Vol 8 No 2 (2025):
JURNAL PENDIDIKAN TEKNOLOGI INFORMASI
(JUKANTI) EDISI NOPEMBER 2025

PENERAPAN OCR UNTUK DIGITALISASI DAN PENGARSIPAN DATA DIGITAL

Justin Clarence Setiawan[a], Ariya Dwika Cahyono, S. Kom, M.T.[b]

[a,b]Fakultas Teknik Informatika, Universitas Kristen Satya Wacana, Jawa Tengah

[a]672021030@student.uksw.edu, [b]ariyadc@uksw.edu

ABSTRAK

Penelitian dalam bidang Optical Character Recognition (OCR) saat ini didorong oleh kemajuan pesat machine learning, ketersediaan dataset yang besar dan kebutuhan akan pemrosesan informasi yang efisien. Penelitian ini berhasil membangun sebuah sistem tertutup untuk mendigitalisasi dan mengarsipkan data dengan mengekstrak teks dari gambar dan file PDF yang terintegrasi dengan database. Sistem dirancang dengan koneksi lokal (LAN) dan dihosting pada localhost untuk menjaga keamanan data sensitif. Inti dari proses digitalisasi ini adalah teknologi OCR dengan mengimplementasikan library Tesseract. Untuk file PDF, proses ekstraksi dibantu oleh library PDF.js yang pertama-tama mengonversi PDF menjadi gambar. Pengujian akurasi sistem dilakukan pada dua jenis dokumen. Hasilnya menunjukkan performa yang sangat tinggi untuk dokumen yang diketik komputer dengan nilai Character Error Rate (CER) 0,16%, di mana kesalahan yang terjadi bersifat minor dan sporadis akibat ambiguitas visual font dan noise pada dokumen. Di sisi lain, kinerja pada dokumen mesin ketik menunjukkan hasil yang memadai namun belum maksimal dengan CER 6,41%. Tingginya error substitusi pada dokumen ini diduga kuat disebabkan oleh kualitas fisik dokumen yang memudar, hasil scan yang buram, karakteristik font mesin ketik yang khas, serta gangguan dari stempel atau tanda tangan. Secara keseluruhan, sistem ini terbukti sangat efektif untuk digitalisasi dokumen komputer dan cukup baik untuk dokumen mesin ketik, sehingga dapat sangat mempermudah proses pengarsipan data yang terorganisir dan aman.
Kata Kunci: *Data Digital, OCR, Pengarsipan Data*

ABSTRACT

Research in the field of Optical Character Recognition (OCR) is currently driven by rapid advances in machine learning, the availability of large datasets, and the need for efficient information processing. This research has successfully developed a closed system for digitizing and archiving data by extracting text from images and PDF files integrated with a database. The system is designed with a local connection (LAN) and hosted on localhost to maintain the security of sensitive data. At the core of this digitization process is OCR technology, which implements the Tesseract library. For PDF files, the extraction process is aided by the PDF.js library, which first converts PDFs into images. Accuracy testing of the system was conducted on two types of documents. The results showed very high performance for computer-typed documents with a Character Error Rate (CER) of 0.16%, where the errors that occurred were minor and sporadic due to visual font ambiguity and noise in the documents. On the other hand, performance on typewritten documents showed adequate but not optimal results with a CER of 6.41%. The high substitution error rate in these documents is strongly suspected to be caused by the faded physical quality of the documents, blurred scan results, the distinctive characteristics of typewritten fonts, and interference from stamps or signatures. Overall, this system has proven to be very effective for digitizing computer documents and quite good for typewritten documents, thereby greatly facilitating the process of organized and secure data archiving.
Keywords: *Digital Data, OCR, Data Archiving*

1. PENDAHULUAN

Teknologi informasi berkembang dengan sangat cepat, terutama teknologi pengolahan data. Pengolahan data yang dulu hanya berupa pembukuan data fisik dan dilakukan secara manual, kini berkembang menjadi data digital yang dapat diolah menggunakan komputer. Semua proses pengolahan data secara otomatis saat ini tidak lepas dari perkembangan *artificial intelligence* (AI) dan *machine learning*. Pengolahan data dapat dilakukan secara otomatis oleh model *machine learning* yang sebelumnya sudah dilatih untuk mengolah data tanpa instruksi dan campur tangan manusia. Salah satu contoh penggunaan *machine learning* adalah mengubah data dari gambar menjadi teks untuk mendigitalisasi data.

Proses mengubah data dari gambar menjadi teks ini dapat dilakukan karena adanya *Optical Character Recognition* (OCR). OCR merupakan sebuah metode untuk melakukan ekstraksi teks dari sebuah dokumen yang paling banyak digunakan saat ini [1]. OCR akan mengkonversi huruf yang terdapat di dalam gambar menjadi karakter *American Standard Code for Information Interchange* (ASCII) sehingga komputer dapat mengenali dan memproses data yang diperoleh [2].

Untuk menjalankan tugasnya, sistem OCR menggunakan dua modul utama, yaitu modul deteksi teks dan modul pengenalan teks. Modul deteksi teks bertujuan untuk melokalisasi semua blok teks yang terdapat dalam gambar, baik pada tingkat kata maupun tingkat baris teks. Sedangkan modul pengenalan teks bertujuan untuk memahami dan mentranskripsikan sinyal visual dari gambar [3].

Penelitian dalam bidang OCR saat ini didorong oleh kemajuan pesat *deep learning*, ketersediaan dataset yang besar dan kebutuhan akan pemrosesan informasi yang efisien. OCR modern telah jauh melampaui metode tradisional yang berbasis fitur dan *template matching*. Bahkan penelitian OCR telah meluas ke pengenalan teks dalam gambar dan video, termasuk teks alami misalnya rambu jalan, teks pada produk dan caption video. Contohnya adalah penggunaan OCR untuk memeriksa KTP [4] dan pengenalan tulisan pada nota pembelian material [5].

