

# **FINETUNING MODEL LAYOUTLM UNTUK PEMBACAAN NOTA BERBAHASA INDONESIA**

## **PROPOSAL TUGAS AKHIR**

Diajukan guna memenuhi sebagian persyaratan dalam rangka menyelesaikan  
Pendidikan Sarjana Strata Satu (S1) Program Studi Teknologi Informasi



**I Made Andre Dwi Winama Putra**  
**NIM: 1905551003**

**PROGRAM STUDI TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK  
UNIVERSITAS UDAYANA  
2022**

# **FINETUNING MODEL LAYOUTLM UNTUK PEMBACAAN NOTA BERBAHASA INDONESIA**

## **PROPOSAL TUGAS AKHIR**

Diajukan guna memenuhi sebagian persyaratan dalam rangka menyelesaikan  
Pendidikan Sarjana Strata Satu (S1) Program Studi Teknologi Informasi



**I Made Andre Dwi Winama Putra**  
**NIM: 1905551003**

**PROGRAM STUDI TEKNOLOGI INFORMASI**  
**FAKULTAS TEKNIK**  
**UNIVERSITAS UDAYANA**  
**2022**

## KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadapan Ida Sang Hyang Widhi Wasa/Tuhan Yang Maha Esa, karena atas Asung Kerta Wara Nugraha-Nya, penulis dapat menyelesaikan tugas akhir dengan judul “*Finetuning Model LayoutLM untuk Pembacaan Nota berbahasa Indonesia*”. Selama pelaksanaan tugas akhir ini penulis mendapat banyak masukan dan bimbingan dari berbagai pihak. Untuk itu, penulis ingin mengucapkan rasa terima kasih kepada:

1. Bapak Ir. I Ketut Sudarsana, ST., Ph.D, selaku Dekan Fakultas Teknik universitas Udayana.
2. Bapak Dr. Eng. I Putu Agung Bayupati, ST.,MT., selaku Ketua Program Studi Teknologi Informasi Universitas Udayana.
3. Ibu Ni Kadek Ayu Wirdiani, ST., MT , selaku dosen pembimbing I dan Bapak Dr. A.A. Kompang Oka Sudana, S.Kom., MT, selaku dosen pembimbing II yang telah banyak memberikan masukan dan bimbingan selama penyusunan tugas akhir ini.
4. Bapak Gusti Made Arya Sasmita, ST., MT , selaku dosen pembimbing akademik, yang telah memberikan bimbingan selama menempuh pendidikan di Program Studi Teknologi Informasi Fakultas Teknik Universitas Udayana.
5. Bapak I Nyoman Piarsa, ST., MT., Bapak Anak Agung Ngurah Hary Susila, S.TI., M.MT., dan Bapak I Nyoman Prayana Trisna, S.Kom., M.Cs. selaku penguji ujian proposal Tugas Akhir yang senantiasa membantu dan mengoreksi penulisan Tugas Akhir
6. Kedua orang tua dan keluarga yang telah memberikan dukungan dan motivasi dalam pembuatan tugas akhir ini.
7. Teman-teman seperjuangan dan segenap civitas di Program Studi Teknologi Informasi Universitas Udayana yang telah memberikan sumbangan ide, pemikiran dan dukungan dalam penyusunan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih jauh dari sempurna. Akhir kata penulis memohon maaf jika ada kesalahan dalam penulisan tugas akhir ini.

Denpasar, Januari 2023

I Made Andre Dwi Winama Putra

## ABSTRAK

Struk atau nota belanja adalah sebuah catatan yang memuat kesepakatan transaksi antara penjual dan pembeli. Penyimpanan informasi struk atau nota belanja ini penting untuk dilakukan agar pengeluaran kas dapat terlihat dengan jelas. Catatan struk atau nota belanja ini biasanya berupa kertas hasil *print* yang membuatnya mudah hilang. Pemindahan informasi dari struk atau nota ke dalam bentuk digital akan memakan banyak waktu jika proses pemindahannya dilakukan secara manual. Pembuatan sistem yang menerapkan *Optical Character Recognition* (OCR) untuk membaca struk atau nota dapat membantu proses pemindahan informasi yang terkandung menjadi bentuk digital.

**Kata Kunci:** *Computer Vision, Optical Character Recognition, Android, Deep Learning, LayoutLM, Nota, Struk*

## **ABSTRACT**

Receipt is a record that contains the transaction agreement between the seller and the buyer. Keeping track of these receipt is important to clearly see cash expenditures. These receipt record usually came in the form of a printed paper which makes it easy to forget and in turn lose them. The process of transferring contents of the receipt into digital form will take a long time if the transfer process is done manually. The creation of a system that applies Optical Character Recognition (OCR) to read receipts automatically can help speeds up process of transferring the contained information into digital form.

**Kata Kunci:** *Computer Vision, Optical Character Recognition, Android, Deep Learning, LayoutLM, Nota, Struk*

## DAFTAR ISI

<b>KATA PENGANTAR.....</b>	<b>iii</b>
<b>ABSTRAK .....</b>	<b>v</b>
<b>ABSTRACT .....</b>	<b>vi</b>
<b>DAFTAR ISI.....</b>	<b>vii</b>
<b>DAFTAR GAMBAR.....</b>	<b>ix</b>
<b>DAFTAR KODE PROGRAM .....</b>	<b>x</b>
<b>DAFTAR TABEL .....</b>	<b>xi</b>
<b>BAB I PENDAHULUAN.....</b>	<b>1</b>
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	3
1.3 Tujuan Penelitian.....	3
1.4 Manfaat Penulisan .....	3
1.5 Batasan Masalah .....	4
1.6 Sistematika Penulisan .....	4
1.6.1 Bab I Pendahuluan .....	4
1.6.2 Bab II Tinjauan Pustaka .....	4
1.6.3 Bab III Metodologi Penelitian.....	5
1.6.4 Bab IV Pembahasan dan Analisis Hasil.....	5
1.6.5 Bab V Penutup .....	5
<b>BAB II TINJAUAN PUSTAKA.....</b>	<b>6</b>
2.1 State of the Art .....	6
2.2 OCR.....	10
2.3 Google Vision.....	10
2.4 LayoutLM.....	11
2.5 Metriks Evaluasi Model .....	12
<b>BAB III METODOLOGI PENELITIAN .....</b>	<b>14</b>
3.1 Tempat dan Waktu Penelitian .....	14
3.2 Data Penelitian .....	14
3.2.1 Data Primer .....	14
3.2.2 Data Sekunder .....	15
3.3 Instrumen Pembuatan Sistem .....	15
3.4 Alur Penelitian.....	15
3.4.1 Pembuatan Dataset .....	16
3.4.2 Finetuning Model LayoutLM.....	18
3.4.3 Evaluasi Model.....	20
3.5 Gambaran Umum Sistem .....	20
3.6 Alur Aplikasi .....	21
3.7 Rancangan Sistem .....	25
<b>BAB IV HASIL DAN PEMBAHASAN .....</b>	<b>30</b>

4.1	Pembuatan Dataset .....	30
4.1.1	Segmentasi Gambar .....	30
4.1.2	Pembacaan Karakter.....	38
4.1.3	Annotasi Dataset .....	39
4.2	Finetuning Model LayoutLM.....	42
4.2.1	Data Preparation.....	42
4.2.2	Data Pipelining.....	44
4.2.3	Model Finetuning .....	48
4.2.4	Model Evaluation .....	50
4.3	Deployment Model LayoutLM .....	50
4.3.1	Flask API Server .....	50
4.3.2	Android APK .....	52
<b>BAB V PENUTUP .....</b>		<b>53</b>
5.1	Kesimpulan.....	53
5.2	Saran .....	53
<b>DAFTAR PUSTAKA .....</b>		<b>54</b>



## DAFTAR GAMBAR

Gambar 2.1 Gambar Pelatihan LayoutLM (Xu et al., 2020) .....	11
Gambar 3.1 Flowchart Alur Penelitian .....	16
Gambar 3.2 Flowchart Pembuatan Dataset .....	17
Gambar 3.3 Flowchart Segmentasi ROI Nota.....	18
Gambar 3.4 Flowchart Pelatihan LayoutLM .....	19
Gambar 3.5 Gambaran Umum Sistem .....	21
Gambar 3.6 Flowchart Pembacaan Nota pada Aplikasi Android .....	22
Gambar 3.7 Flowchart Proses Pembacaan Nota pada Web Server.....	23
Gambar 3.8 Flowchart Preprocessing Gambar pada Server .....	24
Gambar 3.9 Flowchart Melihat Histori Pembacaan Nota pada Aplikasi Android....	25
Gambar 3.10 Rancangan Tampilan Depan Sistem .....	26
Gambar 3.11 Rancangan Segmentasi Nota .....	27
Gambar 3.12 Rancangan Hasil Preview Pembacaan .....	28
Gambar 3.13 Rancangan Riwayat Hasil Scan.....	29
Gambar 4.1 Hasil Proses Pembacaan Gambar Serta Resize .....	31
Gambar 4.2 Hasil Proses Bluring dan Dilasi Pada Gambar.....	32
Gambar 4.3 Hasil Deteksi Tepi Dengan Menggunakan Canny .....	33
Gambar 4.4 Deteksi Kontur Hasil tepi Canny .....	35
Gambar 4.5 Hasil Pemilihan Kontur .....	36
Gambar 4.6 Nota Hasil Segmentasi .....	37
Gambar 4.7 Hasil Deteksi Google Vision.....	38
Gambar 4.8 Gambar Penggunaan Fungsi Annotasi tahap 1 .....	41
Gambar 4.9 Gambar Penggunaan Fungsi Annotasi tahap 2 .....	42
Gambar 4.10 Hasil Segmentasi Bounding Box.....	44
Gambar 4.11 Hasil Pengolahan Dataset.....	46
Gambar 4.12 Hasil Pengubahan Data Menjadi Encoding.....	47
Gambar 4.13 Hasil Web Server Deployment.....	51

## **DAFTAR KODE PROGRAM**

Kode Program 4.1 Pembacaan File Gambar Nota .....	30
Kode Program 4.2 Preprocessing Nota Tahap 1 .....	31
Kode Program 4.3 Proses Deteksi Tepi Canny .....	33
Kode Program 4.4 Deteksi Kontur Hasil tepi Canny .....	34
Kode Program 4.5 Pemilihan Kontur.....	35
Kode Program 4.6 Segmentasi Nota .....	37
Kode Program 4.7 Pembacaan Karakter Dengan Google Vision .....	38
Kode Program 4.8 Kode Program Otomasi Pembacaan OCR pada Setiap Nota.....	39
Kode Program 4.9 Fungsi Pembantu Annotasi Dataset .....	41
Kode Program 4.10 Persiapan Dataset.....	43
Kode Program 4.11 Fungsi Normalisasi Dataset .....	43
Kode Program 4.12 Segmentasi Kata Bounding Box .....	44
Kode Program 4.13 Inisiasi AutoProcessor LayoutLM.....	44
Kode Program 4.14 Pengolahan Dataset.....	45
Kode Program 4.15 Pengubahan Data Menjadi Encoding.....	47
Kode Program 4.16 Metrics Pelatihan Model.....	49
Kode Program 4.17 Memuat Model LayoutLM .....	49
Kode Program 4.18 Pelatihan Model .....	49
Kode Program 4.19 Evaluasi Model .....	50
Kode Program 4.20 Web Server Deployment.....	51

## DAFTAR TABEL

Tabel 2.1 Confusion Matrix .....	12
Tabel 3.1 Spesifikasi Perangkat Keras.....	15
Tabel 4.1 Pelabelan Dataset .....	40

# **BAB I**

## **PENDAHULUAN**

Bab I pendahuluan pada laporan penelitian tugas akhir ini berisi tentang latar belakang, rumusan masalah, batasan masalah, manfaat, serta sistematika yang akan digunakan dalam pembuatan penelitian tugas akhir ini.

### **1.1 Latar Belakang**

Transaksi adalah sebuah kesepakatan antara pembeli dan penjual untuk menukar barang dan jasa yang mereka miliki. Belanja adalah salah satu jenis transaksi yang menghasilkan sebuah catatan berbentuk struk atau nota pembayaran. Belanja dilakukan dengan cara membayarkan sejumlah uang kepada penjual, kemudian pihak penjual akan memberikan barang/jasa beserta struk atau nota belanja yang memuat isi dari transaksi yang dilakukan. Catatan dalam bentuk struk atau nota ini penting untuk disimpan agar pengeluaran dana dapat terlihat dengan jelas.

Penyimpanan informasi yang terdapat pada catatan belanja struk atau nota sebaiknya dilakukan dalam bentuk digital agar tidak mudah hilang. Pemindahan informasi transaksi ke dalam bentuk digital tentunya akan memakan banyak waktu jika setiap data transaksi tersebut harus di-*input* secara manual ke dalam komputer (Kumar, Kaware, & Singh, 2020). Keberadaan sistem yang dapat mengekstrak informasi pada struk atau nota dan menyimpannya dalam format digital secara otomatis akan meningkatkan efisiensi kerja. Pemindahan media penyimpanan ke dalam bentuk digital juga akan mengurangi risiko struk dan nota tersebut hilang. Penyimpanan informasi dalam bentuk digital membuat informasi tentang pengeluaran dalam rentang waktu tertentu lebih mudah terlihat. Salah satu metode yang dapat digunakan untuk mengekstrak informasi *text* adalah metode *Optical Character Recognition*.

*Optical Character Recognition* (OCR) adalah proses konversi gambar huruf menjadi karakter ASCII yang dikenali oleh komputer. Teknologi OCR dapat mengubah gambar yang berasal dari dokumen yang di-*scan*, tulisan digital, maupun tulisan tangan (Mohammad et al., 2014). Proses ekstraksi informasi menggunakan *Optical Character Recognition* (OCR) saat ini sudah memiliki tingkat akurasi yang tinggi yaitu hingga 95%, baik dalam pengenalan tulisan digital (*digital character*) maupun dalam pengenalan tulisan tangan (*handwritten character*). Penerapan teknologi OCR dapat digunakan untuk membuat proses ekstraksi informasi yang terdapat pada struk dan nota dapat dilakukan secara otomatis (Kumar, Kaware, Singh, et al., 2020).

Teknologi OCR dapat digunakan untuk mendeteksi kalimat pada struk dan nota. Penerapan OCR dalam mendeteksi struk dan nota akan mengurangi waktu yang dibutuhkan untuk pemindahan informasi belanja menjadi bentuk digital. Manfaat lainnya dari keberadaan aplikasi OCR struk dan nota ini adalah total pengeluaran dalam jangka waktu tertentu dapat terlihat dengan mudah tanpa perlu menghitungnya secara manual.

Sistem pembacaan bukti transaksi berbentuk nota sudah pernah dibuat dengan menggunakan metode *Template Matching* serta OCR oleh Lin C. Penggunaan metode *Template Matching* memiliki kekurangan yaitu pembacaan informasi pada nota tidak dapat diimplementasikan pada bentuk nota yang beragam. LayoutLM adalah sebuah Model *Deep Learning* baru yang dapat berlatih pada *text* serta *bounding box* sebagai letak dari *text* tersebut. Penggunaan Model LayoutLM untuk mendeteksi nota diharapkan membuat pembacaan informasi dapat dilakukan pada bentuk nota yang beragam (Lin et al., 2022).

Sistem pembacaan bukti transaksi berbentuk nota akan bekerja dengan cara mengakuisisi citra bukti transaksi yang diambil menggunakan kamera dari *smartphone* android terlebih dahulu. Citra tersebut selanjutnya akan dikirim menuju *web server* dengan menggunakan *API call*. *Web server* akan menerima dan memproses citra bukti tersebut hingga menjadi *text* dan mengirimkannya kembali

kepada *smartphone* Android. Informasi tentang bukti pembayaran yang terdeteksi selanjutnya akan ditampilkan dalam layar *smartphone* dan pengguna dapat memilih untuk menyimpan bukti transaksi tersebut.

## **1.2 Rumusan Masalah**

Berdasarkan latar belakang yang telah dijelaskan, maka masalah-masalah yang dapat dirumuskan adalah sebagai berikut.

1. Bagaimana cara merancang dan membangun aplikasi yang dapat membaca bukti transaksi berbentuk kertas menjadi bentuk digital secara otomatis.
2. Bagaimanakah akurasi dari sistem pembacaan nota belanja yang dibuat.

## **1.3 Tujuan Penelitian**

Berdasarkan rumusan masalah yang didapatkan, tujuan yang ingin dicapai dari penelitian tugas akhir ini adalah sebagai berikut.

1. Menghasilkan sebuah aplikasi yang dapat membaca dan menyimpan bukti transaksi dalam bentuk digital secara otomatis.
2. Menghasilkan sistem yang dapat mengekstrak informasi dalam bukti transaksi dengan akurasi yang baik.

## **1.4 Manfaat Penulisan**

Manfaat penelitian ini dibuat berdasarkan masalah yang ingin diselesaikan, serta tujuan ingin dicapai oleh penulis. Penelitian ini diharapkan akan memberikan manfaat kepada Pengguna dari aplikasi, yaitu Pengguna dapat merasakan manfaat dari kemudahan pencatatan pembayaran hanya dengan menggunakan gambar serta dapat memahami dan melakukan manajemen keuangan dengan lebih baik.

## **1.5 Batasan Masalah**

Batasan masalah yang digunakan dalam penelitian ini dibuat untuk memastikan agar ruang lingkup penelitian tidak terlampau jauh dan melebar. Batasan masalah dari penyusunan laporan tugas akhir ini adalah sebagai berikut.

