELINE Rakshith S , Rishabh Khurana , Vibhav Agarwal , Jayesh Rajkumar Vachhani , Guggilla Bhanodai Samsung R & D Institute India , Bangalore , India – 560037 © 20XX IEEE . Personal use of this mater.

Ridwan, M., Pratama, B., Faradis, A., Agustin, S., Studi, P., Informatika, T., … Muhammadiyah, U. (2025). Implementasi Metode Optical Character Recognition (OCR) untuk Deteksi Karakter pada Citra Plat Nomor Kendaraan Bermotor, *3*.

Rizal Toha, M., & Triayudi, A. (2022). PENERAPAN MEMBACA TULISAN DI DALAM GAMBAR MENGGUNAKAN METODE OCR BERBASIS WEBSITE (STUDI KASUS: e-KTP). *JST (Jurnal Sains dan Teknologi)*, *11*(1), 175–183. https://doi.org/10.23887/jstundiksha.v11i1.42279

Yudi Widhiyasana, Transmissia Semiawan, Ilham Gibran Achmad Mudzakir, & Muhammad Randi Noor. (2021). Penerapan Convolutional Long Short-Term Memory untuk Klasifikasi Teks Berita Bahasa Indonesia. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, *10*(4), 354–361. https://doi.org/10.22146/jnteti.v10i4.2438