Selain itu, sebuah perusahaan atau lembaga tentunya memiliki ratusan bahkan hingga ribuan data sehingga penggunaan website dinilai kurang efisien. Proses penyimpanan data juga akan sangat sulit karena semakin banyak data maka akan semakin sulit untuk mencari sebuah data spesifik, terutama data fisik. Di sinilah peran dari data digital. Data digital merupakan informasi berupa serangkaian simbol diskrit, di mana setiap simbol dapat mengambil salah satu dari sejumlah nilai terbatas dari sebuah huruf atau angka. Data digital merepresentasikan informasi menggunakan nilai biner diskrit (0 dan 1), yang memungkinkan penyimpanan, pemrosesan dan transmisi melalui sistem elektronik. Data digital memiliki banyak sekali kegunaan, misalnya digitalisasi data spasial desa, seperti peta dan data wilayah [6]. Dengan adanya data digital, semua data fisik cukup didigitalisasi ke dalam sistem komputer.

Sama halnya dengan data fisik, data digital juga memerlukan sebuah sistem pengarsipan agar data tetap terorganisir. Pengarsipan data digital atau yang lebih dikenal dengan arsip digital merupakan kumpulan data berbentuk dokumen

elektronik yang disimpan dan diproses menggunakan teknologi komputer. Dokumen elektronik ini kemudian dapat dilihat dan dipergunakan kembali. Beberapa contoh arsip elektronik yang sering dipakai dalam kehidupan sehari-hari seperti gambar, surat elektronik (*e-mail*), CD (*compact disc*) dan berbagai hasil proses digital lainnya [7]. Pemanfaatan pengarsipan data digital ini telah merevolusi cara organisasi beroperasi. Sistem pengarsipan digital menawarkan keunggulan signifikan dibandingkan pengarsipan data fisik/manual. Hal ini dikarenakan data fisik sangat rentan terhadap kerusakan dan kehilangan [8]. Keuntungan lainnya dari pengarsipan data digital adalah pengambilan file yang cepat, lebih mudah membagikan file dan tingkat keamanan yang sangat tinggi [9]. Selain itu, pengarsipan data digital mempermudah akses data, mempersingkat pencarian data, mengurangi biaya penyimpanan fisik, peningkatan keamanan data serta mempermudah pemenuhan standar hukum dan regulasi penyimpanan data [10].

Penelitian tentang penerapan Optical Character Recognition (OCR), digitalisasi dan pengarsipan memang telah berkembang dengan pesat. Namun, sebagian besar kajian hanya berfokus pada satu aspek tertentu, misalnya hanya pada OCR, digitalisasi atau pengarsipan saja. Sebagai contoh, penelitian yang dilakukan oleh Nisha dkk. mengenai Pemeriksaan KTP Menggunakan Optical Character Recognition (OCR) dan Pengenalan Background serta Komponen KTP hanya berfokus pada implementasi OCR [4].

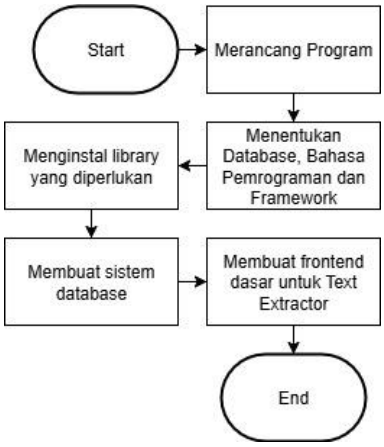
Sementara itu, kebutuhan di lapangan justru memerlukan suatu sistem terpadu yang tidak hanya mampu mengonversi dokumen fisik ke bentuk digital , tetapi juga dapat mengategorikan, mengindeks, serta mengarsipkannya secara efisien dan otomatis. Kesenjangan ini menunjukkan adanya kebutuhan akan pengembangan solusi yang dapat mengotomatiskan seluruh alur kerja mulai dari input dokumen fisik, pemrosesan, digitalisasi, hingga pengarsipan digital yang terorganisir, akurat, cepat dan efisien dengan tetap menjaga keamanan data [11].

2. METODE PENELITIAN

Metode penelitian dibagi menjadi tiga, yaitu analisis, implementasi dan pengujian. Proses analisis dilakukan untuk mengetahui kebutuhan sistem, yang kemudian akan diimplementasikan pada program. Selanjutnya, melakukan pengujian terhadap sistem yang digunakan.

Program dibuat menggunakan framework Next JS untuk *frontend* dan Laravel untuk *backend*. Bahasa pemrograman yang digunakan adalah TypeScript XML untuk *frontend* dan php untuk *backend*. Sistem database untuk software ini dibuat menggunakan database Postgre SQL.

Langkah pertama adalah melakukan analisis sistem. Proses analisis ini bertujuan untuk mengetahui kebutuhan dalam membuat sistem digitalisasi dan pengarsipan data. Proses analisis secara keseluruhan dijelaskan pada Gambar 1.



Gambar 1. Alur Analisis Sistem

Rancangan antarmuka untuk program terdiri beberapa halaman. Halaman pertama merupakan halaman *login*. Halaman ini berisi *textbox* untuk memasukkan *username* dan *password* serta tombol *login*.

Selanjutnya ada halaman ekstrak teks yang dapat menerima masukan berupa file image atau pdf, lalu menampilkan hasil ekstrak teksnya. Kemudian, terdapat tombol untuk download dan tombol untuk menyimpan teks hasil ekstrak.

Fungsi OCR untuk ekstrak teks dari gambar akan dijalankan menggunakan *library* Tesseract OCR. *Tesseract* merupakan mesin dan *library Optical Character Recognition* (OCR) *open source* yang dikembangkan oleh Google. *Tesseract* berfungsi untuk membaca atau mengekstrak teks dari sebuah citra dengan menggunakan metode OCR berada di bawah lisensi Apache 2.0 [12]. *Tesseract* awalnya dibuat menggunakan Bahasa Pemrograman C++. Seiring berjalannya waktu, diciptakanlah *library* dan *framework* tambahan dengan menggunakan bahasa pemrograman lain seperti Flutter. *Library* dan *framework* ini dibuat sebagai jembatan antara *Tesseract* dengan bahasa pemrograman tersebut sehingga *library Tesseract* dapat digunakan oleh berbagai macam bahasa pemrograman hingga platform atau perangkat selain PC seperti Android [13].