1. *Dataset* yang digunakan berasal dari berbagai nota belanja dari minimarket dan restoran yang berupa nota *print out*.
2. Nota yang digunakan berbentuk persegi panjang dan tidak terlipat.
3. Sistem hanya akan membaca satu buah nota pada setiap gambar yang dikirimkan ke *web server*.

## **1.6 Sistematika Penulisan**

Sistematika penulisan yang digunakan pada laporan tugas akhir ini terdiri dari pendahuluan, tinjauan pustaka, pembahasan metodologi penelitian, hasil pembuatan sistem, serta simpulan dan saran yang dirangkum secara urut dan sistematis.

### **1.6.1 Bab I Pendahuluan**

Bab I dimulai dari penjelasan mengenai latar belakang diambilnya topik penelitian, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, hingga sistematika penulisan yang digunakan untuk menyusun penelitian ini.

### **1.6.2 Bab II Tinjauan Pustaka**

Bab II berisikan tentang *State of The Art* (SOTA) dari permasalahan yang diangkat serta dasar-dasar teori yang bersumber dari artikel-artikel akademik dan platform pembelajaran akademik yang digunakan sebagai dasar dalam pembahasan permasalahan dan solusi penelitian yang dilakukan.

### **1.6.3 Bab III Metodologi Penelitian**

Bab III metodologi penelitian memuat informasi mengenai tempat dan waktu pelaksanaan penelitian, sumber data yang digunakan dalam penelitian, serta alur penelitian. Pada bab III ini juga terdapat informasi mengenai instrumen pembuatan sistem serta rancangan dari sistem yang akan dibuat.

### **1.6.4 Bab IV Pembahasan dan Analisis Hasil**

Bab IV berisikan informasi tentang pembahasan hasil dari penelitian mulai dari hasil pengumpulan data yang dilakukan, proses pemodelan, hingga evaluasi Model. Pada bab ini juga terdapat tampilan dari sistem yang dibuat serta pembahasan mengenai hasil penelitian secara keseluruhan.

### **1.6.5 Bab V Penutup**

Bab V berisikan mengenai simpulan dan saran yang dibuat selama dilaksanakannya penelitian ini. Simpulan mengacu pada hasil yang didapatkan selama pengerjaan pembuatan sistem. Saran berisikan tentang masukan yang dapat diberikan kepada pembaca untuk mendapatkan hasil yang lebih baik, maupun untuk mempermudah replikasi dari penelitian.



## BAB II

### TINJAUAN PUSTAKA

Bab II ini berisikan tentang dasar-dasar teori yang bersumber dari artikel-artikel akademik, serta platform pembelajaran akademik yang digunakan sebagai dasar dalam mengerjakan *Finetuning* dari Model LayoutLM ini.

#### 2.1 State of the Art

*State of The Art* adalah kumpulan artikel, jurnal dan literatur yang membahas topik serupa dengan penelitian ini. Pembuatan *State of the art* dilakukan dengan tujuan untuk membandingkan teknologi yang saat ini digunakan untuk menyelesaikan masalah yang serupa, yaitu tentang pengenalan *Natural Language Processing* (NLP) dengan menggunakan *Optical Character Recognition* (OCR).

Howard menyampaikan dalam artikelnya bahwa sampai saat ini penyelesaian masalah NLP masih bersifat spesifik pada satu permasalahan. Penyelesaian yang bersifat spesifik pada satu permasalahan membuat pembuatan arsitektur baru serta pelatihan ulang harus dilakukan agar Model dapat menyelesaikan permasalahan baru. Penyelesaian yang bersifat spesifik ini membuat Howard membuat Model bernama *Universal Language Model Fine-tuning* (ULMFiT). ULMFiT adalah sebuah Model yang dirancang dan dikembangkan dengan tujuan agar proses *Transfer Learning* dapat dilakukan dengan lebih efektif dan dapat digunakan untuk menyelesaikan berbagai permasalahan NLP. Hasilnya Howard berhasil membuat Model ULMFiT dengan hasil akurasi yang setara dengan berbagai Model SOTA pada kategori permasalahan NLP yang berbeda (Howard & Ruder, 2018).

Devlin melanjutkan perkembangan penggunaan *transfer learning* pada permasalahan NLP dan membuat sebuah Model bernama *Bidirectional Encoder Representations from Transformers* (BERT). BERT adalah sebuah *pretrained* Model yang menggunakan representasi *Deep Bidirectional* dari teks dengan cara

menggabungkan konteks kiri dan kanan dari teks tersebut pada setiap *layer*-nya. Hasil pembuatan Model BERT ini berhasil mendapatkan akurasi di atas 82.1% yang berhasil melampaui Model SOTA dalam berbagai *dataset*. Model BERT yang dihasilkan juga dapat dengan mudah di-*tuning* untuk menyelesaikan berbagai jenis permasalahan NLP seperti *language inference*, dan *question answering* tanpa mengubah banyak dari arsitektur Model (Devlin et al., 2018).

Penelitian tentang *Finetuning* dari Model BERT, pernah digunakan untuk memahami literatur biomedis yang terus berkembang dengan sangat cepat. Setiap harinya terdapat sekitar 3000 artikel baru tentang literatur biomedis yang membuat ekstraksi informasi biomedis memerlukan *tools* yang akurat. Hasil penerapan *Finetuning* pada Model BERT berhasil membuat model bernama BioBERT dan mendapatkan akurasi mencapai 72 % dalam berbagai *dataset* serta berhasil melampaui Model *state of the art* dalam beberapa kategori ekstraksi informasi biomedis (Lee et al., 2020).

Metode pembacaan OCR pada nota belanja pernah dilakukan dengan mengaplikasikan *Template Matching* pada nota yang ingin dibaca. Hasil dari penggunaan metode *Template Matching* tersebut selanjutnya akan dibaca dengan menggunakan Model *Deep Learning YOLOv4-s*. Sistem yang dikembangkan ini berhasil mendapatkan akurasi sebesar 80.93% dengan percobaan menggunakan CNN dan sebesar 99.39% dengan menggunakan *YOLOv4-s* yang dikembangkan dalam pembacaan karakter (Lin et al., 2022).

Pembacaan nota pernah dilakukan dengan memanfaatkan *Library BlinkReceipt*. *BlinkReceipt* merupakan sebuah *Application Programming Interface* (API) yang dapat digunakan dalam platform *IOS*, *Android*, serta melalui *javascript*. *Library BlinkReceipt* adalah sebuah *library* yang dapat digunakan secara khusus untuk membaca nota belanja. Hasil yang didapatkan dalam penggunaan *Library BlinkReceipt* memiliki beberapa kekurangan seperti aplikasi tidak dapat membaca potongan harga, serta pembacaan harus dilakukan berulang-ulang jika jumlah daftar belanja dalam struk cukup banyak (Andreas et al., 2020).

Pembacaan merek toko dengan menerapkan lokalisasi pada gambar nota belanja pernah dilakukan pada tahun 2018. Lokalisasi gambar nota belanja ini dilakukan dengan cara mendeteksi tepi yang digunakan untuk mengklasifikasi merek toko. Pembacaan OCR dari merek toko dilakukan dengan *Deep Convolutional Neural Networks* (CNN). Hasil dari percobaan ini berhasil melokalisasi merek toko sebesar 86% (Raoui-Outach et al., 2018).

Pembacaan pengaduan ketenagakerjaan dengan menggunakan *Tesseract OCR* dan *NTLK toolkit* pernah dilakukan untuk mengekstrak informasi pada surat. Penelitian ini menggunakan *NTLK toolkit* untuk memahami isi surat, sedangkan proses ekstraksi tulisan surat dilakukan dengan menggunakan *Tesseract OCR*. Penelitian ini berhasil mengekstrak informasi sebesar 66.7% pada surat tulisan tangan dan sebesar 91,67% pada surat yang diketik (Puspitarani & Syukriyah, 2017).

Penggunaan metode *Weighted Bounding Box Regression Loss* untuk melakukan segmentasi pada dokumen pernah dilakukan untuk meningkatkan akurasi. Penggunaan metode ini membuat hasil segmentasi dokumen yang memiliki banyak objek-objek kecil dapat dilakukan dengan lebih akurat. Hasil yang didapatkan dalam pengujian segmentasi pada *Dense Article Dataset* (DAD) dan *dataset PubLayNet* memiliki *f1 score* sebesar 96.26% dan 97.11% dengan menggunakan Model *DeeplabV3+* (Markewich et al., 2022).

Model *TableSegNet* merupakan Model segmentasi yang dapat mendeteksi tabel dalam sebuah gambar dokumen. *TableSegNet* menggunakan arsitektur *fully convolutional network* untuk mendeteksi dan membedakan tabel secara bersamaan. Hasilnya penelitian Model segmentasi *TableSegNet* ini dapat menghasilkan akurasi sebesar 90% dalam dataset ICDAR2019 (Nguyen, 2022).

Ekstraksi kode MRZ pada dokumen *visa* dan *passport* pernah dilakukan dengan menggunakan *Convolutional Neural Networks* (CNN). Model CNN dibuat untuk mendeteksi *MRZ code* dari gambar *passport* digital. Hasilnya Model ini dapat mendeteksi 100% dari kode MRZ dan 99.25% *macro-f1* dari pengenalan karakter pada *dataset passport* dan Visa (Liu et al., 2022).

Penelitian tentang ekstraksi informasi pada dokumen yang terstruktur pernah dilakukan dengan menggunakan arsitektur *Siamese Networks*. Penggunaan arsitektur *Siamese Networks* dikombinasikan dengan metode *Similarity*, *One-Shot Learning*, dan *Context/Memory Awareness* untuk melakukan proses ekstraksi informasi pada dokumen. Hasil dari arsitektur pada penelitian ini berhasil meningkatkan skor f1 sebesar 8.25% bila dibandingkan dengan arsitektur *query answer* (Holeček, 2020).

Penelitian tentang pembacaan karakter huruf Bali pada *wrésastra script* dan karakter huruf Bali pernah dilakukan dengan menggunakan *Zoning Feature Extraction* serta *K-Nearest Neighbor* (KNN). Penggunaan *Zoning Feature Extraction* dilakukan untuk mengekstrak fitur dari setiap karakter dan menjadikannya *embedding* untuk diklasifikasikan dengan Model KNN. Hasil yang pada penelitian ini adalah Model terbaik didapatkan saat parameter *neighbor* yang digunakan adalah tiga dengan akurasi mencapai 97.5% (Darma, 2019).

Penelitian tentang klasifikasi karakter penulisan tangan pernah dilakukan dengan menggunakan *Convolutional Neural Network* (CNN). Arsitektur CNN digunakan untuk mengklasifikasi tulisan tangan adalah VGG19 dengan layer *output* yang telah dimodifikasi. Hasil dari penelitian ini mendapatkan 90% akurasi dengan menggunakan gambar *grayscale* dan berhasil mengklasifikasi karakter penulisan orang dengan benar (Sudana et al., 2020).

Penelitian tentang deteksi berita bohong pernah dilakukan dengan menggunakan Model CNN, *Bidirectional LSTM*, dan *ResNet*. Proses pengubahan kata menjadi *token* pada penelitian ini dilakukan dengan menggunakan *embedding Word2Vec*, *GloVe*, dan *FastText*. Hasil dari penelitian ini mendapatkan kombinasi *embedding GloVe* dengan arsitektur *Bidirectional LSTM* memberikan hasil yang terbaik dengan akurasi melampaui 94.6%. (Sastrawan et al., 2022).

Penelitian tentang pengenalan karakter bahasa Bali pernah dilakukan dengan menggunakan *Tesseract OCR 5*. Pelatihan Model *Tesseract OCR 5* dilakukan menggunakan *tools* pelatihan *Tesseract OCR*. Penggunaan *tools* pelatihan *Tesseract OCR* dilakukan untuk mengekstrak dan mengenali karakter bahasa Bali yang

terdeteksi pada sistem. Penelitian ini berhasil mendapatkan skor *coincidence* sebesar 66.67% dengan memperhitungkan hierarki dari masing-masing karakter, kata, kalimat, serta paragraf dari penulisan bahasa Bali (Indrawan et al., 2022).

## 2.2 OCR

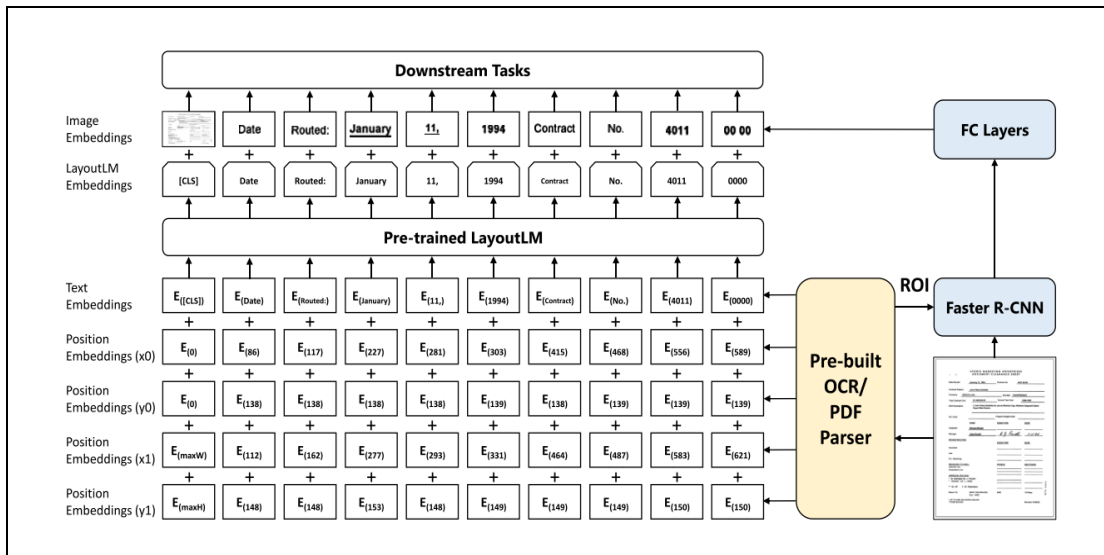
*Optical Character Recognition* (OCR) adalah proses konversi *text* dalam citra gambar menjadi format *text* yang dapat dibaca oleh mesin. Gambar yang digunakan untuk proses konversi *text* dapat berasal dari *text* yang dicetak maupun *text* hasil tulisan tangan (*handwritten character*) (Memon et al., 2019). Teknologi OCR ini merupakan bagian dari *artificial intelligence* yang banyak digunakan dalam bidang automasi seperti pemindaian dokumen, pembacaan pelat nomor kendaraan, verifikasi dokumen, dll. Beberapa arsitektur Model yang dapat digunakan sebagai dasar untuk melakukan OCR adalah Model *Long Short Term Memory* (LSTM), *Convolutional Neural Network* (CNN), dll. Model OCR sering dipadukan dengan Model *Natural Language Processing* (NLP) untuk meningkatkan akurasi dari pembacaan *text* (Hajiali et al., 2022). Arsitektur NLP yang sering digunakan dalam hal ini adalah *Bidirectional Encoder Representations from Transformers* (BERT).

## 2.3 Google Vision

*Google Vision* API merupakan sebuah Model *machine learning* yang telah dilatih untuk melakukan deteksi OCR melalui REST serta RPC API. *Google vision* API ini dapat melakukan deteksi pada gambar dan memberikan label pada masing-masing kategori yang terdeteksi pada gambar tersebut. Penggunaan *Google Vision* pada sistem ini digunakan untuk mendeteksi tiap *text* yang ada pada sebuah nota serta letak *bounding box* dari *text* tersebut (Google Cloud, 2022).

## 2.4 LayoutLM

LayoutLM merupakan sebuah Model *document understanding* yang dibuat untuk dapat memahami struktur dari sebuah dokumen. Model ini adalah dibuat dengan memperhatikan perkembangan permasalahan *Natural Language Processing* (NLP), di mana pada setiap Model NLP selalu berfokus pada *text-level manipulation*. Model LayoutLM ini dibuat dengan menggunakan interaksi antar informasi pada *teks* dalam sebuah dokumen beserta *layout* dari dokumen tersebut. Pengembangan Model ini dilakukan dengan menggunakan data dari dokumen-dokumen yang telah di-*scan* yang berasal dari berbagai macam kategori seperti surat, *memo*, *email*, *invoice*, *news*, *articles*, *questionnaire*, *resume*, dll (Xu et al., 2020).



**Gambar 2.1** Gambar Pelatihan LayoutLM (Xu et al., 2020)

Gambar pelatihan *LayoutLM* ini menjelaskan alur kerja pemrosesan data dalam Model *LayoutLM*. Pelatihan Model *LayoutLM* bekerja dengan cara membagi pemrosesan data menjadi dua tahapan. Tahap pertama adalah pemrosesan pada *text* dan *bounding box* yang merupakan posisi dari *text* tersebut. Tahap pertama dimulai dengan menggunakan *pre-built OCR parser* untuk mengekstrak informasi *text* beserta posisi *text* tersebut dalam bentuk *embedding*. Kedua informasi tersebut selanjutnya akan dilatih dengan menggunakan *pretrained LayoutLM*. Tahap kedua

adalah dengan mengambil *Region of Interest* (ROI) dari OCR *parser* yang sama, gambar akan diproses dengan menggunakan *Faster R-CNN* dengan *layer* akhir berupa *Fully Connected Layers*. Hasil dari proses pada kedua tahapan ini berupa *embedding* yang akan dibandingkan untuk menjadi hasil akhir dari Model ini.

## 2.5 Metriks Evaluasi Model

Evaluasi pada Model dilakukan menggunakan *confussion matrix* serta nilai akurasi, presisi, *recall*, dan *f1-score* yang didapatkan Model pada data uji. Penggunaan *confussion matrix* akan menunjukkan hasil prediksi label dari setiap kelas dengan lebih baik. *Confussion matrix* memberikan nilai *true positive* (TP), *false positive* (FP), *false negatif* (FN) dan *true negative* (TN) yang merepresentasikan hasil prediksi Model (Ramsay et al., 2011).