Proses pengenalan karakter yang dilakukan oleh Tesseract dibagi ke dalam tiga tahap. Tahap pertama adalah tahap *Convolutional Feature Extraction.* Dalam tahap ini, gambar input akan diproses oleh beberapa *Convolutional Neural Network* (CNN) untuk mengekstraksi *feature map* atau output dari proses CNN yang representatif. *Feature map* ini kemudian diinterpretasikan sebagai urutan *feature vectors* (kolom-kolom dari *feature map*).

Tahap selanjutnya adalah *Sequence Modeling* dengan LSTM, di mana urutan feature vectors ini kemudian dimasukkan ke dalam jaringan *Bidirectional* LSTM (Bi-LSTM). Hal ini memungkinkan konteks dari kedua arah (kiri dan kanan) digunakan untuk memprediksi setiap karakter, sehingga dapat membantu mengenali karakter yang ambigu dengan melihat karakter di sekitarnya.

Tahap terakhir adalah *decoding*. Pada tahap ini, Tesseract menggunakan algoritma CTC *Beam Search* untuk mencari urutan karakter yang paling mungkin, dengan mempertimbangkan probabilitas dari LSTM dan juga model bahasa (seperti kata dalam bahasa tertentu) untuk memperbaiki hasilnya. CTC akan menghitung probabilitas suatu label dengan menjumlahkan probabilitas semua kemungkinan *alignment paths* yang dapat dipetakan ke label dengan menghilangkan karakter kosong, "-" dan pengulangan menggunakan mekanisme CTC *Loss*. CTC *Loss* adalah mekanisme yang menangani masalah alignment antara urutan input (lebih panjang) dan urutan output (lebih

pendek). Mekanisme CTC *Loss* bekerja dengan formula berikut [14].

$$p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x)\text{b} \quad (1)$$

x = *input*
l = *output*
π = *alignment path*
B = fungsi pemetaan

Untuk file dengan format PDF, Tesseract OCR akan dibantu *library* PDFjs. PDFjs adalah *library* open-source JavaScript yang dikembangkan oleh Mozilla yang menyajikan berkas PDF (Portable Document Format) secara langsung di peramban web tanpa memerlukan *plugin* seperti Adobe Acrobat. *Library* ini memungkinkan file yang awalnya berformat PDF diubah menjadi file yang berformat *image* sehingga *library* Tesseract OCR dapat melakukan proses ekstrak teks terhadap file.

Langkah selanjutnya adalah menginstal semua *library* yang diperlukan untuk membuat program. *Library* ini akan menunjang framework yang digunakan sehingga program dapat melakukan berbagai macam fungsi. *Library* yang digunakan adalah React, ReactCrop, pdfjsLib, argparse, Tesseract, json, sys, traceback.

Proses selanjutnya adalah analisis rancangan database. Database yang akan dipakai adalah database PostgreSQL. PostgreSQL merupakan sebuah *Object Relational Database Management System* (ORDBMS) *open-source*. PostgreSQL mendukung penggunaan bahasa pemrograman SQL secara luas. Hal ini didukung dengan ketersediaan fitur yang banyak [15]. PostgreSQL dipilih karena dapat mendukung berbagai macam format data, terutama JSON dan XML yang digunakan dalam program ini. Selain itu, PostgreSQL juga sangat mendukung penggunaan *query* JSON.

Agar program dapat membaca kolom pada database, diperlukan model data yang sesuai dengan tabel pada database. Model yang diperlukan program ini ada dua, yaitu model file dan model user.

Model file berisi variabel *Name* yang berisi nama file yang disimpan ke database, *Path* yang berisi path menuju storage tempat file disimpan, *Type* yang berisi format file yang tersimpan dan terakhir *Size* yang berisi ukuran file yang tersimpan.

Model file berisi variabel *Username* yang berisi nama pengguna yang terdaftar di database dan *Password* yang berisi *password* untuk *username* yang bersangkutan. *Password* yang tersimpan di dalam database sudah dienkripsi menggunakan hash agar sulit diketahui. *Password* juga tidak akan tampil pada *console log* maupun *output* JSON dan terakhir *Api Token* yang berisi token yang akan diberikan kepada pengguna bila pengguna melakukan *login*. Token ini berfungsi untuk memastikan bahwa pengguna yang sudah *logout* tidak dapat masuk kembali ke dalam program tanpa melalui proses *login* terlebih dahulu.

Sistem database ini memungkinkan pengguna menyimpan dan mengakses file yang disimpan. Sistem ini dibuat dalam sebuah sistem tertutup, di mana hanya pengguna yang menggunakan perangkat komputer yang memiliki software ini. Tujuan dibuatnya sistem ini adalah agar *pengguna* dapat dengan leluasa mengunggah dan memakai file rahasia/sensitif tanpa harus takut bila ada orang atau pihak lain yang akan mengetahuinya.

Oleh karena itu, program ini dibuat dalam sebuah sistem tertutup dengan koneksi local (LAN). *Local Area Network* (LAN) merupakan jaringan komputer yang menghubungkan perangkat dalam area geografis terbatas. LAN biasanya digunakan di rumah, sekolah, dan kantor. Teknologi yang digunakan pada LAN saat ini berbasis pada teknologi IEEE 802.3 Ethernet yang menggunakan perangkat switch dan mempunyai kecepatan transfer data 10, 100 atau 1000 Mbit/s [16].

Tujuan penggunaan jaringan LAN adalah untuk membatasi akses ke database sehingga hanya perangkat yang terhubung ke jaringan LAN yang sama yang dapat mengakses program dan database. Meskipun memiliki akses ke perangkat tersebut, tidak semua orang dapat mengakses data di dalam database karena akses ke program dan data di dalam database hanya diberikan kepada pengguna yang terdaftar di database saja. *Password* milik pengguna yang tersimpan di dalam database juga telah melalui proses enkripsi atau hashing menggunakan fungsi bcrypt sehingga sulit untuk diketahui.

Selain itu, sistem *login* juga menggunakan authentication token sehingga pengguna harus melalui proses *login* terlebih dahulu untuk menggunakan software dan jika pengguna sudah *logout*, maka pengguna tidak dapat kembali ke halaman sebelumnya. Jadi, pengguna harus *login* kembali untuk menggunakan software

Proses selanjutnya adalah pemberian *idle time* untuk program. Jika pengguna tidak melakukan apapun selama 5 menit, maka fungsi akan langsung menghapus token dan mengeluarkan *pengguna* dari program. Jika pengguna melakukan sesuatu seperti menggerakkan kursor, maka *timer* akan diatur ulang.

Terakhir adalah proses pengujian. Pengujian Text Extractor dilakukan secara manual. Terdapat dua buah file untuk diekstrak. Satu file diketik menggunakan komputer, sedangkan file yang lain diketik menggunakan mesin ketik. Selain itu, telah disiapkan juga hasil ekstrak yang diharapkan dari kedua file yang akan menjadi acuan dalam pengukuran tingkat akurasi program. Hasil ekstrak dari kedua file akan dibandingkan dengan file referensi, kemudian tingkat akurasinya akan dihitung menggunakan rumus *Character Error Rate* (CER). CER adalah metrik evaluasi yang membandingkan karakter yang ada pada *plaintext* dengan teks referensi [17]. Rumus CER adalah sebagai berikut [18]:

$$CER = \frac{S+D+I}{N} \times 100\% \qquad (2)$$

S = Jumlah substitusi (karakter yang salah dikenali).

D = Jumlah penghapusan (karakter yang terhapus atau tidak dikenali).

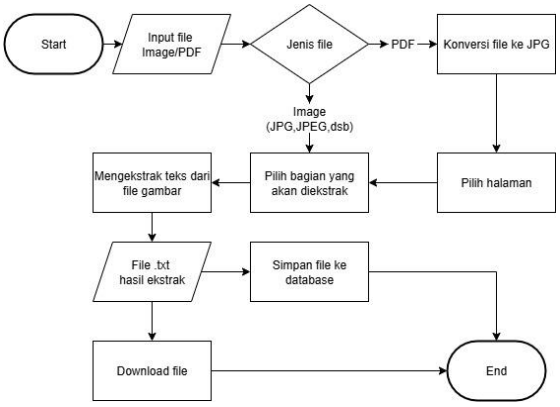
I = Jumlah penyisipan (karakter tambahan yang tidak seharusnya ada).

N = Total jumlah karakter dalam teks referensi yang benar.

Hasil yang diharapkan adalah tidak lebih dari 5% CER, karena menurut para ahli 5% CER sudah cukup bagus untuk kepentingan penelitian [19].

3. HASIL DAN PEMBAHASAN

Kebutuhan sistem program ekstraksi teks dari gambar dan pdf dijelaskan pada Gambar 2.



Gambar 2. Alur Ekstrak Teks

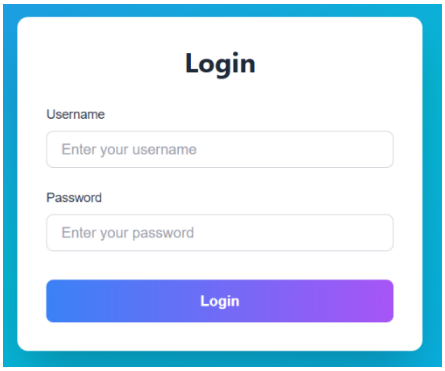
Awalnya program akan menerima masukan berupa file *image* atau PDF. Jika file yang diterima berupa gambar, maka gambar akan langsung ditampilkan pada preview. Jika file yang diterima berupa PDF, maka *library* PDFjs akan mengubah file dari format PDF ke format *image*. Jika ada dua halaman atau lebih dalam file PDF tersebut, maka setiap halaman dalam file tersebut akan diubah menjadi gambar dan akan muncul pada preview sehingga dapat dipilih.

Pada bagian preview, gambar dapat dipangkas agar hanya mengekstrak bagian yang dipilih saja. Setelah itu, program akan menjalankan proses ekstrak teks dari gambar, kemudian program akan menampilkan hasilnya. Setelah itu, file dapat diunduh atau disimpan ke dalam database. Implementasi sistem dijelaskan dalam Tabel 1.

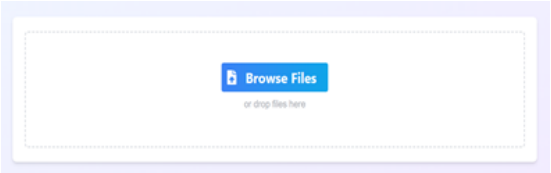
Tabel 1. Tampilan Interface

Nama Bagian	Keterangan
Login	Tampilan halaman *login*. Terdapat kolom yang akan diisi dengan *username* dan *password*. Tampilan dapat dilihat pada Gambar 3.
Drag Drop File	Bagian drag drop file. File yang akan diekstrak dapat diletakkan pada kotak yang tersedia. Tersedia juga tombol "Browse Files" untuk mencari langsung file di dalam perangkat. Tampilan dapat dilihat pada Gambar 4.
Preview dan Extract	Menampilkan preview dari file yang akan diekstrak. Pada bagian ini juga, file dapat dipangkas untuk memilih bagian yang akan diekstrak. Terdapat dua tombol, yaitu tombol "Extract Selected Part" untuk mengekstrak bagian yang dipilih dan "Extract All" untuk mengekstrak keseluruhan file. Tampilan dapat dilihat pada Gambar 5.
Hasil, Download dan Simpan ke Database	Menampilkan hasil ekstrak teks dari gambar atau file PDF. Terdapat dua tombol, yaitu tombol "Download as Text" dan "Save to Database". Tampilan dapat dilihat pada Gambar 6.