**Tabel 2.1** *Confusion Matrix*

	Prediksi Negatif	Prediksi Positif
Aktual Negatif	TN	FP
Aktual Positif	FN	TP

Tabel 2.1 merupakan tabel contoh penggunaan *confusion matrix*. *Confusion matrix* sering digunakan merepresentasikan hasil prediksi sebuah Model dalam permasalahan klasifikasi. Penjelasan dari istilah-istilah yang terdapat pada *confusion matrix* adalah sebagai berikut.

- True Positive* (TP) merupakan label positif yang berhasil diprediksi positif
- False Positive* (FP) merupakan label negatif yang salah diprediksi sebagai positif
- False Negative* (FN) merupakan label positif yang salah diprediksi negatif
- True Negative* (TN) merupakan label negatif yang berhasil diprediksi negatif

Hasil dari *confussion matrix* dapat digunakan untuk menghitung nilai akurasi, presisi, *recall*, dan *f1-score* dari Model yang telah dibuat. Perhitungan nilai ini dilakukan pada setiap label yang ada dalam proses pelatihan untuk mengetahui performa Model pada masing-masing label.

$$\text{Akurasi} = \frac{TP + TN}{TP + FP + FN + TN} \quad (2.1)$$

Akurasi dihitung dengan menjumlahkan nilai TP dan TN pada hasil *confusion matrix* dan membaginya dengan nilai total data yang diprediksi. Hasil akurasi menunjukkan keakuratan model dalam memprediksi kelas yang tepat.

$$\text{Presisi} = \frac{TP}{TP + FP} \quad (2.2)$$

Presisi dihitung dengan membagi nilai TP dengan jumlah TP dan FP pada hasil *confusion matrix*. Hasil presisi keakuratan prediksi model dalam memprediksi data positif.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.3)$$

*Recall* dihitung dengan membagi nilai TP dengan jumlah TP dan FN pada hasil *confusion matrix*. Hasil *recall* menunjukkan keberhasilan model dalam menemukan data berlabel positif.

$$F1 \text{ Score} = \frac{2 * (\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} \quad (2.4)$$

*F1-score* dihitung dengan mencari nilai *recall* dan presisi terlebih dahulu. Penggunaan *f1-score* sering dilakukan pada *dataset* yang memiliki kelas yang tidak seimbang sebagai alternatif dari nilai akurasi.



## **BAB III**

### **METODOLOGI PENELITIAN**

Bab III merupakan metodologi penelitian yang digunakan dalam Tugas Akhir *Finetuning* Model LayoutLM Untuk Pembacaan Nota Berbahasa Indonesia. Bab ini membahas tentang tempat dan waktu penelitian, data yang digunakan, gambaran umum sistem, alur aplikasi dan perancangan *database*.

#### **3.1 Tempat dan Waktu Penelitian**

Pembuatan Tugas Akhir sistem pembacaan nota ini dilakukan di Kampus Teknologi Informasi Universitas Udayana. Waktu pelaksanaan pembuatan Tugas Akhir dimulai pada bulan September 2022.

#### **3.2 Data Penelitian**

Data pelatihan yang digunakan dalam penelitian ini menggunakan data primer yang ditambahkan dengan data sekunder. Data primer merupakan data yang dikumpulkan oleh penulis untuk melakukan penelitian ini. Sedangkan data sekunder merupakan data yang tersedia secara publik dan dapat diambil untuk membantu penelitian.

##### **3.2.1 Data Primer**

Data Primer yang digunakan dalam *Finetuning* Model LayoutLM ini merupakan data yang dikumpulkan dari hasil bertransaksi pada berbagai restoran dan minimarket yang memberikan nota belanja *print out* digital. Total dari nota belanja yang telah dikumpulkan berjumlah 100. Nota-nota belanja ini nantinya akan difoto untuk dijadikan gambar yang digunakan sebagai data pelatihan Model LayoutLM.

### 3.2.2 Data Sekunder

Data Sekunder yang digunakan pada penelitian ini adalah *dataset WildReceipt* yang merupakan *dataset* nota belanja yang telah di-*scan*. Informasi yang terdapat pada *dataset WildReceipt* ini adalah informasi setiap kata, letak kata *bounding box*, serta label untuk setiap kata yang ada dalam nota belanja. *Dataset WildReceipt* ini memiliki data berjumlah 1267 data latih 472 data evaluasi (Theivaprakasham, 2022).

### 3.3 Instrumen Pembuatan Sistem

Instrumen pembuatan sistem memuat perangkat keras dan perangkat lunak yang digunakan dalam penelitian *Finetuning Model Layoutlm Untuk Pembacaan Nota Berbahasa Indonesia*. Pembuatan sistem memerlukan sebuah laptop untuk *deploy* sistem serta sebuah *smartphone* Android sebagai sarana pengujian aplikasi. Spesifikasi yang dimiliki oleh dari kedua perangkat tersebut adalah sebagai berikut.

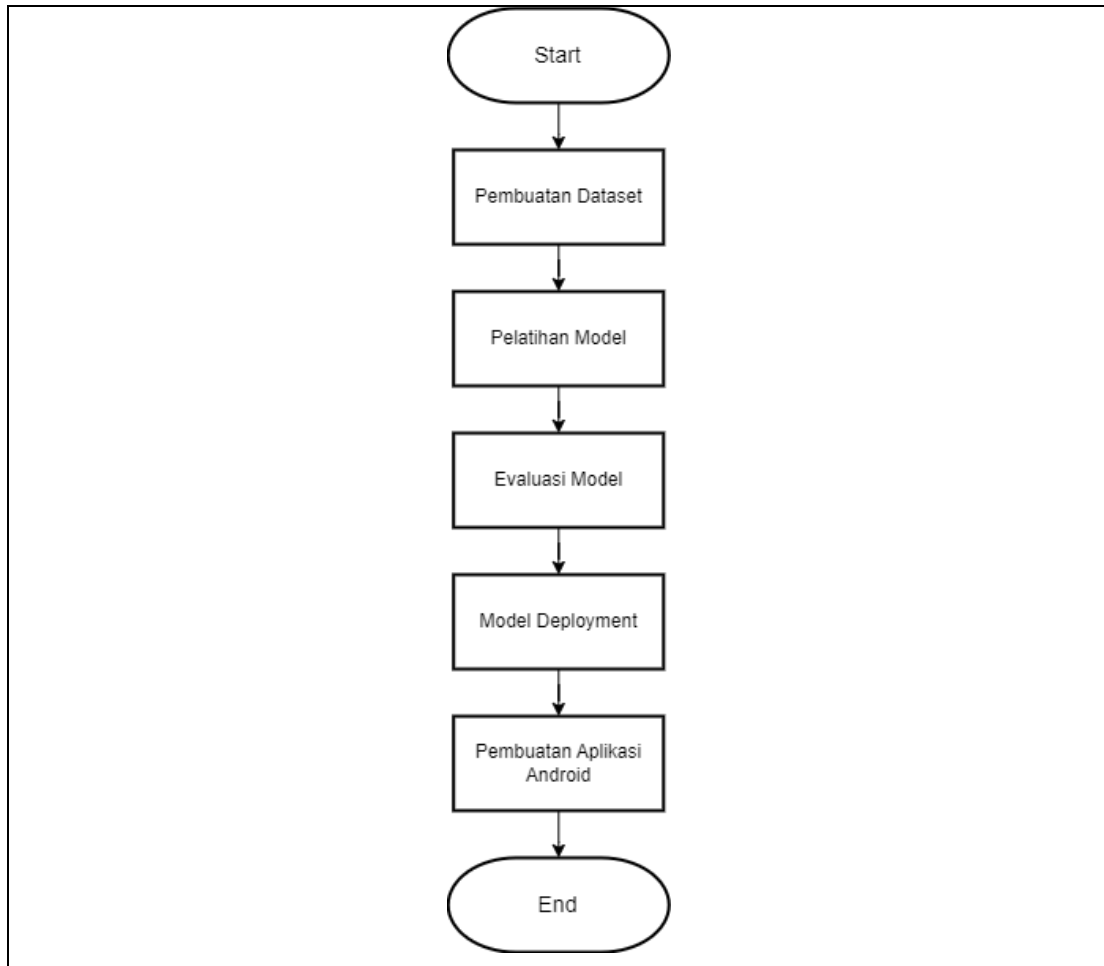
**Tabel 3.1** Spesifikasi Perangkat Keras

NO	Perangkat	Spesifikasi
1	Laptop	Windows, Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz, 24GB RAM, 512GB PCIe NVMe M.2 SSD, NVIDIA GeForce GTX 1660 TI
2	Smartphone Android	Android Oreo, API 28, ram 4gb

Tabel 3.1 memuat instrumen perangkat keras yang digunakan dalam pembuatan sistem pembacaan nota dengan OCR dan LayoutLM. Perangkat Laptop digunakan dalam perancangan sistem serta sebagai *web server* yang akan menerima dan mengolah gambar. Perangkat Android akan berperan dalam pengujian aplikasi.

### 3.4 Alur Penelitian

Alur penelitian membahas tentang proses pembuatan sistem *Finetuning Model LayoutLM* yang akan digunakan untuk membaca nota belanja. Proses yang dilakukan dalam pembuatan sistem ini meliputi pembuatan *dataset*, *Finetuning Model LayoutLM*, evaluasi Model, serta pembuatan aplikasi Android.

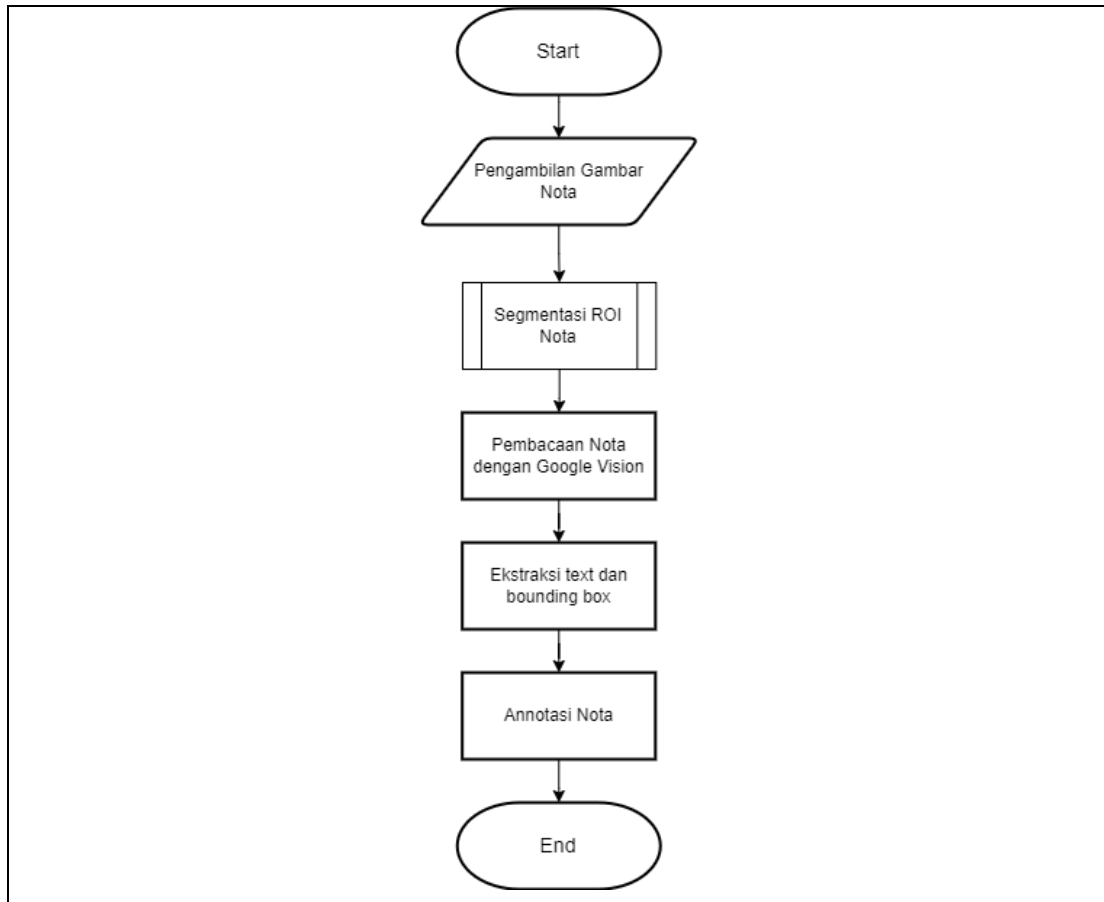


**Gambar 3.1** *Flowchart Alur Penelitian*

Gambar 3.1 merupakan alur dari pengerjaan penelitian ini yang menjelaskan proses-proses yang dilakukan dalam pembuatan sistem pembacaan nota belanja. Pengerjaan dimulai dari pembuatan *dataset* hingga menjadi sebuah aplikasi yang dapat digunakan pada *smartphone* Android.

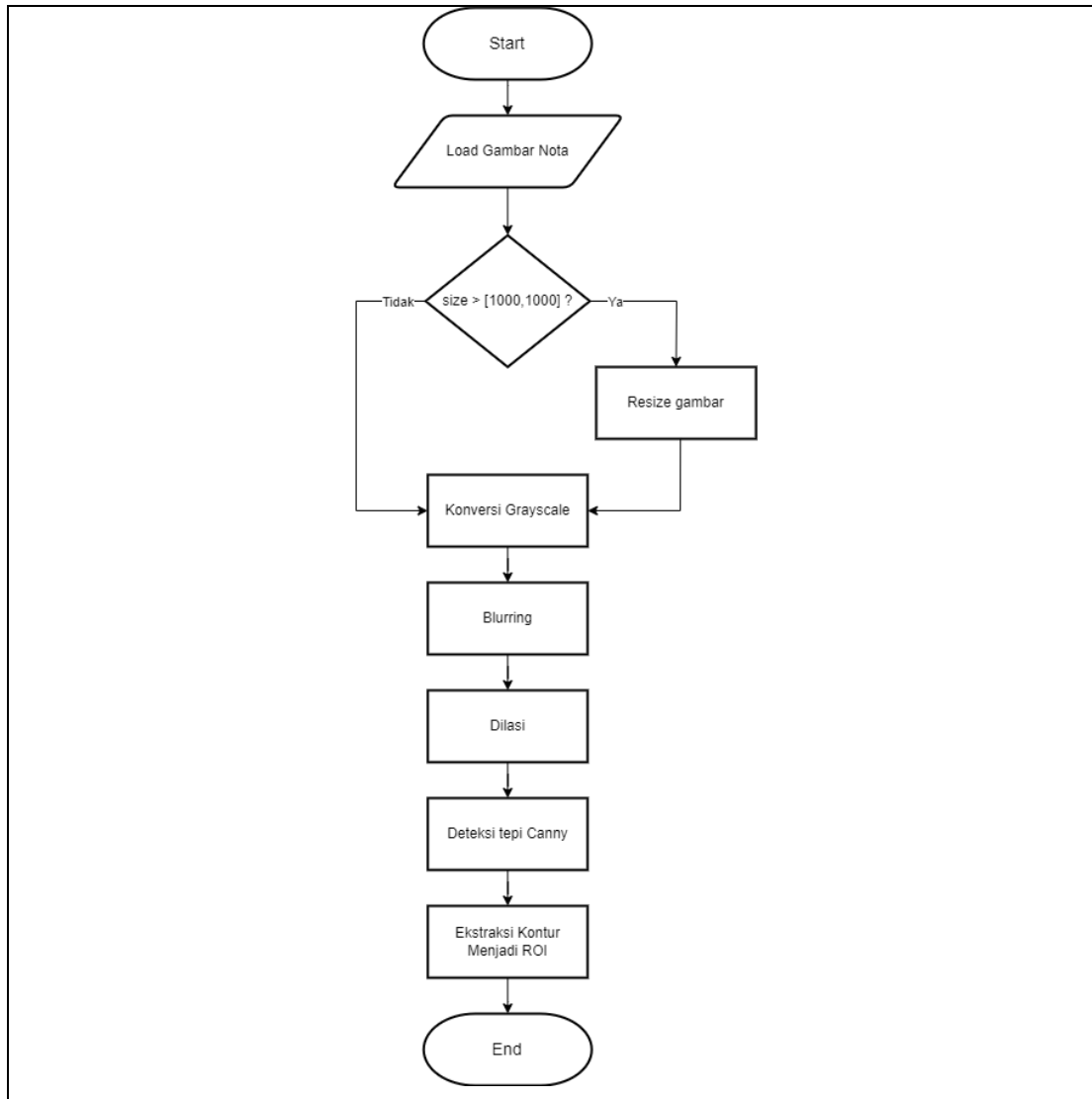
#### **3.4.1 Pembuatan Dataset**

Pembuatan *dataset* primer dilakukan dengan cara mengumpulkan nota belanja dari berbagai minimarket dan restoran. Nota belanja tersebut kemudian difoto dan diproses agar dapat digunakan dalam pelatihan Model.



**Gambar 3.2** Flowchart Pembuatan Dataset

Gambar 3.2 menjelaskan tentang alur proses pembuatan *dataset* primer dalam penelitian ini. Proses pembuatan *dataset* ini menghasilkan sebuah file *.json* yang memuat informasi setiap kata, *bounding box*, dan label yang terdapat pada sebuah nota.