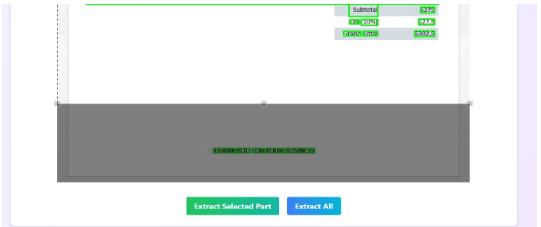
Berikut ini adalah tampilan dari masing - masing bagian.



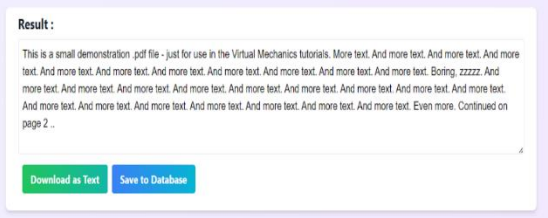
Gambar 3. *Login*



Gambar 4. Drag Drop File



Gambar 5. Preview



Gambar 6. Hasil

Selanjutnya adalah penjelasan mengenai cara kerja sistem. Setelah program menerima masukan berupa file *image*, maka Tesseract.js akan melakukan proses ekstrak teks dari image. Jika masukan file yang diterima berupa file PDF, maka *library* PDFjs akan mengubah file dari format PDF ke format *image.*

Cara kerja Tesseract OCR dimulai dengan proses ekstrak teks dilakukan oleh *library* Tesseract.js. Saat program dijalankan, *Worker* akan melakukan proses inisialisasi untuk membuat *thread* latar belakang *Worker* untuk mengurangi pemrosesan intensif CPU dan memuat data bahasa OCR (bahasa Inggris secara default) dari file data yang telah dilatih. Kode untuk mendefinisikan worker dapat dilihat pada Kode 1.

Kode 1 Mendefinisikan Worker

```
const worker = await
initializeWorker(); // Creates a Web
Worker
```

Selanjutnya, Tesseract akan memulai *preprocessing* gambar. *Preprocessing* yang dilakukan Tesseract di antaranya adalah menerapkan ambang batas adaptif untuk membuat gambar biner, melakukan analisis komponen yang terhubung untuk menemukan wilayah teks serta mendeteksi orientasi skrip dan arah penulisan.

Setelah itu, gambar akan diproses melalui jalur pengenalan Tesseract. Proses ini meliputi analisis tata letak untuk mendeteksi blok teks, segmentasi baris dan kata, pengenalan karakter dan pemodelan bahasa untuk meningkatkan akurasi.

Terakhir, program akan menampilkan hasil ekstrak dengan mengembalikan teks yang dikenali. Kemudian, teks akan ditampilkan pada bagian hasil ekstrak.

Proses selanjutnya adalah cara kerja PDFjs. Jika program menerima masukan file berupa PDF, maka *library* PDFjs akan mengubah format file menjadi image terlebih dahulu. Pertama, *library* PDFjs akan membuat URL sementara untuk berkas PDF menggunakan URL.createObjectURL(). URL ini merupakan lokasi disimpannya file yang diinput di dalam perangkat. Kemudian, PDFjs akan memuat dokumen PDF. Kode untuk memuat dokumen ke PDFjs dapat dilihat pada Kode 2.

Kode 2 Memuat Dokumen PDF ke PDFjs

```
const pdf = await
pdfjsLib.getDocument(URL.createObjectU
RL(file)).promise;
```

Selanjutnya, PDFjs akan mencari dan mendapatkan setiap halaman dalam file PDF. Kemudian PDFjs akan mengambil setiap objek yang terdapat di setiap halaman. Kode untuk mendapatkan semua halaman dalam dokumen dapat dilihat pada Kode 3.

Kode 3 Mendapatkan Semua Halaman pada File

```
for (let i = 1; i <= pdf.numPages;
i++) {
  const page = await pdf.getPage(i);
  // ... render page to canvas ...
}
```

Langkah selanjutnya adalah mengatur skala dan kualitas gambar yang akan dihasilkan. Viewport menggunakan skala tinggi (5x) untuk mendapatkan akurasi OCR yang lebih baik. Kemudian PDFjs akan membuat

elemen *canvas* baru dan mengatur dimensi *canvas* agar sesuai dengan ukuran halaman PDF yang diskalakan. Kode untuk membuat elemen *canvas* dapat dilihat pada Kode 4.

Kode 4 Membuat Elemen *Canvas*

```
const viewport = page.getViewport({
scale: 5 }); // High resolution for
OCR
const canvas =
document.createElement("canvas");
const context =
canvas.getContext("2d");
canvas.height = viewport.height;
canvas.width = viewport.width;
```

Selanjutnya, program akan memulai proses *rendering* gambar menggunakan metode *render* dari PDF.js untuk menggambar halaman PDF ke *canvas*. *Rendering* mencakup teks dan grafik vektor. Kode agar program memulai proses *rendering* dapat dilihat pada Kode 5.

Kode 5 *Rendering*

```
await page.render({
  canvasContext: context!,
  viewport
}).promise;
```

Setelah proses *rendering* selesai, program akan mengkonversi konten *canvas* ke URL gambar yang disandikan dengan *base64*. Hal ini penting karena *library* ReactCrop mengharapkan sumber gambar berupa URL. Gambar akan menggunakan format JPEG secara default. Terakhir, data gambar disimpan dalam bentuk array untuk digunakan nanti. Setelah proses mengubah format file dari PDF menjadi gambar selesai, program akan menjalankan proses ekstrak teks dari gambar oleh Tesseract OCR seperti yang telah dijelaskan sebelumnya.

Selanjutnya adalah penjelasan mengenai system backend. Setelah teks di ekstrak dari file image atau PDF, teks tersebut dapat disimpan melalui backend. Backend akan memastikan permintaan berisi nama file (string) dan konten (string). Jika validasi gagal, maka backend akan secara otomatis mengembalikan respon *error*. Kode untuk proses validasi dapat dilihat pada Kode 6.

Kode 6 Validasi

```
$request->validate([
    'fileName' => 'required|string',
    'content' => 'required|string',
]);
```

Selanjutnya, backend akan membuat jalur berkas di dalam berkas/direktori menggunakan nama berkas yang disediakan. Kemudian konten akan disimpan ke folder yang dituju. Kode untuk mendefinisikan nama dan lokasi penyimpanan file dapat dilihat pada Kode 7.

Kode 7 Mendefinisikan Nama dan Lokasi Penyimpanan File

```
$fileName = $request->fileName;
$filePath = "files/$fileName";
Storage::disk('public')-
>put($filePath, $request->content);
```

Setelah itu, backend akan membuat *record* baru di tabel file (menggunakan model File). *Record* akan menyimpan data berupa nama file, *path* file, jenis MIME yang dikodekan sebagai *'text/plain'* dan ukuran file. Kemudian, *record* disimpan di database. Kode untuk membuat *record* dapat dilihat pada Kode 8.

Kode 8 Membuat *Record*

```
$file = new File();
$file->name = $fileName;
$file->path = $filePath;
$file->type = 'text/plain';
$file->size = strlen($request-
>content);
$file->save();
```

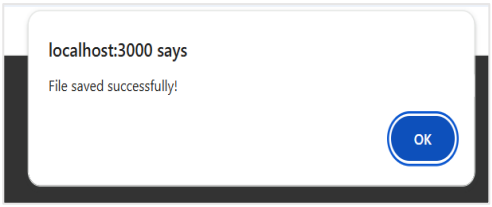
Setelah proses menyimpan file berhasil, *backend* akan mengembalikan respon JSON dengan pesan sukses. Hal ini ditandai dengan kode status HTTP 201. Kode untuk

menampilkan respon dapat dilihat pada Kode 9.

Kode 9 Respon

```
return response()->json([
    'message' => 'File uploaded
successfully',
    'file' => $file,
], 201);
```

Fungsi-fungsi backend yang telah dijelaskan sebelumnya sudah berfungsi sesuai target. Jika proses simpan file berhasil, maka akan muncul pesan *File saved successfully!* seperti yang terlihat pada Gambar 7.



Gambar 7. Simpan File Berhasil

Selain itu, backend juga berfungsi untuk melakukan proses autentikasi pengguna. Backend akan mendeteksi dan memeriksa apakah *password* dan *username* yang diinputkan oleh pengguna dari *frontend* benar atau salah.

Setelah frontend mengirim *username* dan *password* yang diinput oleh pengguna, backend akan memeriksa apakah *username* dan *password* yang diinputkan oleh pengguna terdaftar di database. Jika pengguna terdaftar maka backend akan mengirimkan token ke frontend agar pengguna dapat mengakses program, sedangkan jika pengguna tidak terdaftar di database maka akan muncul pesan "Error: Invalid credentials" seperti yang terlihat pada Gambar 8.

Gambar 8. *Password* atau *Username* Salah

Selain autentikasi, backend juga memberikan proteksi idle. Ketika pengguna diam (*idle*) selama lebih dari 5 menit, maka program secara otomatis akan mengeluarkan pengguna dari program.

Setelah melakukan proses pengujian Text Extractor, diperoleh hasil yang menjelaskan tingkat akurasi program dalam mengekstrak teks dari file pertama berupa dokumen yang diketik menggunakan komputer dan file kedua berupa dokumen yang diketik menggunakan mesin ketik. Hasil pengujian yang diperoleh seperti yang tercatat dalam Tabel 2.

Tabel 2. Hasil Pengujian

File	Jumlah Kesalahan	Total Karakter	CER
File 1	3	1826	0,16%
File 2	138	2154	6,41%

Berdasarkan hasil pengujian pada table di atas, skor CER yang diperoleh hanya 0,33%. Angka tersebut sangatlah kecil dan jauh dibawah skor CER yang artinya program OCR dapat mengekstrak teks dari dokumen yang diketik menggunakan komputer dengan sangat baik.

Terdapat beberapa kesalahan minor yaitu angka romawi "I" yang justru terbaca sebagai angka "1". Ada dua kemungkinan yang dapat menjadi penyebab kesalahan ini. Pertama adalah kurangnya data pelatihan OCR. Jika dalam data pelatihan, representasi angka Romawi "II" dalam font yang mirip dengan dokumen ini kurang, program akan kesulitan membedakannya.

Kemungkinan kedua adalah ambiguitas visual. Pada beberapa font serif (seperti Times New Roman), huruf kapital "I" dan angka "1" terlihat hampir identik. Program mungkin tidak menggunakan konteks yang cukup kuat pada level karakter, program salah membaca dua huruf "I" (II) sebagai "I" (huruf) dan "1" (angka).

Kesalahan berikutnya adalah terdapat satu kali penambahan karakter “B” yang tidak ada pada file referensi. Hal ini mungkin disebabkan karena noktah atau bintik pada kertas tepat di bagian akhir baris sebelum kata "Mengingat". Program OCR menginterpretasi titik atau bintik terisolasi ini sebagai karakter dan berdasarkan konteks sekitarnya, program menebak titik itu adalah huruf "B".

Selain itu, terdapat kesalahan dalam membaca tanda titik menjadi tanda koma. Hal ini dapat disebabkan oleh dua factor, yaitu resolusi rendah atau font khusus. Resolusi rendah atau cetakan yang buram dapat menyebabkan tanda titik "." dapat dengan mudah dibaca sebagai koma (",") oleh program OCR, terutama jika terdapat sebuah noda atau garis asing yang dianggap oleh program sebagai ekor tanda “,”. Alasan lainnya adalah font khusus dengan desain titik tidak bulat sempurna. Beberapa font seperti font Courier New, Rockwell atau Georgia memiliki bentuk titik kotak atau bahkan persegi panjang. Jika titik ini tidak bulat sempurna dan terhubung dengan huruf di depannya karena masalah ketukan (bleed) pada cetakan, titik ini bisa terlihat seperti gumpalan tinta kecil yang menyerupai bentuk koma.