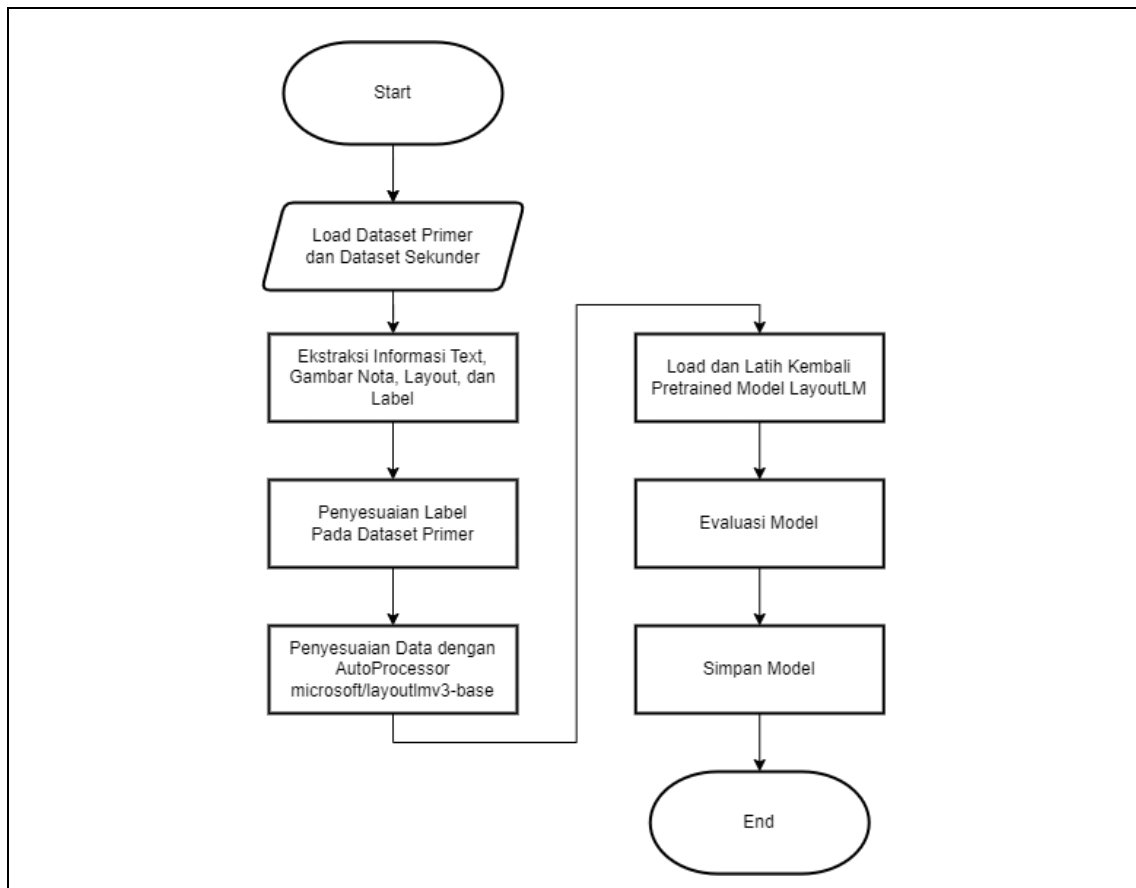
**Gambar 3.3** Flowchart Segmentasi ROI Nota

Gambar 3.3 memuat tentang proses segmentasi *ROI* pada gambar nota untuk mendapatkan gambar nota yang menyerupai hasil *scan*. Gambar hasil segmentasi berupa nota berwarna dengan ukuran panjang maksimal 1000 *pixel*.

### 3.4.2 Finetuning Model LayoutLM

Model *Finetuning* adalah sebuah istilah dalam *machine learning* untuk menyelesaikan permasalahan *Natural Language Processing* (NLP). *Finetuning* adalah merupakan proses yang serupa seperti *transter learning* pada *Convolutional*

Model yang dapat menggunakan arsitektur yang sama untuk menyelesaikan berbagai permasalahan. *Finetuning* Model LayoutLM dilakukan dengan melatih Model dasar dengan data baru yaitu nota yang telah dikumpulkan.



**Gambar 3.4** Flowchart Pelatihan LayoutLM

Gambar 3.4 merupakan *flowchart* dari proses pelatihan Model LayoutLM. Alur pelatihan dimulai dengan memuat *dataset* dan melakukan ekstraksi informasi *layout* dan label dari setiap kata dalam nota. Informasi ini selanjutnya akan dimuat ke dalam proses *auto encoder* yang dimiliki oleh Model LayoutLM untuk menyamakan format data dengan format *original* pelatihan. Data *encoding* hasil pengubahan format ini akan digunakan untuk melakukan *Finetuning* pada Model LayoutLM. Hasil terbaik dari proses pelatihan Model akan disimpan untuk digunakan dalam mendeteksi nota.

### 3.4.3 Evaluasi Model

Evaluasi Model *finetuned* LayoutLM dilakukan dengan cara membandingkan hasil prediksi label dari setiap kata, dengan label aslinya. Evaluasi Model akan dilakukan dengan menggunakan *confussion matrix* serta *classification report* yang akan menunjukkan hasil pada setiap label yang berbeda.

#### 3.4.3.1 Confussion Matrix

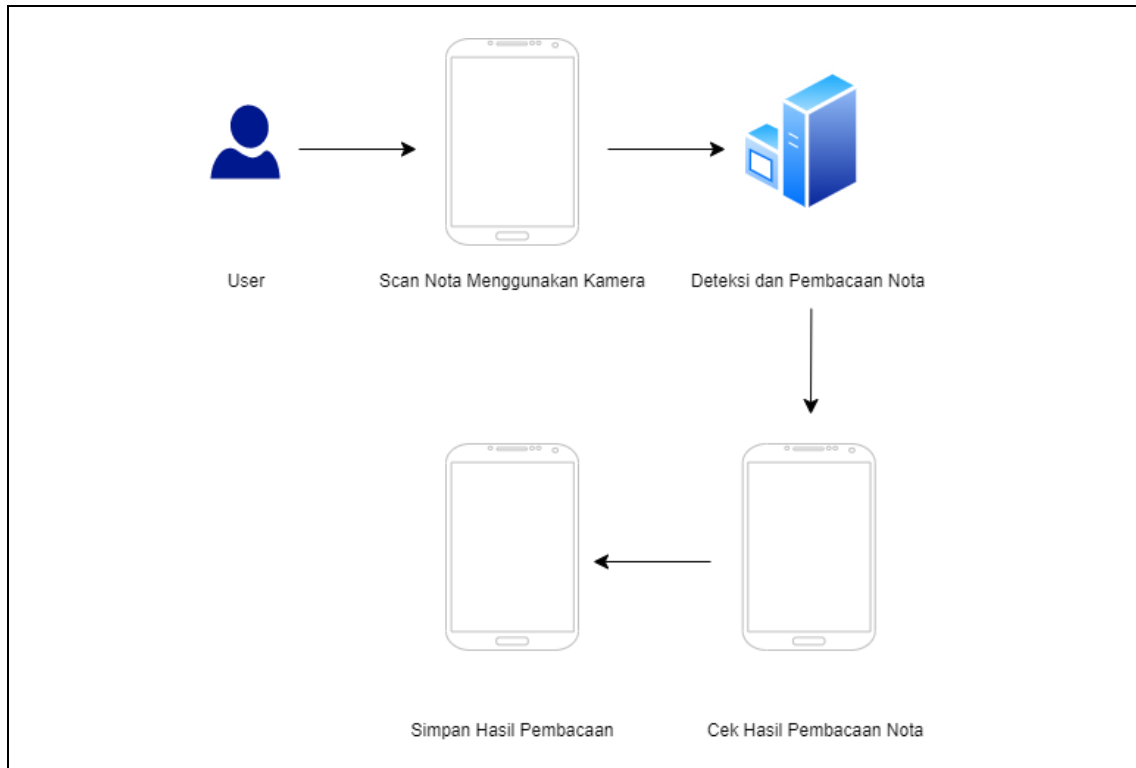
Pada tahap penilaian *confussion matrix* hasil deteksi pada setiap label dapat dideteksi dengan melihat jumlah label yang berhasil dideteksi dengan benar pada diagonal pada hasil dari *confussion matrix*. Pada data yang digunakan terdapat 15 label yang harus dibedakan oleh Model antara lain label *ignore*, *Store\_name\_value*, *Date\_value*, *Time\_value*, *Prod\_item\_key*, *Prod\_item\_value*, *Prod\_quantity\_key*, *Prod\_quantity\_value*, *Prod\_price\_key*, *Prod\_price\_value*, *Subtotal\_key*, *Subtotal\_value*, *Total\_key*, *Total\_value*, dan *Others*.

#### 3.4.3.2 Classification Report

Pada penilaian *classification report* hasil nilai akurasi, presisi, dan *recall* dibedakan menjadi 2 tahap. Tahap pertama adalah penilaian pada 15 label yang terdapat pada *dataset* secara langsung. Tahap penilaian kedua adalah penilaian pada 8 label yang merupakan informasi yang ingin diekstrak dari hasil pembacaan nota, yaitu *Store\_name\_value*, *Date\_value*, *Time\_value*, *Prod\_item\_value*, *Prod\_quantity\_value*, *Prod\_price\_value*, *Subtotal\_value*, dan *Total\_value*.

## 3.5 Gambaran Umum Sistem

Gambaran umum sistem menjelaskan tentang desain perancangan dari sistem yang akan digunakan pada pembuatan sistem pembacaan nota ini. Sistem yang dibuat akan terbagi menjadi rancangan server beserta rancangan aplikasi android sebagai berikut.



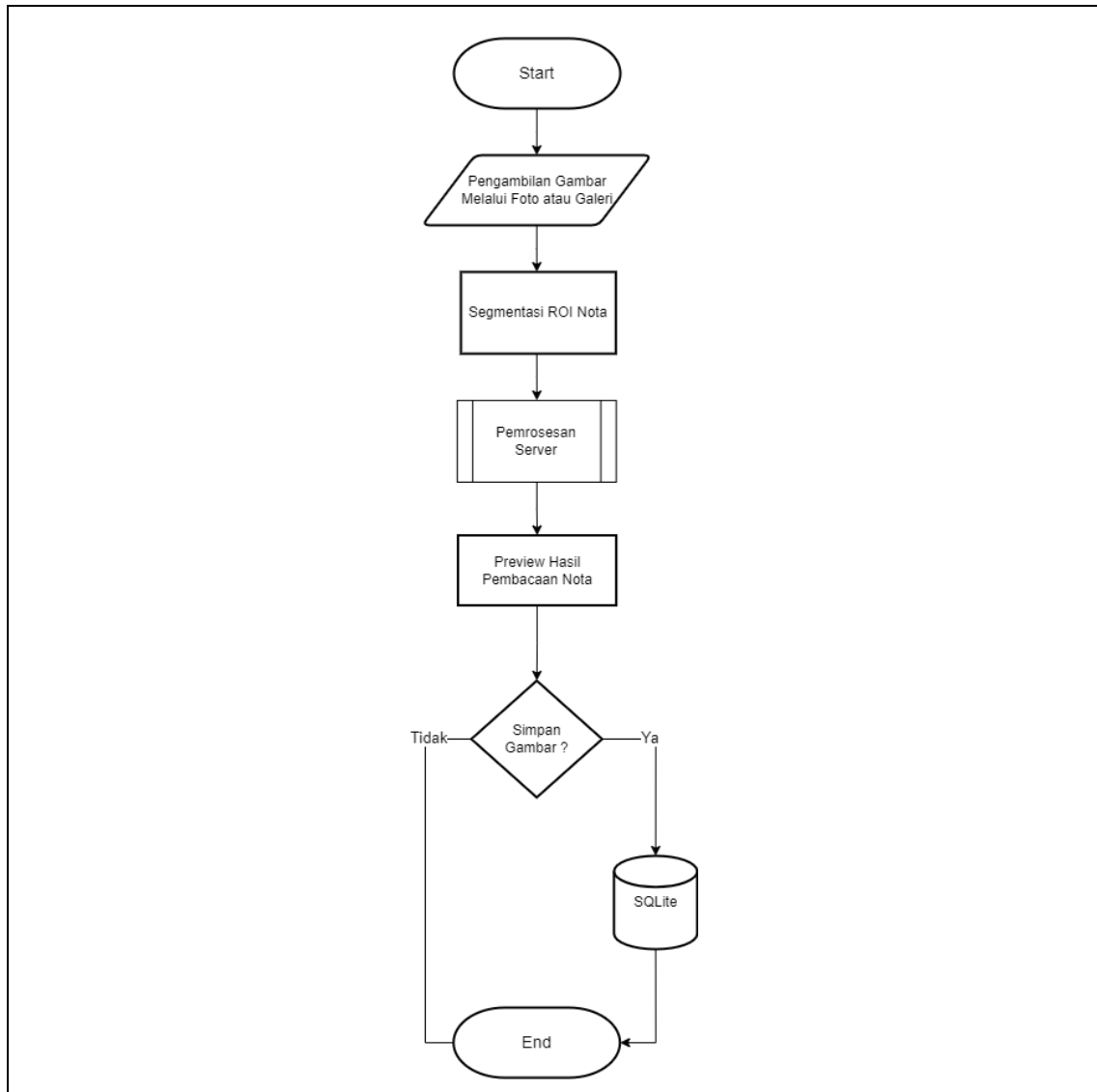
**Gambar 3.5** *Gambaran Umum Sistem*

Gambar 3.5 merupakan gambaran umum dari aplikasi android yang akan dibuat. Aplikasi ini memiliki dua fungsi utama yaitu untuk mengirim foto dari nota belanja kepada *web server* menggunakan *API Call* dan menyimpan hasil pembacaan nota yang telah diterima dari *web server* serta untuk melihat *history* data belanja bulanan yang tersimpan pada sistem.

### 3.6 Alur Aplikasi

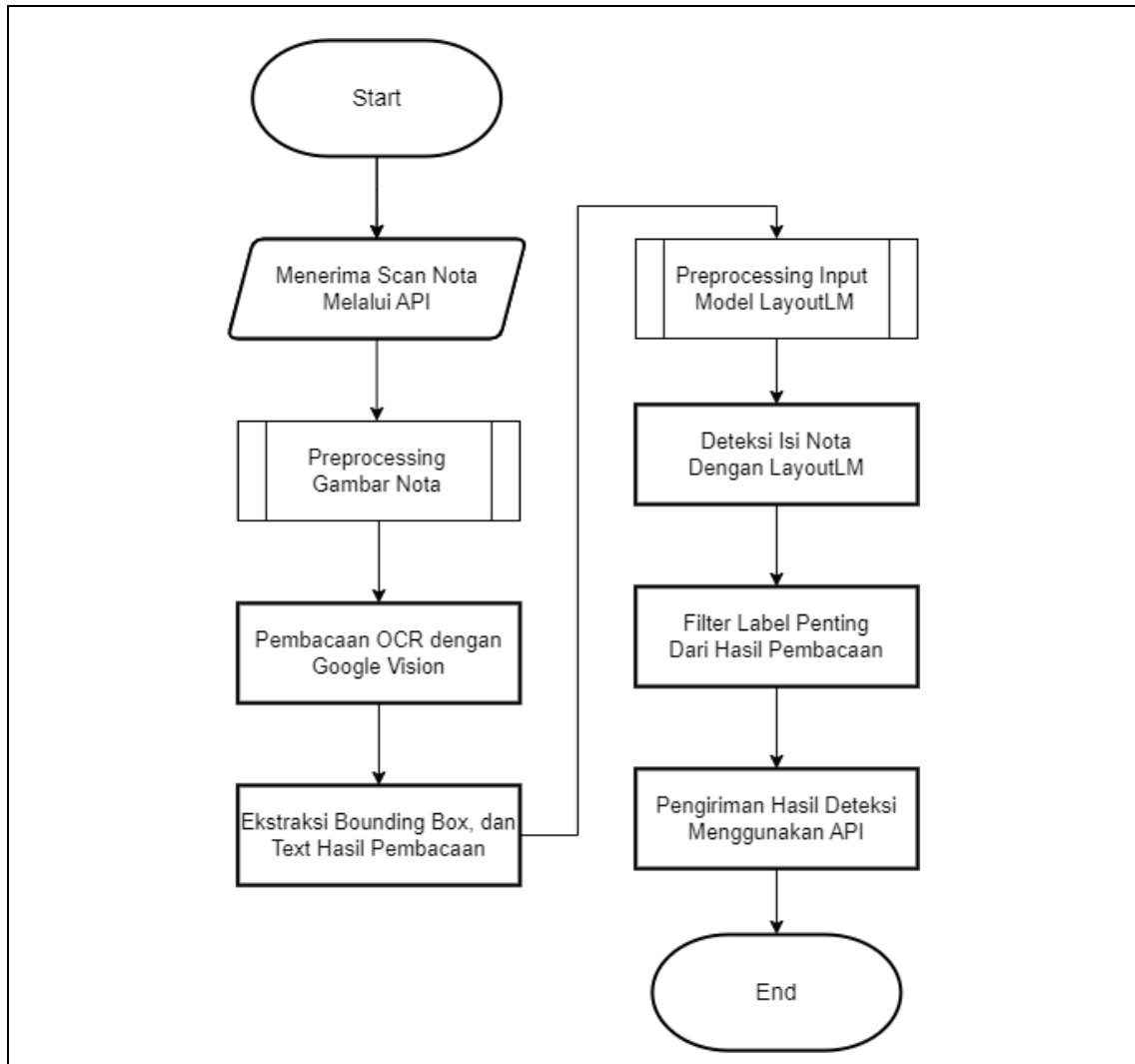
Alur aplikasi menjelaskan tentang alur pemrosesan data yang terdapat pada aplikasi pembacaan nota belanja ini. Terdapat dua buah fungsi utama dari aplikasi yang dibuat yaitu fungsi pembacaan dan penyimpanan nota serta fungsi melihat riwayat nota yang disimpan.