Sedangkan pada file kedua, skor CER yang diperoleh mencapai 6,41%. Hasil yang didapatkan sedikit di atas skor CER yang diharapkan. Angka ini menunjukkan bahwa program OCR sudah mampu mengekstrak teks dari dokumen yang diketik menggunakan mesin ketik dengan cukup baik , tetapi hasilnya belum maksimal.

Pada hasil ekstrak teks, ditemukan banyak kesalahan berupa substitusi atau penggantian karakter. Beberapa di antaranya seperti penulisan kata “Nomor” menjadi “komor” (Huruf “N” disalahtafsirkan sebagai “k”), kata Asisten menjadi Asister, kata “Nama” menjadi “Yama”, huruf “PP” menjadi “P?”, kata “kepada” menjadi “kepads”, dan huruf “m” yang dibaca “rn”.

Salah satu penyebab kesalahan ini adalah kualitas cetakan atau scan. Dokumen fisik yang sudah lama mengalami penurunan kualitas cetak, tintanya semakin memudar atau terdapat noda pada dokumen. Hal tersebut mengakibatkan garis pada beberapa karakter menjadi terputus dan menyebabkan program salah melakukan penafsiran terhadap karakter yang dibaca. Faktor lain yang menjadi penyebab kesalahan adalah hasil scan yang buram, beresolusi rendah atau memiliki *noise* (titik-titik kecil) sangat menyulitkan algoritma OCR untuk membaca karakter pada dokumen. Selain itu, penyebab kesalahan adalah keberadaan stempel atau tanda tangan. Tinta stempel atau tanda tangan yang menimpa teks dapat mengganggu latar belakang dan menyulitkan OCR untuk mengisolasi karakter teks dengan benar.

Faktor lain yang mungkin menjadi penyebab kesalahan yang terjadi adalah kurangnya variasi data untuk melatih model OCR. Model OCR yang tidak terlatih untuk membaca jenis *font* yang digunakan pada mesin ketik akan kesulitan mengenali dan menafsirkan karakter dari teks.

4. KESIMPULAN

Penelitian ini menghasilkan sistem untuk mengekstrak teks dari gambar dan file PDF, yang terintegrasi dengan fungsi pengarsipan data digital. Sistem ini bertujuan untuk mendigitalisasi data dan

mengelompokkannya berdasarkan jenisnya sehingga data mudah dicari. Proses digitalisasi data menggunakan teknologi OCR, khususnya dengan mengimplementasikan *library* Tesseract untuk mengekstrak teks. Untuk file PDF, Tesseract OCR dibantu oleh *library* PDF.js, yang mengubah format file PDF menjadi gambar sebelum proses ekstraksi teks. Sistem ini dirancang sebagai sistem tertutup dengan koneksi lokal (LAN) dan menggunakan *localhost* untuk menjaga keamanan data sensitif. Fitur keamanan tambahan mencakup enkripsi *password*, penggunaan *authentication token* untuk *login* dan *idle time* yang secara otomatis akan mengeluarkan pengguna setelah 5 menit tidak aktif.

Berdasarkan pengujian terhadap kinerja program ekstrak teks terhadap dokumen yang diketik menggunakan komputer, program menunjukkan akurasi yang sangat tinggi dengan nilai Character Error Rate (CER) hanya 0,16%. Kesalahan yang terjadi bersifat minor dan sporadis, disebabkan terutama oleh ambiguitas visual pada font tertentu serta adanya noise atau noktah kecil pada dokumen yang disalahartikan sebagai karakter.

Namun kinerja program ekstrak teks terhadap dokumen yang diketik menggunakan mesin ketik masih belum maksimal. Program masih dapat mengekstrak teks dari dokumen mesin ketik dengan nilai CER 6,41%, yang menunjukkan kemampuan dasar yang memadai. Namun, tingginya jumlah error substitusi mengindikasikan bahwa kualitas fisik dokumen dan karakteristik font mesin ketik yang khas merupakan tantangan utama. Faktor lain seperti keberadaan stempel yang menimpa teks dan kemungkinan kurangnya variasi data pelatihan model OCR untuk font mesin ketik juga berkontribusi terhadap kesalahan.

Ada beberapa saran untuk penelitian berikutnya. Pertama, melatih model OCR menggunakan dataset yang berisi gambar teks hasil mesin ketik, dokumen yang sudah tua dan dokumen yang memiliki stemple atau watermark. Dataset ini akan membuat model menjadi lebih *robust* dalam mengenali karakteristik *font* dan *noise* yang khas pada dokumen analog. Kedua, menerapkan implementasi pra-pemrosesan yang lebih intensif untuk meningkatkan kualitasnya. Teknik yang dapat digunakan seperti *noise reduction* untuk mengurangi bintik-bintik dan noise, *contrast enhancement* untuk meningkatkan kontras antara teks dan latar belakang, *deskewing* untuk memastikan teks lurus sebelum proses ekstraksi dan *morphological operations* untuk menutupi lubang kecil pada karakter atau memisahkan karakter yang menyatu. Ketiga, mengembangkan kemampuan sistem tidak hanya untuk mengekstrak teks , tetapi juga memahami struktur dokumen (misalnya membedakan header, footer, paragraf, tabel dan tanda tangan). Hal ini akan membuat sistem pengarsipan lebih cerdas karena dapat mengelompokkan dan mengindeks data berdasarkan bagian-bagiannya. Keempat, menginvestigasi cara untuk memparalelkan proses ekstraksi OCR untuk lebih banyak dokumen sekaligus sehingga sistem dapat menangani volume digitalisasi yang lebih besar tanpa mengorbankan kecepatan respons.

UCAPAN TERIMA KASIH

Ucapan terima kasih disampaikan kepada :

Bapak Ariya Dwika Cahyono, S. Kom, M.T., selaku dosen pembimbing.
Bapak Yos Richard Beeh, ST., M.Cs. selaku dosen reviewer.
Bapak Budhi Kristianto, S.Kom., M.Sc., Ph.D. selaku Kepala Program Studi S1 Teknik Informatika.