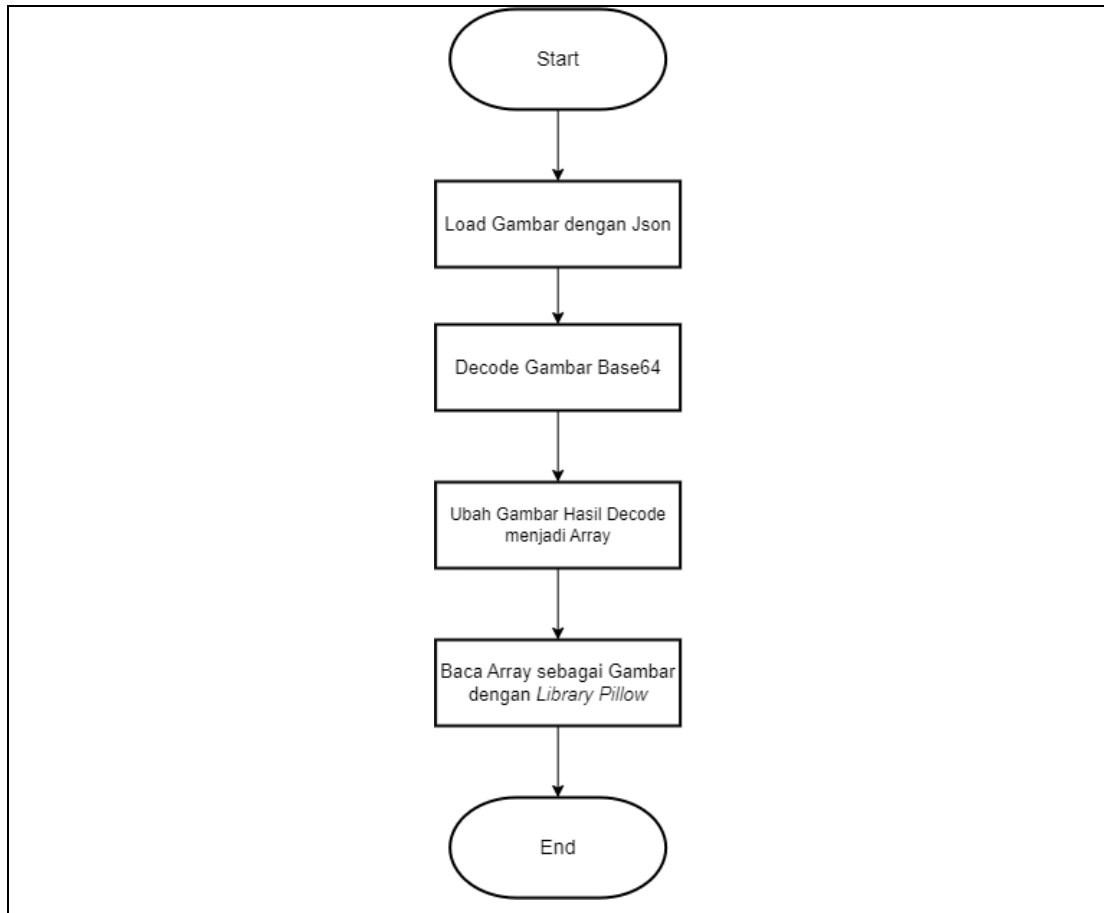
**Gambar 3.6** Flowchart Pembacaan Nota pada Aplikasi Android

Gambar 3.6 merupakan *flowchart* alur pembacaan nota yang terdapat pada aplikasi android. Pembacaan nota belanja dapat dilakukan dengan mengirimkan foto dari nota yang ingin dibaca kepada *web server*. Hasil dari pembacaan nota akan dikirim kembali oleh *web server* untuk dapat disimpan pada sistem android.



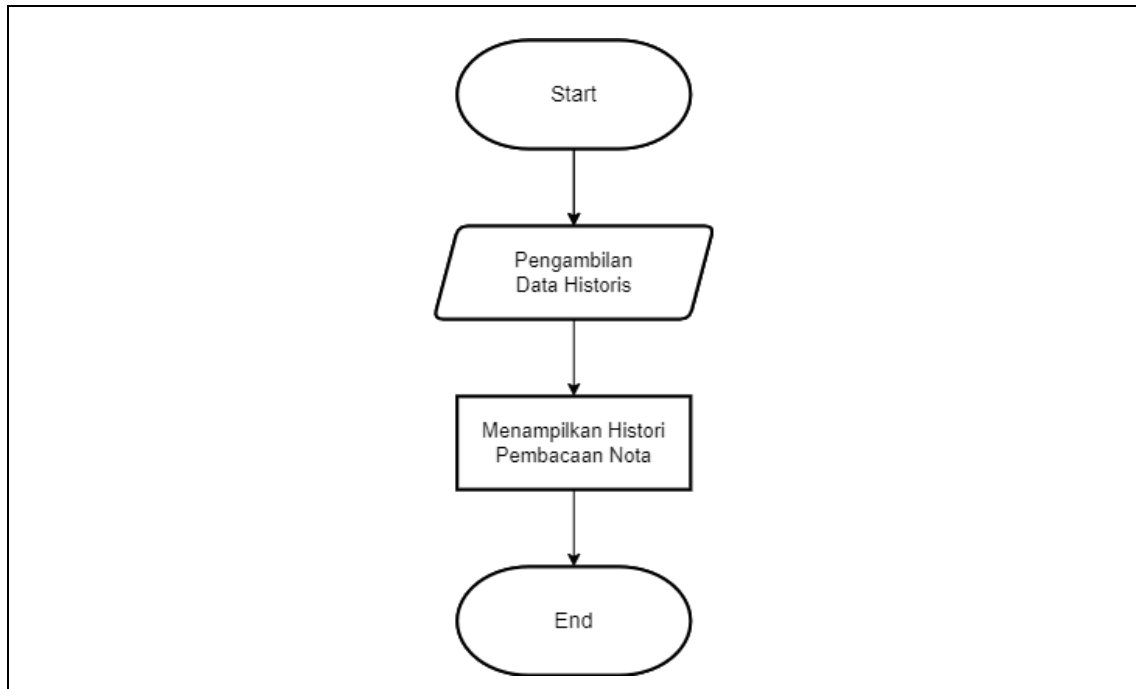
**Gambar 3.7** Flowchart Proses Pembacaan Nota pada Web Server

Gambar 3.7 merupakan gambaran *flowchart* pemrosesan pada *web server* yang dijalankan dengan menggunakan *framework Flask*. *Web server* akan menerima gambar yang telah dikirimkan dari aplikasi android dan melakukan *preprocessing* pada gambar tersebut agar dapat terbaca pada sistem. Gambar tersebut selanjutnya akan diproses kembali untuk mendapatkan *embedding text* dan *layout* nota serta hasil prediksi Model. Hasil prediksi dari Model kemudian akan diinterpretasikan menjadi data tanggal, barang belanja, subtotal item, serta total belanja dan dikirimkan kembali pada aplikasi android.



**Gambar 3.8** *Flowchart Preprocessing Gambar pada Server*

Gambar 3.8 merupakan gambaran *flowchart* proses pemrosesan gambar pada *server*. Pertama gambar yang diterima melalui *API Call* akan dibaca menggunakan *library Json* untuk membaca *file enkripsi*. *File* enkripsi tersebut lalu akan diterjemahkan menggunakan *library Base64*. Hasil terjemahan *file* kemudian diproses untuk dijadikan sebuah *array* dan dibaca sebagai sebuah gambar dengan menggunakan *library Pillow*.

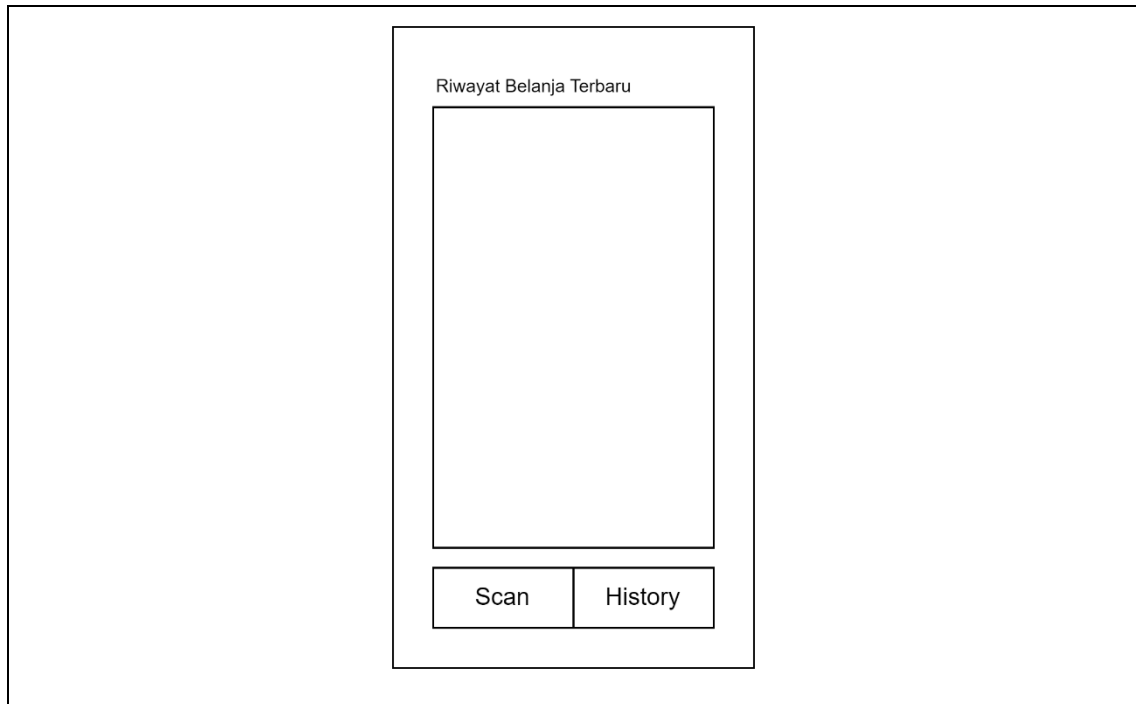


**Gambar 3.9** *Flowchart Melihat Histori Pembacaan Nota pada Aplikasi Android*

Gambar 3.9 merupakan *flowchart* alur melihat riwayat pembacaan nota yang terdapat pada aplikasi android. Proses melihat riwayat pembacaan nota diawali dengan pengambilan data historis dari *database SQLite* Android. Data historis kemudian ditampilkan pada aplikasi android.

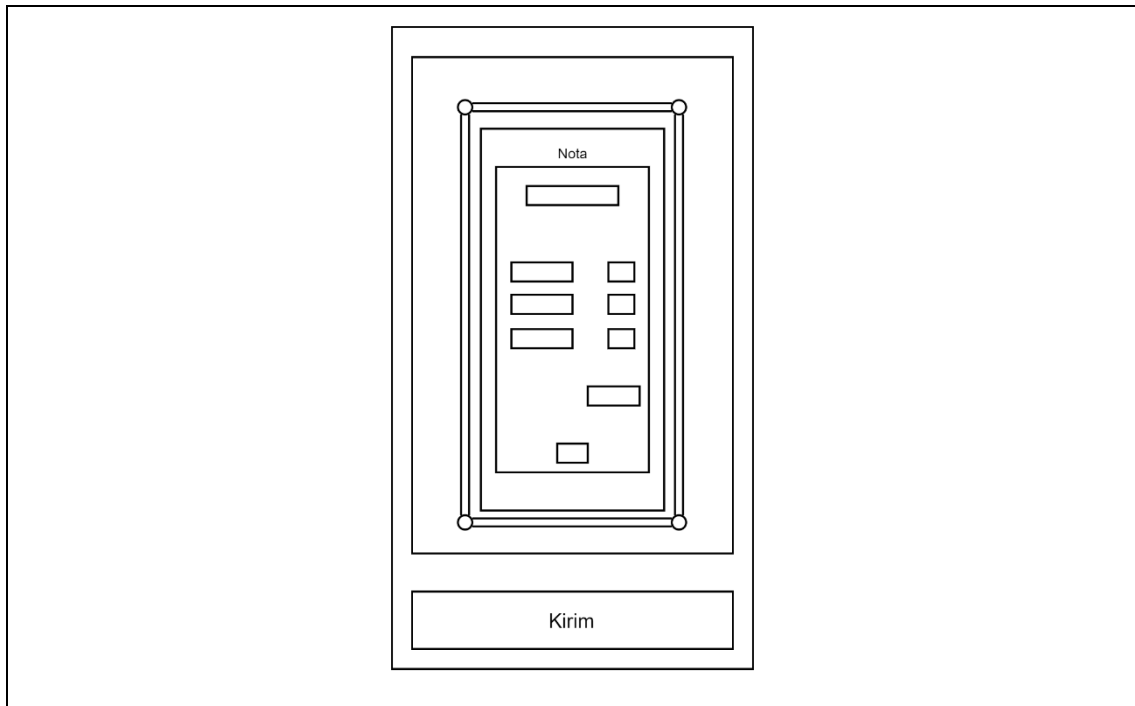
### 3.7 Rancangan Sistem

Rancangan sistem memuat tentang tampilan proses yang ada pada sistem android yang akan dibuat. Rancangan dari sistem ini memuat proses pengambilan gambar, segmentasi nota belanja, tampilan hasil deteksi, serta halaman *history* yang memuat informasi riwayat pembacaan nota yang telah tersimpan.



**Gambar 3.10** *Rancangan Tampilan Depan Sistem*

Gambar 3.10 adalah gambar rancangan tampilan depan saat aplikasi ini dibuka. Aplikasi akan menampilkan riwayat hasil *scan* terbaru, serta dua buah tombol untuk men-*scan* nota baru serta melihat riwayat hasil *scan* dengan lebih lengkap.



**Gambar 3.11** *Rancangan Segmentasi Nota*

Gambar 3.11 adalah gambar rancangan segmentasi nota pada aplikasi Android. Terdapat empat buah titik yang dapat digunakan untuk menentukan ROI pada gambar nota. Setelah titik-titik tersebut dipasang tombol kirim akan ditekan untuk mengirimkan gambar nota ke *web server*.

The diagram illustrates a receipt preview layout. It consists of a main container with a central 'Nota' (Receipt) area and a right-hand sidebar. The 'Nota' area includes a header, a barcode placeholder, a list of items with checkboxes, and a total field. The sidebar contains input fields for 'Nama Toko', 'Tanggal Waktu', and 'Total', followed by a table with three rows of 'barang' and 'harga'. At the bottom are a back arrow button and a 'Simpan' button.

**Gambar 3.12** Rancangan Hasil Preview Pembacaan

Gambar 3.12 adalah gambar rancangan halaman *preview* setelah nota dikirim dan diberikan hasil pembacaan oleh *web server*. Gambar akan menampilkan informasi-informasi penting yang akan disimpan oleh aplikasi. Terdapat dua buah tombol untuk kembali ke halaman sebelumnya dan membatalkan *scan* serta tombol untuk menyimpan hasil *scan*.

Rentang Waktu	
	⇌
Total Pengeluaran	
Rincian	

**Gambar 3.13** *Rancangan Riwayat Hasil Scan*

Gambar 3.13 adalah gambar rancangan halaman riwayat hasil *scan*. Terdapat rentang waktu yang dapat dipilih serta total pengeluaran akan terlihat sesuai rentang waktu yang dipilih. Data yang sesuai dengan rentang waktu akan terlihat pada daftar rincian yang memuat informasi nama toko, tanggal, serta total belanja pada transaksi tersebut.



## BAB IV

### HASIL DAN PEMBAHASAN

Bab IV hasil dan pembahasan membahas mengenai proses pengerjaan sistem yang digunakan dalam melakukan *Finetuning* Model LayoutLM untuk membaca nota berbahasa Indonesia beserta hasil yang ditemukan selama pengerjaan sistem.

#### 4.1 Pembuatan Dataset

Pembuatan *dataset* dilakukan dengan cara mengumpulkan nota belanja yang didapatkan setelah bertransaksi di toko Alfamart, Circle K, Indomaret, Mixue, dll. Nota-nota tersebut selanjutnya akan difoto menggunakan perangkat *smartphone*.

##### 4.1.1 Segmentasi Gambar

Proses segmentasi dilakukan pada gambar nota yang telah dikumpulkan untuk menghilangkan *background* dan mendapatkan gambar yang berfokus kepada nota yang merupakan informasi utama dalam pengerjaan sistem.

```
file_name = '20221013_192759.jpg'
file_path = os.path.join(DIR, 'Nota', file_name)
img_read = cv2.imread(file_path)
img_read = cv2.cvtColor(img_read, cv2.COLOR_BGR2RGB)
if img_read is None:
    raise Exception(f"Image {file_name} not found")

resize_ratio = 1000 / img_read.shape[0]
img_rezise = resize_img(img_read, resize_ratio)
gray = cv2.cvtColor(img_rezise, cv2.COLOR_BGR2GRAY)
plt.imshow(img_rezise)
plt.show()
```

**Kode Program 4.1** Pembacaan File Gambar Nota

Kode Program 4.1 merupakan kode program yang digunakan untuk membaca *file* foto dengan menggunakan *library* OpenCv. Pertama dilakukan proses `resize_img` untuk memperkecil ukuran gambar dan operasi `cv2.cvtColor` untuk mengambil gambar dalam bentuk *grayscale*.



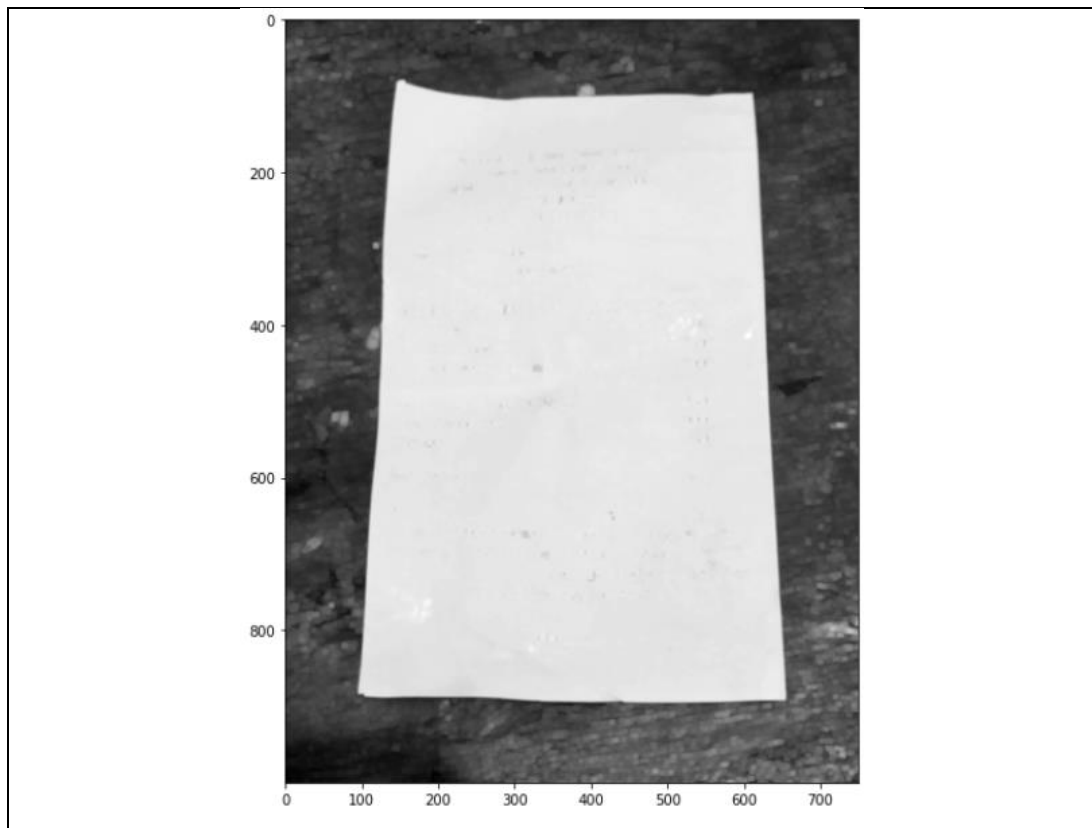
**Gambar 4.1** Hasil Proses Pembacaan Gambar Serta Resize

Gambar 4.1 adalah hasil dari proses pembacaan gambar dan proses `resize_img` yang telah dilakukan. Gambar nota hasil akan memiliki ukuran panjang 1000. Selain itu sebuah gambar *grayscale* akan disimpan dalam variabel `gray`.

```
blurred = cv2.GaussianBlur(gray, (3, 3), 3)
rectKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
dilated = cv2.dilate(blurred, rectKernel)
plt.figure(figsize=(10,10))
plt.imshow(dilated, cmap='gray')
plt.show()
```

**Kode Program 4.2** Preprocessing Nota Tahap 1

Kode Program 4.2 adalah kode program untuk memproses gambar *grayscale* dengan menggunakan proses *blurring*, dan proses dilasi. Proses *blurring* dilakukan dengan menggunakan `GaussianBlur` pada gambar *grayscale* dan proses `dilate` pada gambar yang telah di-*blur*.



**Gambar 4.2** Hasil Proses Blurring dan Dilasi Pada Gambar

Gambar 4.2 adalah hasil dari proses *blurring* dan dilasi pada gambar *grayscale*. Proses *blurring* dan proses dilasi dilakukan pada gambar agar tulisan pada gambar menjadi tidak terbaca saat dilakukan deteksi tepi.

```
def auto_canny(image, sigma=1):
    # compute the median of the single channel pixel intensities
    v = np.median(image)

    # apply automatic Canny edge detection using the computed
    median
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    edged = cv2.Canny(image, lower, upper)

    # return the edged image
    return edged
```

```

edged = auto_canny(dilated)

plt.figure(figsize=(10,10))

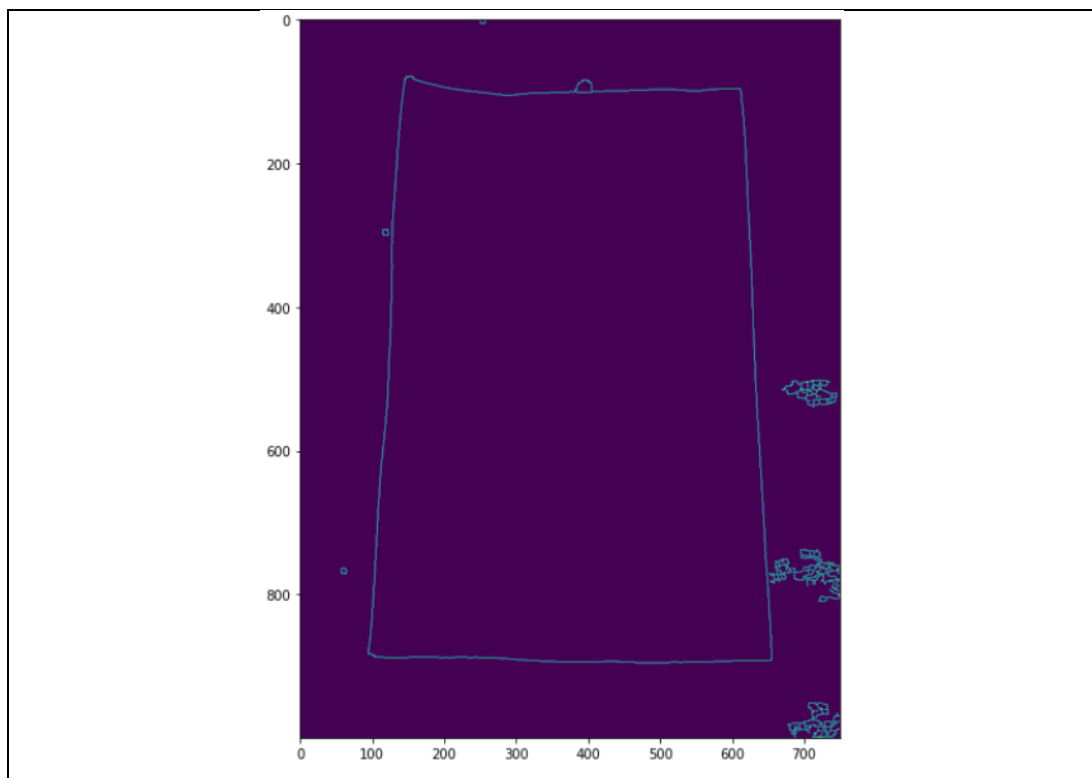
plt.imshow(edged)

plt.show()

```

**Kode Program 4.3** *Proses Deteksi Tepi Canny*

Kode Program 4.3 merupakan kode program yang digunakan untuk mendeteksi tepi nota agar dapat dilakukan proses segmentasi. Hasil dari proses deteksi tepi ini dilanjutkan dengan mendeteksi area *contour* terbesar.



**Gambar 4.3** *Hasil Deteksi Tepi Dengan Menggunakan Canny*

Gambar 4.3 merupakan hasil dari proses deteksi tepi pada nota yang sudah di-*blur*. Hasil dari proses deteksi ini adalah tepi dari nota dapat terlihat dan tulisan yang ada di dalam nota tidak terdeteksi sebagai tepi yang membuat proses segmentasi dapat berjalan dengan lebih baik.

```

detected_lines = cv2.HoughLinesP(edged, rho = 0.5, theta =
1*np.pi/180, threshold = 30, minLineLength = 20, maxLineGap = 100)
edged_line = np.zeros_like(edged)
line_extender = 0
for line in detected_lines:
    x1, y1, x2, y2 = line[0]

    is_vertical = abs(x1 - x2) < abs(y1 - y2)
    if is_vertical:
        x1 = int(x1-(abs(x1 - x2)/2*line_extender)*0.314)
        x2 = int(x2+(abs(x1 - x2)/2*line_extender)*0.314)

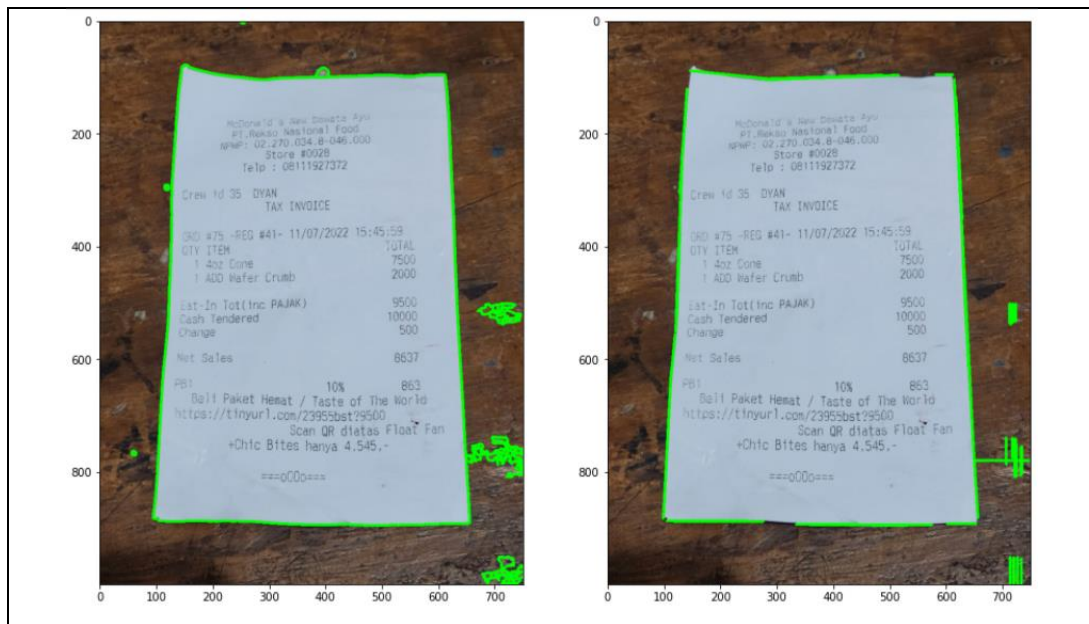
        if y1<y2:
            y1 = int(y1-(abs(y1 - y2)/2*line_extender))
            y2 = int(y2+(abs(y1 - y2)/2*line_extender))
        else:
            y1 = int(y1+(abs(y1 - y2)/2*line_extender))
            y2 = int(y2-(abs(y1 - y2)/2*line_extender))
    else:
        x1 = int(x1-(abs(x1 - x2)/2*line_extender))
        x2 = int(x2+(abs(x1 - x2)/2*line_extender))
        y1 = int(y1-(abs(y1 - y2)/2*line_extender)*0.314)
        y2 = int(y2+(abs(y1 - y2)/2*line_extender)*0.314)

    cv2.line(edged_line, (x1, y1), (x2, y2), (255), 1)
#main and backup method
contours_m, _ = cv2.findContours(edged, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
contours_b, hierarchy = cv2.findContours(edged_line,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
all_contours_m = cv2.drawContours(img_resize.copy(), contours_m, -
1, (0,255,0), 3)
all_contours_b = cv2.drawContours(img_resize.copy(), contours_b, -
1, (0,255,0), 3)
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(all_contours_m)
plt.subplot(1,2,2)
plt.imshow(all_contours_b)
plt.show()

```

**Kode Program 4.4** *Deteksi Kontur Hasil tepi Canny*

Kode Program 4.4 adalah kode program yang digunakan untuk mengekstrak tepi pada nota yang telah dideteksi tepinya oleh deteksi tepi *Canny*. Kode program ini akan menampilkan gambar asli dengan kontur tepi yang telah terdeteksi, serta membuat garis tambahan pada tepi dengan menggunakan fungsi `cv2.HoughLinesP`.



**Gambar 4.4** Deteksi Kontur Hasil tepi *Canny*

Gambar 4.4 adalah gambar hasil deteksi kontur yang digambar pada gambar asli. Terdapat dua buah metode yang digunakan dalam penggambaran kontur. Metode pertama adalah dengan menggambarkan kontur secara langsung setelah deteksi tepi *Canny*. Metode kedua adalah dengan menggunakan fungsi `cv2.HoughLinesP` pada hasil *Canny* untuk mendeteksi garis terluar pada nota.

```
largest_contours = sorted(contours_m, key = cv2.contourArea,
reverse = True) [0:4]

image_with_largest_contours = cv2.drawContours(img_resize.copy(),
largest_contours, -1, (0,255,0), 3)

plt.figure(figsize=(10,10))
plt.imshow(image_with_largest_contours)
plt.show()
```

**Kode Program 4.5** Pemilihan Kontur

Kode program 4.5 adalah kode program yang digunakan untuk pemilihan kontur terbesar yang akan digunakan untuk segmentasi nota. Kontur dengan area terbesar yang terdeteksi dengan fungsi `cv2.contourArea` akan terlihat pada gambar dan sisanya akan diabaikan menyesuaikan kategori pohon kontur.



**Gambar 4.5** Hasil Pemilihan Kontur

Gambar 4.5 adalah gambar hasil pemilihan kontur pada nota. Kontur yang terbesar akan berada pada gambar sesuai dengan area dan hierarki pohon kontur yang dimiliki oleh kontur tersebut.

```
try:
    receipt_contour = get_receipt_contour(largest_contours)

    detected_receipt = cv2.drawContours(img_resize.copy(),
[receipt_contour], -1, (0, 255, 0), 2)

    result = wrap_perspective(img_resize.copy(),
contour_to_rect(receipt_contour, corner_tolerance = 0))

    print("using main method successfully")
except:

    print("using backup method")
    longest = 0

    for i, cont in enumerate(largest_contours):
        x, y, w, h = cv2.boundingRect(largest_contours[i])

        cur_length = w+h

        if (cur_length > longest):
```

```

        longest = cur_length
        biggest_idx = i

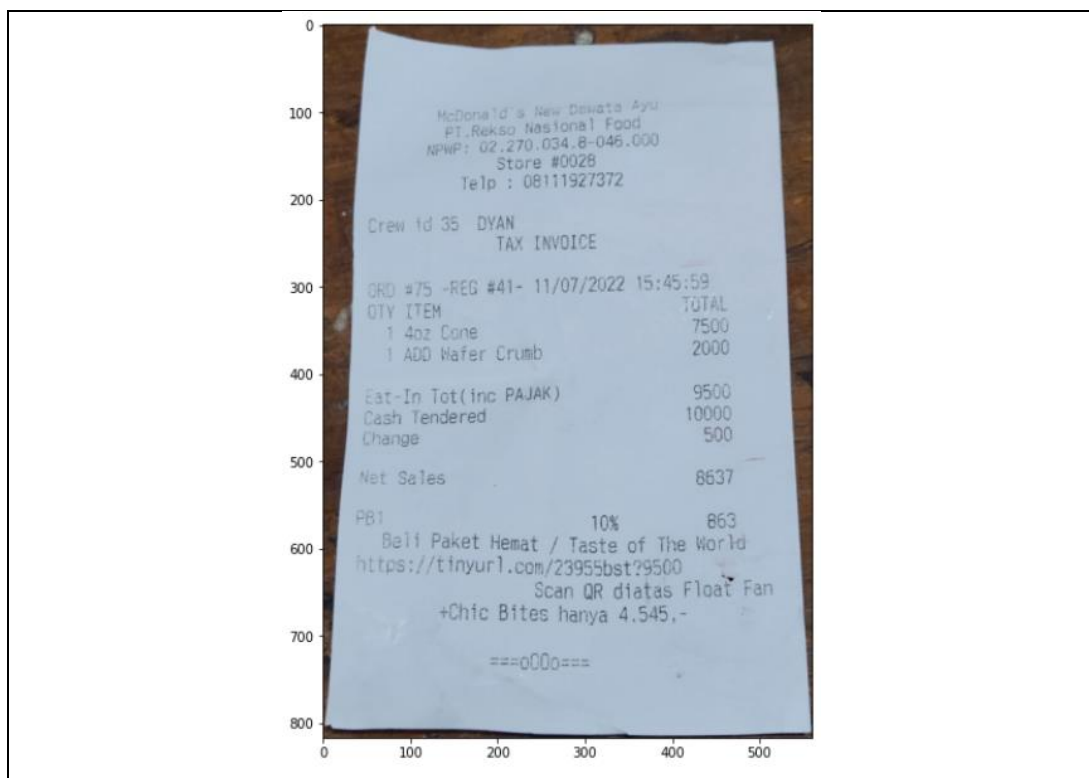
    x,y,w,h = cv2.boundingRect(largest_contours[biggest_idx])
    result = img_rezise[y:y+h, x:x+w]

plt.figure(figsize=(10,10))
plt.imshow(result)
plt.show()

```

**Kode Program 4.6 Segmentasi Nota**

Kode program 4.6 adalah kode yang digunakan untuk melakukan segmentasi pada nota. Kode program ini menggabungkan tahap-tahap deteksi nota sebelumnya serta mengambil ordinat yang dimiliki oleh kontur terbesar untuk melakukan segmentasi.