DAFTAR PUSTAKA

[1] T. Nasir, M. K. Malik, and K. Shahzad, "MMU-OCR-21: Towards End-to-End Urdu Text Recognition Using Deep Learning," *IEEE Access*, vol. 9, pp. 124945–124962, 2021, doi: 10.1109/ACCESS.2021.3110787.

[2] A. L. Firdaus, M. S. Kurnia, T. Shafera, and W. I. Firdaus, "Implementasi Optical Character Recognition (OCR) Pada Masa Pandemi Covid-19," *JUPITER J. Penelit. Ilmu Dan Teknol. Komput.*, vol. 13, no. 2, pp. 188–194, Oct. 2021.

[3] M. Li *et al.*, "TrOCR: Transformer-Based Optical Character Recognition with Pre-trained Models," *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 11, pp. 13094–13102, Jun. 2023, doi: 10.1609/aaai.v37i11.26538.

[4] K. Nisha, T. Wahyuni, and M. A. M. Hayat, "Pemeriksaan KTP Menggunakan Optical Character Recognition (OCR) dan Pengenalan Background serta Komponen KTP," *Arus J. Sains dan Teknol.*, vol. 2, no. 2, pp. 490–495, Oct. 2024.

[5] R. M. I. Indrakusuma, A. S. Ahmadiyah, and N. F. Ariyani, "Pengenalan dan Klasifikasi Tulisan pada Nota Pembelian Material (Studi Kasus Proyek Konstruksi)," *J. Tek. ITS*, vol. 10, no. 2, pp. 478–483, 2021.

[6] B. Rahman, "Analisis Manfaat Data Digital Spasial Bagi Desa," *Pondasi*, vol. 27, no. 1, p. 88, Jul. 2022, doi: 10.30659/pondasi.v27i1.22891.

[7] Ghifari Aminudin Fad'li, Marsofiyati Marsofiyati, and Suherdi Suherdi, "Implementasi Arsip Digital Untuk Penyimpanan Dokumen Digital," *J. Manuhara Pus. Penelit. Ilmu Manaj. dan Bisnis*, vol. 1, no. 4, pp. 01–10, Aug. 2023, doi: 10.61132/manuhara.v1i4.115.

[8] Z. Patmawati and Ismaya, "Strategi Digitalisasi dan Pengelolaan Arsip Elektronik Era Revolusi Industri 4.0 di Dinas Perpustakaan dan Kearsipan Kabupaten Bantul," in *Seminar Nasional Hukum Ilmu Sosial dan Ilmu Politik*, Banten: Universitas Terbuka, 2024.

[9] K. D. K. A. Wardani, N. P. I. P. Dewi, and A. A. N. E. S. Gorda, "Optimalisasi Kinerja Karyawan Melalui Pengelolaan Arsip Digital Di Kadin Bali," *J. Soc. Sci. Technol. Community Serv.*, vol. 4, no. 2, pp. 239–248, 2023.

[10] H. Farahdiba, C. W. Wolor, and Marsofiyati, "Analisis pengelolaan arsip digital pada PT Anugrah Alam Karunia Abadi," *J. Adm. Soc. Sci.*, vol. 5, no. 1, pp. 41–53, Dec. 2023.

[11] M. Rahman bin Abdullah, "A Review of Intelligent Document Processing Applications Across Diverse Industries," 2022.

[12] K. Kusnantoro, T. Rohana, and D. S. Kusumaningrum, "Implementasi Metode Tesseract OCR(Optical Character Recognition) untuk Deteksi Plat Nomor Kendaraan Pada Sistem Parkir," *Sci. Student J. Inf.*, vol. 3, no. 1, pp. 59–67, 2022.

[13] S. M. Angela, A. Eviyanti, and M. I. Mauliana, "PENGEMBANGAN TEKNOLOGI OPTICAL CHARACTER RECOGNITION DI FLUTTER BERUPA DETEKSI TEKS PADA GAMBAR," *J. Tek.*

Inf. dan Komput., vol. 7, no. 1, p. 17, Jun. 2024, doi: 10.37600/tekinkom.v7i1.1167.

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification," in *Proceedings of the 23rd international conference on Machine learning - ICML '06*, New York, New York, USA: ACM Press, 2006, pp. 369–376. doi: 10.1145/1143844.1143891.

[15] M. A. Rifqi, M. Awaluddin, and L. M. Sabri, "Perancangan WebGIS Persebaran Rumah Sakit Kota Semarang," *J. Geod. Undip*, vol. 12, no. 3, pp. 321–329, Nov. 2023.

[16] M. Arman, "Analisa Jaringan Local Area Network (Lan) Dengan Aplikasi Cisco Packet Tracer Pada PT. Bank Negara Indonesia (Persero) Tbk Kcp Watansoppeng," *J. Ilm. Sist. Inf. dan Tek. Inform.*, vol. 5, no. 2, pp. 41–50, Oct. 2022, doi: 10.57093/jisti.v5i2.126.

[17] N. A. Karima, A. N. Aisyah, H. V. Silla, L. B. Handoko, and R. R. Sani, "Kriptografi Teks Berbasis Algoritma Substitusi Vigenere Cipher 8 Bit," *J. Masy. Inform.*, vol. 15, no. 1, pp. 1–13, May 2024, doi: 10.14710/jmasif.15.1.60836.

[18] M. N. Darpito, Kartika Firdausy, and Abdul Fadlil, "Perbandingan Unjuk Kerja Library Optical Character Recognition (OCR) dalam Pengenalan Teks pada Dokumen Digital," *J. Inform. Polinema*, vol. 11, no. 3, pp. 273–282, May 2025, doi: 10.33795/jip.v11i3.7025.

[19] H. M. Al-Barhamtoshy, K. M. Jambi, S. M. Abdou, and M. A. Rashwan, "Arabic Documents Information Retrieval for Printed, Handwritten, and Calligraphy Image," *IEEE Access*, vol. 9, pp. 51242–51257, 2021, doi: 10.1109/ACCESS.2021.3066477.