**Gambar 4.6 Nota Hasil Segmentasi**

Gambar 4.6 adalah hasil setelah nota mengalami segmentasi. Gambar hasil segmentasi akan berfokus hanya pada nota agar pembacaan OCR dapat dilakukan dengan benar. Setelah ini setiap kata pada gambar akan dideteksi dengan menggunakan API *Google Vision*.



#### 4.1.2 Pembacaan Karakter

Pembacaan karakter adalah tahap pembacaan setiap kata yang terdapat pada nota dengan menggunakan *Google Vision*. Proses pembacaan nota ini akan dibantu dengan *library Layout Parser*.

```
ocr_agent = lp.GCVAgent.with_credential(os.path.join(DIR, 'gcv_credential.json'), languages = ['id'])

res = ocr_agent.detect(image_result, return_response=True)
texts = ocr_agent.gather_text_annotations(res)
lp.draw_text(image_result, texts, font_size=12, with_box_on_text=True, text_box_width=3)
```

**Kode Program 4.7** Pembacaan Karakter Dengan *Google Vision*

Kode program 4.7 adalah kode program yang digunakan untuk membaca karakter dengan *Google Vision*. Pertama-tama dilakukan inisiasi agen pembaca OCR pada *Layout Parser* dengan menggunakan kredensial dari API *Google Vision*. Lalu dengan menggunakan fungsi *detect*, nota akan dibaca dan menghasilkan *bounding box* serta kata yang terdapat pada nota.



**Gambar 4.7** Hasil Deteksi *Google Vision*

Gambar 4.7 adalah gambar hasil pembacaan karakter oleh *Google Vision*. Pada gambar bagian kiri terdapat letak penggambaran letak *bounding box* yang memuat sebuah *text* dari setiap kata yang terdeteksi oleh *Google Vision*. Pada bagian kanan merupakan gambar nota yang dikirimkan pada *Google Vision*.

```

import json
from tqdm.notebook import tqdm
receipt_list = {}

ocr_agent =
lp.GCVAgent.with_credential(os.path.join(DIR, 'gcv_credential.json'
), languages = ['id'])

for filename in tqdm(os.listdir('Nota_Segmented')):
    filepath = os.path.join(DIR, 'Nota_Segmented', filename)
    image_result = get_receipt(filepath)
    res = ocr_agent.detect(image_result, return_response=True)
    texts = ocr_agent.gather_text_annotations(res)

    inference_words = []
    for words_bbox in texts:
        inference_words.append(words_bbox.text)

    inference_boxes = []
    for words_bbox in texts:
        h = np.min(words_bbox.block.points, axis=0)
        w = np.max(words_bbox.block.points, axis=0)
        inference_boxes.append([h[0], h[1], w[0], w[1]])

    receipt_json = {}
    receipt_json['file_name'] = filename
    receipt_json['size'] = image_result.shape
    receipt_json['bboxes'] = [normalize_bbox(box,
image_result.shape) for box in inference_boxes]
    receipt_json['words'] = inference_words

    receipt_list[filename] = receipt_json

    with open(f"Annotation/{filename.split('.')[0]}.json", "w")
as outfile:
    json.dump(receipt_list, outfile)

```

**Kode Program 4.8** Kode Program Otomasi Pembacaan OCR pada Setiap Nota

Kode program 4.8 adalah kode program yang digunakan untuk melakukan pembacaan OCR pada seluruh nota yang telah disegmentasi. Kode program ini akan menghasilkan sebuah *file* dengan format *.json* yang menyimpan hasil deteksi pada setiap nota.

#### 4.1.3 Annotasi Dataset

*Annotasi dataset* adalah proses yang dilakukan untuk memberikan label pada data yang dibuat. Setiap kata yang ada dalam nota yang telah dideteksi, akan diberikan label secara manual dengan pemberian angka sesuai pada tabel 4.1.

**Tabel 4.1** *Pelabelan Dataset*

<b>Nama Label</b>	<b>Kode Label</b>
<i>Ignore</i>	0
<i>Store_name_value</i>	1
<i>Date_value</i>	2
<i>Time_value</i>	3
<i>Prod_item_key</i>	4
<i>Prod_item_value</i>	5
<i>Prod_quantity_key</i>	6
<i>Prod_quantity_value</i>	7
<i>Prod_price_key</i>	8
<i>Prod_price_value</i>	9
<i>Subtotal_key</i>	10
<i>Subtotal_value</i>	11
<i>Total_key</i>	12
<i>Total_value</i>	13
<i>Others</i>	14

Tabel 4.1 adalah semua label yang digunakan dalam pembuatan *dataset*. Terdapat 15 label yang digunakan pada pembuatan *dataset*. Label-label yang digunakan pada hasil adalah *Store\_name\_value*, *Date\_value*, *Time\_value*, *Prod\_item\_value*, *Prod\_quantity\_value*, *Prod\_price\_value*, *Subtotal\_value*, dan *Total\_value*, dengan label lainnya digunakan sebagai label pembantu pemahaman Model.

```
def annot_helper(file_name,bboxes,words,batch_num):

    file_name = file_name.split('.json')[0]+' .jpg'
    img = cv2.imread(os.path.join(DIR, 'Nota_Segmented',file_name))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    ex_box = []

    for bbox in bboxes:
        bbox = unnormalize_bbox(bbox, img.shape)
        ex_box.append(bbox)

    ex_word = words[batch_num*10:(batch_num+1)*10]
    ex_box = ex_box[batch_num*10:(batch_num+1)*10]

    for box in ex_box:
        cv2.rectangle(img, (box[0], box[1]), (box[2], box[3]),
(255, 0, 0), 1)

    plt.figure(figsize=(10,10))
    plt.imshow(img)

    plt.figure(figsize=(20,10))

    for i in range(len(ex_word)):
```

```

plt.subplot(2,5,i+1)
plt.imshow(img[ex_box[i][1]:ex_box[i][3],
ex_box[i][0]:ex_box[i][2]])

plt.title(ex_word[i])

plt.show()
file_name = '20221013_192759.json'

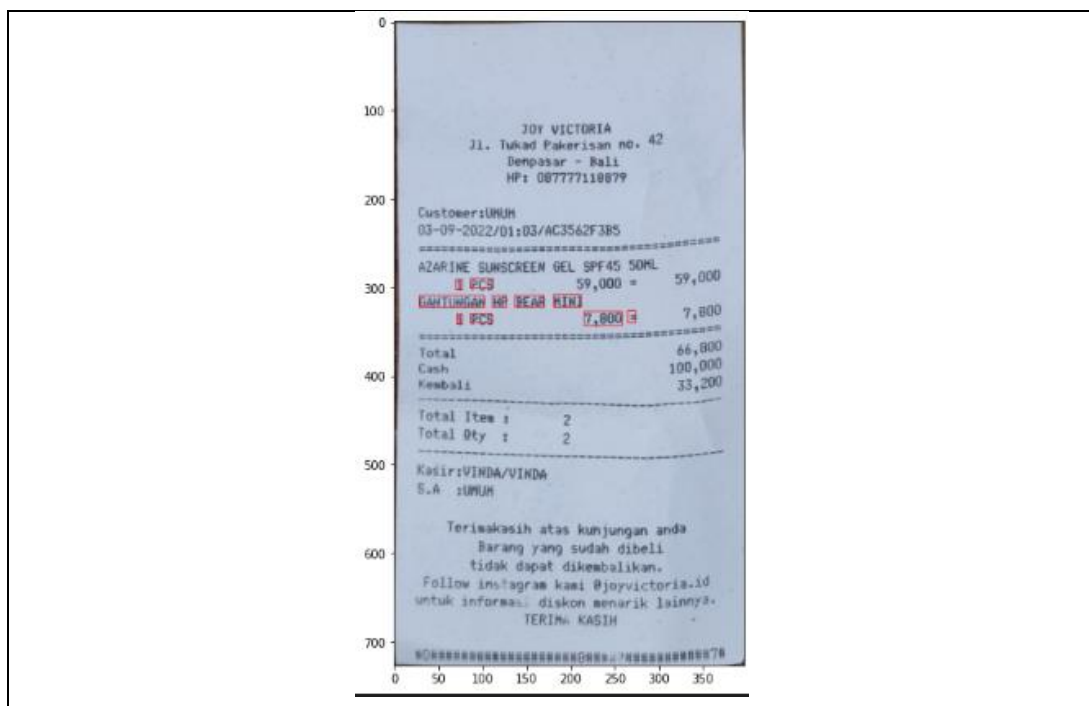
with open(os.path.join(DIR, 'Annotation', file_name)) as f:
    data = json.load(f)

annot_helper(file_name, data['bboxes'], data['words'], batch_num=4)

```

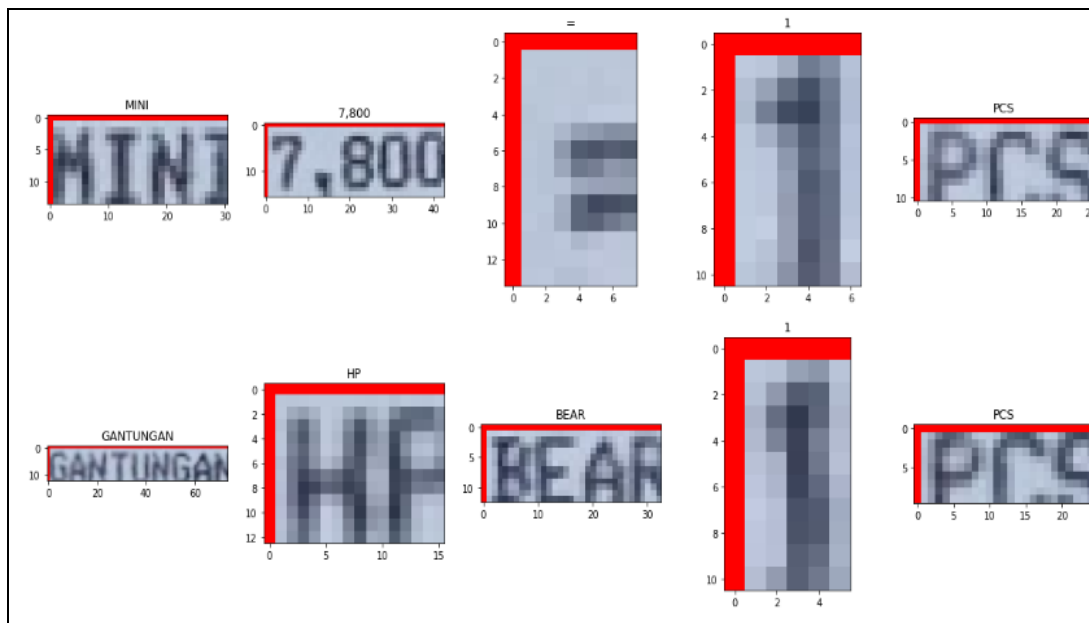
**Kode Program 4.9 Fungsi Pembantu Annotasi Dataset**

Kode program 4.9 adalah kode program yang digunakan untuk membantu proses *annotasi dataset*. Setiap *file* nota dengan meng-input-kan nama *file* pada variabel `file_name` dan memberikan `batch_num` dimulai dari nol. Tujuan penggunaan *batch* adalah agar dapat dipastikan letak dari kata tersebut dan label yang tepat. Jumlah *batch* pada suatu nota akan menyesuaikan dengan jumlah kata pada nota dan terdapat 10 kata pada setiap *batch*.



**Gambar 4.8** Gambar Penggunaan Fungsi Annotasi tahap 1

Gambar 4.8 merupakan gambar hasil penggunaan fungsi *annotasi* bantuan. Terdapat kotak merah pada setiap kata yang perlu dideteksi dan untuk setiap katanya dapat terlihat dengan jelas pada gambar 4.9.



**Gambar 4.9** Gambar Penggunaan Fungsi Annotasi tahap 2

Gambar 4.9 adalah gambar penggunaan fungsi *annotasi* bantuan yang berfokus pada setiap kata. Pelabelan akan dilakukan dengan mengikuti urutan dari kiri atas hingga kanan bawah dengan label yang sesuai.

## 4.2 Finetuning Model LayoutLM

Proses *Finetuning* dilakukan dengan menggunakan data latih sebanyak 80% hasil pembuatan data yang ditambahkan dengan 1267 data latih dari *dataset* sekunder. Proses pengujian dilakukan dengan menggunakan 20% dari data yang dibuat serta 472 *dataset* sekunder.

### 4.2.1 Data Preparation

*Data preparation* adalah tahap persiapan data di mana data dimuat ke dalam program. Setelah data dimuat ke dalam program, data akan diproses untuk mendapatkan data latih dan data uji.

```
dataset = load_dataset("Theivaprakasham/wildreceipt")

example = dataset["train"][0]

example["image_path"]

words, bboxes, ner_tags = example["words"], example["bboxes"],
example["ner_tags"]

print(words)
print(bboxes)
print(ner_tags)
```

**Kode Program 4.10** *Persiapan Dataset*

Kode program 4.10 adalah kode program yang digunakan untuk memuat data sekunder dari Theivaprakasham. *Dataset* akan dimuat dengan menggunakan fungsi `load_dataset` dan akan diambil kata-kata, *bouding box*, serta label pada *dataset*.

```
def normalize_bbox(bbox, size):
    return [
        int(bbox[0] * 1000 / size[1]),
        int(bbox[1] * 1000 / size[0]),
        int(bbox[2] * 1000 / size[1]),
        int(bbox[3] * 1000 / size[0]),
    ]

def unnormalize_bbox(bbox, size):
    return [
        int(bbox[0] * size[1] / 1000),
        int(bbox[1] * size[0] / 1000),
        int(bbox[2] * size[1] / 1000),
        int(bbox[3] * size[0] / 1000),
    ]
```

**Kode Program 4.11** *Fungsi Normalisasi Dataset*

Kode program 4.11 adalah kode yang digunakan untuk melakukan normalisasi pada *bouding box* kata dalam *dataset*. Normalisasi dilakukan agar data dapat diproses menggunakan *AutoProcessor* milik Model LayoutLM.

```
img_read = cv2.imread(example["image_path"])
img_read = cv2.cvtColor(img_read, cv2.COLOR_BGR2RGB)

if img_read is None:
    raise Exception(f"Image {example['image_path']} not found")

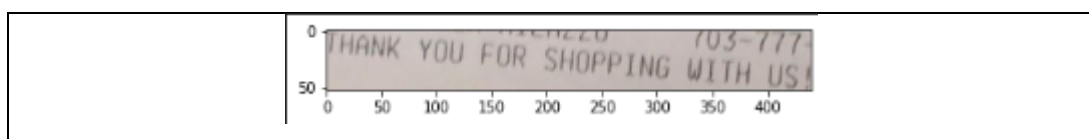
xx = [360, 191, 635, 235]
xx = unnormalize_bbox(xx, img_read.shape)

plt.imshow(img_read[xx[1]:xx[3],xx[0]:xx[2]])
```

```
plt.show()
print(img_read.shape)
xx
```

**Kode Program 4.12** *Segmentasi Kata Bounding Box*

Kode program 4.12 adalah kode program yang digunakan untuk melakukan segmentasi kata pada *bounding box*. Kode ini akan menampilkan contoh kata yang terdapat pada nota dengan segmentasi setelah diterapkan normalisasi *bounding box*.



**Gambar 4.10** *Hasil Segmentasi Bounding Box*

Gambar 4.10 adalah gambar contoh hasil segmentasi *bounding box*. Segmentasi yang tepat pada gambar menunjukkan normalisasi telah berhasil diterapkan pada nota dan siap untuk diproses menggunakan *AutoProcessor* milik LayoutLM.

#### 4.2.2 Data Pipelining

*Data pipelining* merupakan proses yang digunakan untuk memproses data lebih lanjut untuk dapat dilatih pada Model. Proses ini memanfaatkan *AutoProcessor* milik LayoutLM untuk menyesuaikan format *dataset* serta mengubahnya menjadi *encoding* untuk dilatih pada Model.

```
processor = AutoProcessor.from_pretrained("microsoft/layoutlmv3-  
base", apply_ocr=False)
```

**Kode Program 4.13** *Inisiasi AutoProcessor LayoutLM*

Kode program 4.13 adalah kode program yang digunakan untuk mendefinisikan *AutoProcessor* milik LayoutLM. *AutoProcessor* ini adalah sebuah *transformer* yang memiliki *input* dan *output* yang sesuai dengan format pelatihan LayoutLM.

```

features = dataset["train"].features
column_names = dataset["train"].column_names
image_column_name = "image_path"

# In the event the labels are not a `Sequence[ClassLabel]`, we
will need to go through the dataset to get the

# unique labels.

def get_label_list(labels):
    unique_labels = set()
    for label in labels:
        unique_labels = unique_labels | set(label)
    label_list = list(unique_labels)
    label_list.sort()
    return label_list

if isinstance(features["ner_tags"].feature, ClassLabel):
    label_list = features["ner_tags"].feature.names
    # No need to convert the labels since they are already ints.
    id2label = {k: v for k,v in enumerate(label_list)}
    label2id = {v: k for k,v in enumerate(label_list)}
else:
    label_list = get_label_list(dataset["train"]["ner_tags"])
    id2label = {k: v for k,v in enumerate(label_list)}
    label2id = {v: k for k,v in enumerate(label_list)}

num_labels = len(label_list)

```

**Kode Program 4.14** *Pengolahan Dataset*

Kode Program 4.14 adalah kode program yang digunakan untuk mengambil informasi yang terdapat pada *dataset*. Kode ini akan menghasilkan sebuah *list* dengan kumpulan kata, *bounding box*, label, serta letak *file* gambar.



```

dataset["train"][0]['words'] #words
dataset["train"][0]['bboxes'] #boxes
dataset["train"][0]['ner_tags'] #word_labels
dataset["train"].column_names

✓ 0.1s

['id', 'words', 'bboxes', 'ner_tags', 'image_path']

```

**Gambar 4.11** Hasil Pengolahan Dataset

Gambar 4.11 adalah gambar hasil pengolahan *dataset*. Pada gambar terlihat bahwa variabel *dataset*, telah memuat informasi-informasi yang dibutuhkan untuk melakukan pelatihan pada Model LayoutLM.

```

def prepare_examples(examples):
    images = [Image.open(path).convert("RGB") for path in
examples['image_path']] #Image.open(examples[image_column_name])

    words = examples["words"]
    boxes = examples["bboxes"]
    word_labels = examples["ner_tags"]

    encoding = processor(images, words, boxes=boxes,
word_labels=word_labels, truncation=True, padding="max_length")

    return encoding

features = Features({
    'pixel_values': Array3D(dtype="float32", shape=(3, 224, 224)),
    'input_ids': Sequence(feature=Value(dtype='int64')),
    'attention_mask': Sequence(Value(dtype='int64')),
    'bbox': Array2D(dtype="int64", shape=(512, 4)),
    'labels': Sequence(ClassLabel(names=label_list)),
})

train_dataset = dataset["train"].map(
    prepare_examples,
    batched=True,
    remove_columns=column_names,
    features=features,
)

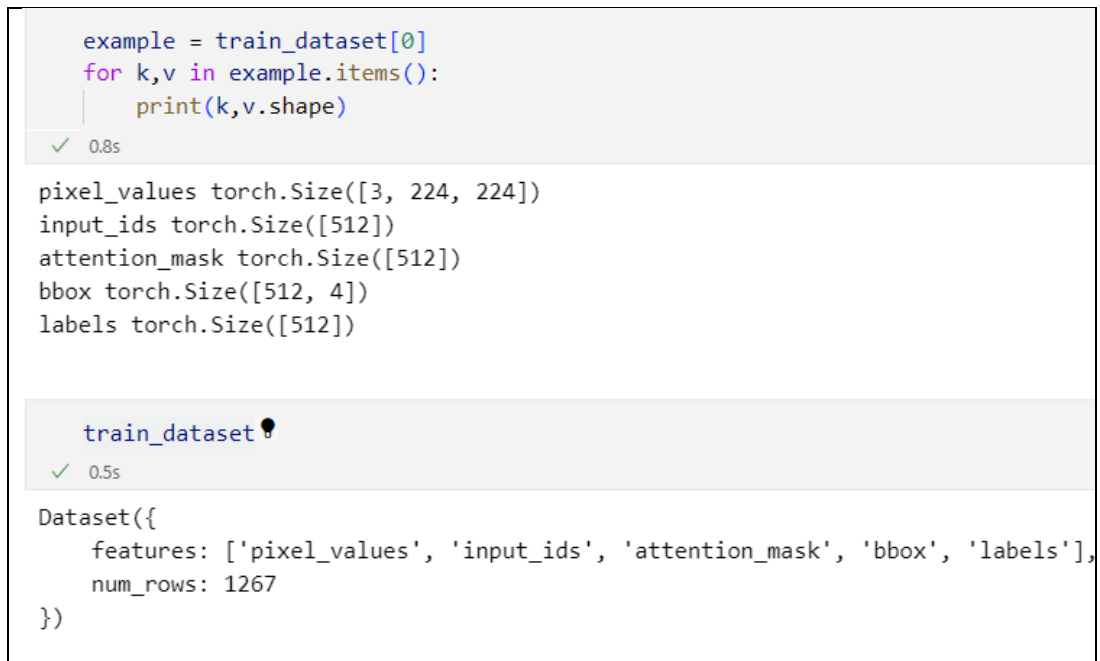
```

```
eval_dataset = dataset["test"].map(
    prepare_examples,
    batched=True,
    remove_columns=column_names,
    features=features,
)

train_dataset.set_format("torch")
```

**Kode Program 4.15** *Pengubahan Data Menjadi Encoding*

Kode program 4.15 adalah kode program yang digunakan untuk mengubah data yang telah dipisahkan menjadi bentuk *encoding* untuk dilanjutkan ke tahap pelatihan Model. Hasil dari *encoding* yang dilakukan adalah dalam format *torch* dengan jumlah data latih sebanyak 1267 dan data uji berjumlah 472.



```
example = train_dataset[0]
for k,v in example.items():
    print(k,v.shape)
```

✓ 0.8s

```
pixel_values torch.Size([3, 224, 224])
input_ids torch.Size([512])
attention_mask torch.Size([512])
bbox torch.Size([512, 4])
labels torch.Size([512])
```

```
train_dataset
```

✓ 0.5s

```
Dataset({
  features: ['pixel_values', 'input_ids', 'attention_mask', 'bbox', 'labels'],
  num_rows: 1267
})
```

**Gambar 4.12** *Hasil Pengubahan Data Menjadi Encoding*

Gambar 4.12 adalah gambar hasil pengubahan data menjadi *encoding*. Pada Gambar terdapat contoh data setelah melalui proses *AutoProcessing* milik LayoutLM. Hasil dari *dataset* memiliki fitur berupa *pixel\_values*, *input\_ids*, *attention\_mask*, *bbox*, dan *labels*.

### 4.2.3 Model Finetuning

Model *Finetuning* adalah tahap pelatihan Model dengan menggunakan data yang telah dipersiapkan. Model ini akan dilatih dengan menggunakan data latih dan disimpan untuk membaca nota yang ada pada data uji.

```
metric = load_metric("segeval")

return_entity_level_metrics = False

def compute_metrics(p):
    predictions, labels = p
    predictions = np.argmax(predictions, axis=2)

    # Remove ignored index (special tokens)
    true_predictions = [
        [label_list[p] for (p, l) in zip(prediction, label) if l
!= -100]
        for prediction, label in zip(predictions, labels)
    ]

    true_labels = [
        [label_list[l] for (p, l) in zip(prediction, label) if l
!= -100]
        for prediction, label in zip(predictions, labels)
    ]

    results = metric.compute(predictions=true_predictions,
references=true_labels)

    if return_entity_level_metrics:
        # Unpack nested dictionaries
        final_results = {}

        for key, value in results.items():
            if isinstance(value, dict):
                for n, v in value.items():
                    final_results[f"{key}_{n}"] = v
            else:
                final_results[key] = value

        return final_results
    else:
```

```

    return {

        "precision": results["overall_precision"],

        "recall": results["overall_recall"],

        "f1": results["overall_f1"],
        "accuracy": results["overall_accuracy"],

    }

```

**Kode Program 4.16** *Metrics Pelatihan Model*

Kode program 4.16 adalah kode program yang digunakan untuk melakukan evaluasi dari pelatihan Model. Evaluasi Model selama pelatihan dilakukan dengan menggunakan fungsi `load_metric` dari *library Datasets* dengan penilaian presisi, *recall*, skor f1, dan akurasi.

```

Model=LayoutLMv3ForTokenClassification.from_pretrained("microsoft/
layoutlmv3-base",id2label=id2label,label2id=label2id)

```

**Kode Program 4.17** *Memuat Model LayoutLM*

Kode Program 4.17 adalah kode program yang digunakan untuk memuat Model LayoutLM dasar. Kode ini akan memuat Model LayoutLM dasar milik Microsoft dengan parameter variabel *dictionary* yang berisikan daftar nama label beserta nomor dari nama label tersebut.

```

training_args = TrainingArguments(output_dir="layoutlmv3-
finetuned-wildreceipt",
max_steps=4000,
per_device_train_batch_size=4,
per_device_eval_batch_size=4,
learning_rate=1e-5,
evaluation_strategy="steps",
eval_steps=100,
load_best_model_at_end=True,
metric_for_best_model="f1",
)
trainer = Trainer(
    Model=Model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    tokenizer=processor,
    data_collator=default_data_collator,
    compute_metrics=compute_metrics,
)
trainer.train()

```

**Kode Program 4.18** *Pelatihan Model*

Kode program 4.18 merupakan kode program yang digunakan untuk melatih Model. Model akan dilatih dengan melihat skor f1 tertinggi serta Model terbaik yang ditemukan akan dimuat secara otomatis pada akhir sesi pelatihan.

#### 4.2.4 Model Evaluation

Model *evaluation* berisi tentang hasil evaluasi pada Model yang telah dilatih sebelumnya. Metode yang digunakan dalam penilaian Model menggunakan *confussion matrix*, serta *classification report* sesuai dengan yang telah dijelaskan pada bab III.

```
trainer.evaluate()
```

#### Kode Program 4.19 Evaluasi Model

Kode program 4.19 adalah kode program yang digunakan untuk melakukan evaluasi pada hasil pelatihan. Model akan dievaluasi dengan menggunakan data *testing* dengan menggunakan metrik yang sama pada saat pelatihan dan menampilkan akurasi pada data *testing*.

### 4.3 Deployment Model LayoutLM

*Deployment* Model adalah pelepasan Model untuk digunakan dalam tahap produksi. Pada tahap ini dilakukan pembuatan sebuah *web server* dengan menggunakan *framework Flask* serta sebuah aplikasi Android untuk mengirimkan gambar nota pada Model.

#### 4.3.1 Flask API Server

*Flask API server* adalah *framework* yang digunakan untuk men-*deploy* Model dalam bentuk sebuah API. *Flask server* akan menerima gambar nota yang telah disegmentasi dari *smartphone* Android dan melakukan *inferensi* Model pada gambar nota serta mengirimkan hasil deteksi kembali kepada perangkat Android.

```
@app.route('/')
def index():

    process_image(os.path.join(DIR, 'Nota_Segmented/20221123_230408.jpg
    '), Model, processor)
    return "<p>Hello, World!</p>"
```

```

@app.route('/detect', methods=['POST'])
def detect(request):
    data = json.loads(request.data)
    bytearrayimg = decodeB64(data)
    bytearrayimg = bytearray(bytearrayimg)
    pil_image = Image.open(io.BytesIO(bytearrayimg))
    cv_img = np.array(pil_image)
    # Convert RGB to BGR
    cv_img = cv_img[:, :, ::-1].copy()
    process_image(cv_img, Model, processor)
    return "<p>detect</p>"

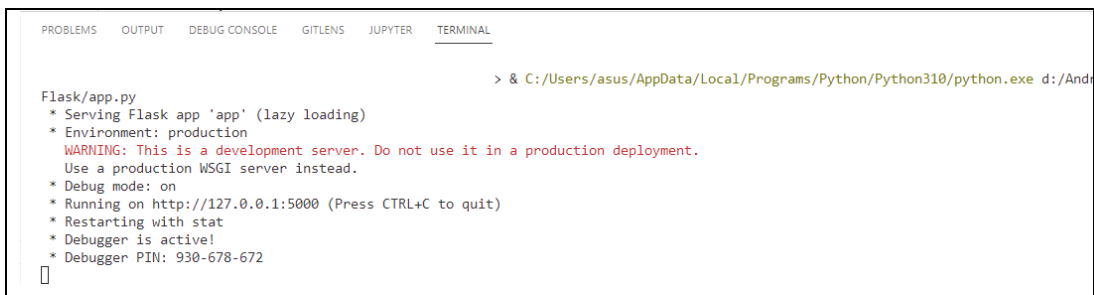
def decodeB64(data):
    image_64_decode = base64.b64decode(data)
    return image_64_decode

if __name__ == '__main__':
    app.run(debug = True)

```

**Kode Program 4.20** *Web Server Deployment*

Kode program 4.20 adalah kode program yang digunakan untuk *deploy* Model dalam bentuk *web server* dengan menggunakan *framework Flask*. Gambar terenkripsi yang diterima dari android akan dimuat dengan menggunakan *library Json* yang akan di-*decode* dengan menggunakan *Base64* hingga menjadi format gambar milik *Opencv* dan diproses dengan menggunakan fungsi `process_image`.



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  GITLENS  JUPYTER  TERMINAL
> & C:/Users/asus/AppData/Local/Programs/Python/Python310/python.exe d:/And

Flask/app.py
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 930-678-672

```

**Gambar 4.13** *Hasil Web Server Deployment*

Gambar 4.13 adalah gambar hasil Model *deployment* dengan menggunakan *web server*. Halaman *web server* yang dibuat hanya dapat diakses secara lokal. Aplikasi Android akan memanggil *server* ini untuk menjalankan proses inferensi Model.

## **4.3.2 Android APK**

### **4.3.2.1 Pembacaan Nota**

### **4.3.2.2 Review Pembacaan Nota**

### **4.3.2.3 Penyimpanan Hasil Pembacaan Pada SQLite**

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

#### **5.2 Saran**



## DAFTAR PUSTAKA

- Andreas, Y., Gunadi, K., & Purbowo, A. N. (2020). Implementasi Tesseract OCR untuk Pembuatan Aplikasi Pengenalan Nota pada Android. *Jurnal Infra*, 8(1), 2–7.
- Darma, I. W. A. S. (2019). Implementation of Zoning and K-Nearest Neighbor in Character Recognition of Wrésastra Script. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, 9. <https://doi.org/10.24843/lkjiti.2019.v10.i01.p02>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. <http://arxiv.org/abs/1810.04805>
- Hajiali, M., Fonseca Cacho, J. R., & Taghva, K. (2022). Generating Correction Candidates for OCR Errors using BERT Language Model and FastText SubWord Embeddings. *Lecture Notes in Networks and Systems*, 283, 1045–1053. [https://doi.org/10.1007/978-3-030-80119-9\\_69](https://doi.org/10.1007/978-3-030-80119-9_69)
- Holeček, M. (2020). *Learning from similarity and information extraction from structured documents*. <https://doi.org/10.1007/s10032-021-00375-3>
- Howard, J., & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text Classification*. <http://arxiv.org/abs/1801.06146>
- Indrawan, G., Asroni, A., Joni Erawati Dewi, L., Gunadi, I. G. A., & Paramarta, I. K. (2022). Balinese Script Recognition Using Tesseract Mobile Framework. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, 13(3), 160. <https://doi.org/10.24843/LKJITI.2022.v13.i03.p03>
- Kumar, V., Kaware, P., & Singh, P. (2020). *Extraction of information from bill receipts using optical character recognition*.
- Kumar, V., Kaware, P., Singh, P., Sonkusare, R., & Kumar, S. (2020). Extraction of information from bill receipts using optical character recognition. *Proceedings - International Conference on Smart Electronics and Communication, ICOSEC 2020, Icosec*, 72–77. <https://doi.org/10.1109/ICOSEC49089.2020.9215246>

- Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4), 1234–1240. <https://doi.org/10.1093/bioinformatics/btz682>
- Lin, C. J., Liu, Y. C., & Lee, C. L. (2022). Automatic Receipt Recognition System Based on Artificial Intelligence Technology. *Applied Sciences (Switzerland)*, 12(2). <https://doi.org/10.3390/app12020853>
- Liu, Y., James, H., Gupta, O., & Raviv, D. (2022). MRZ code extraction from visa and passport documents using convolutional neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 25(1), 29–39. <https://doi.org/10.1007/s10032-021-00384-2>
- Markewich, L., Zhang, H., Xing, Y., Lambert-Shirzad, N., Jiang, Z., Lee, R. K.-W., Li, Z., & Ko, S.-B. (2022). Segmentation for document layout analysis: not dead yet. *International Journal on Document Analysis and Recognition (IJDAR)*, 25(2), 67–77. <https://doi.org/10.1007/s10032-021-00391-3>
- Memon, J., Sami, M., & Khan, R. A. (2019). *Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)*. <http://arxiv.org/abs/2001.00139>
- Nguyen, D.-D. (2022). TableSegNet: a fully convolutional network for table detection and segmentation in document images. *International Journal on Document Analysis and Recognition (IJDAR)*, 25(1), 1–14. <https://doi.org/10.1007/s10032-021-00390-4>
- Puspitarani, Y., & Syukriyah, Y. (2017). *Pemanfaatan Optical Character Recognition Dan Text Feature Extraction Untuk Membangun Basisdata Pengaduan Tenaga Kerja*. 1(3), 704–710.
- Ramsay, B., Ralescu, A., van der Knaap, E., & Visa, S. (2011). *Confusion Matrix-based Feature Selection*. *Confusion Matrix-based Feature Selection*. <https://www.researchgate.net/publication/220833270>
- Raoui-Outach, R., Million-Rousseau, C., Benoit, A., & Lambert, P. (2018). Deep learning for automatic sale receipt understanding. *Proceedings of the 7th International Conference on Image Processing Theory, Tools and Applications, IPTA 2017, 2018-Janua*, 1–6. <https://doi.org/10.1109/IPTA.2017.8310088>
- Sastrawan, I. K., Bayupati, I. P. A., & Arsa, D. M. S. (2022). Detection of fake news using deep learning CNN–RNN based methods. *ICT Express*, 8(3), 396–408. <https://doi.org/10.1016/j.ict.2021.10.003>

- Sudana, O., Gunaya, I. W., & Putra, I. K. G. D. (2020). Handwriting identification using deep convolutional neural network method. *Telkomnika (Telecommunication Computing Electronics and Control)*, 18(4), 1934–1941. <https://doi.org/10.12928/TELKOMNIKA.V18I4.14864>
- Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1192–1200. <https://doi.org/10.1145/3394486.3403172>
- Google Cloud. (2022). *Vision AI*. Diambil kembali dari Cloud Vision API: dikutip tanggal 11 Juni 2022, <<https://cloud.google.com/vision>>.
- Theivaprakasham. (2022, Juni 11). *Theivaprakasham/wildreceipt*. Diambil kembali dari Hugging Face: dikutip tanggal 11 Juni 2022, <<https://huggingface.co/datasets/Theivaprakasham/wildreceipt>>.