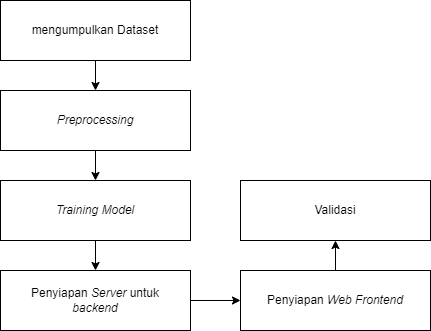
# BAB IV

# IMPLEMENTASI DAN PEMBAHASAN

## 4.1. IMPLEMENTASI

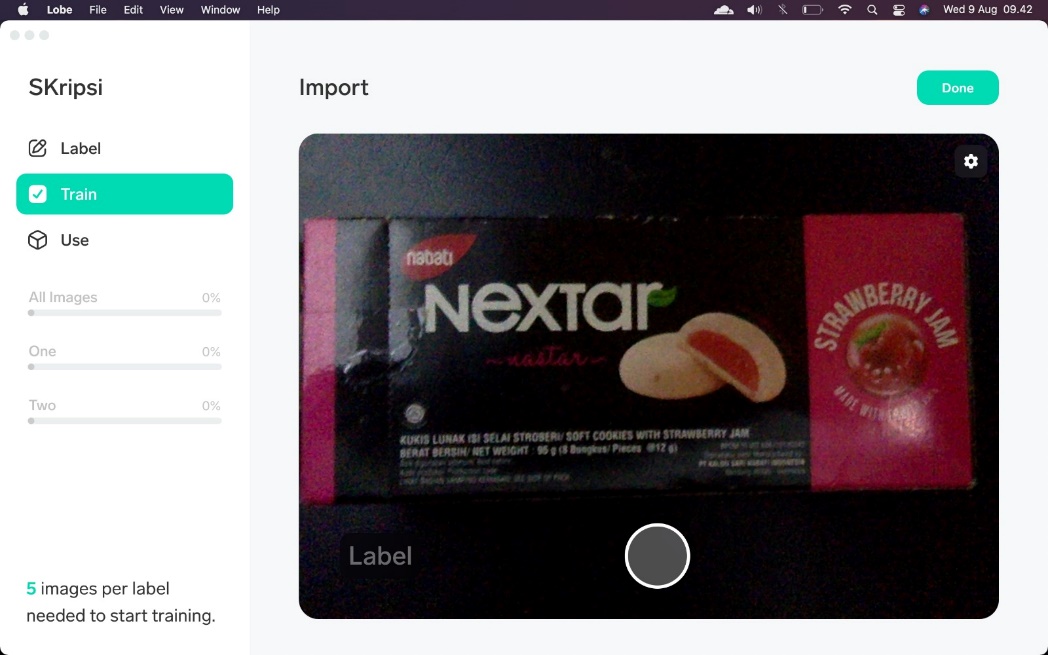
Mengangkat tentang implementasi *deep learning* dalam mengatasi permasalahan kemasan rusak pada kemasan Nextar. Dalam bab ini, akan dijelaskan bagaimana penerapan teknik-teknik *deep learning* untuk mendeteksi dan mendiagnosis kerusakan pada kemasan. Pada penelitian ini menggunakan aplikasi Lobe untuk membantu dalam melakukan pengumpulan dataset , preprocessing, dan training model. Lobe adalah aplikasi buatan tim Microsoft yang ditujukan membantu orang untuk menggunakan *model machine learning* dengan menggunakan desain *interface* yang mudah digunakan. Berikut gambar 4.1 alur dalam pengimplementasian penelitian ini.



Gambar 4.1 *Flowchart* pada Bab 4

**4.1.1 Pengumpulan Dataset**

Dataset gambar kemasan Nextar yang bagus dan rusak dengan rasa stroberi dikumpulkan dan dibuat oleh peneliti dengan bantuan aplikasi Lobe. Hanya gambar yang memenuhi kriteria yang diambil dari hasil pencarian. Sampel data gambar yang digunakan ditunjukkan pada Gambar 3.1 dan Gambar 3.2. Kriteria gambar yang digunakan adalah gambar yang hanya mencakup tampilan depan produk dan berukuran lebih dari 150 piksel. Berikut tampilan pengumpulan dataset menggunakan aplikasi Lobe.



**4.1.2 *Preprocessing***

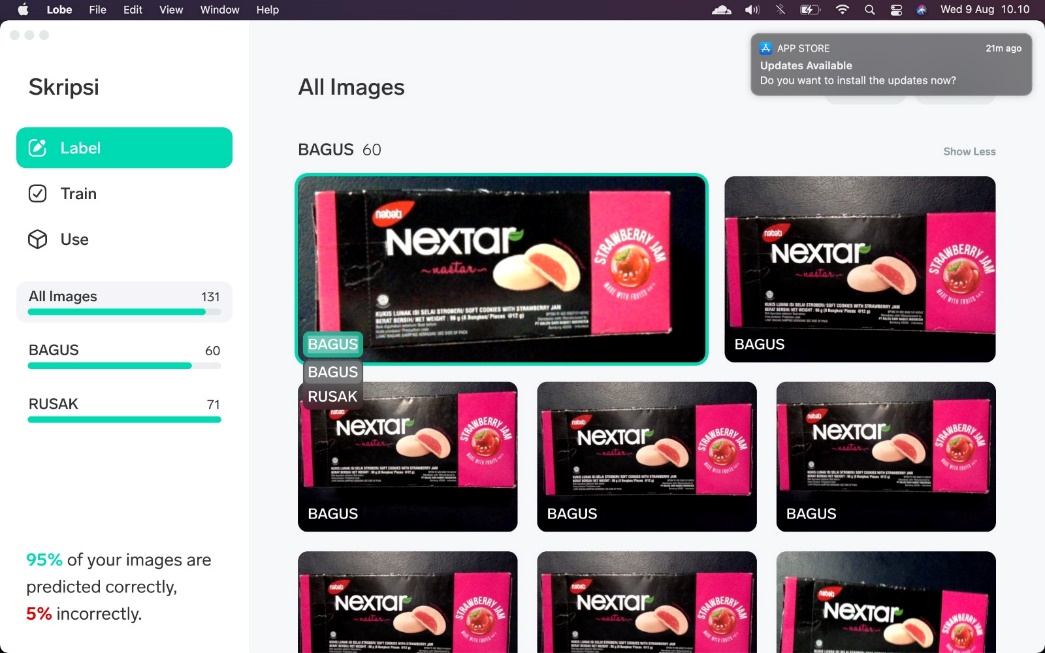
Sebelum memulai proses pelatihan model, preprocessing gambar dilakukan untuk mempersiapkan dataset. pengukuran gambar, dan labeling gambar adalah langkah-langkah yang dilakukan secara bertahap.

**- Kompresi Gambar**

Standarisasi ukuran diperlukan karena ukuran citra input bervariasi. Karena kompleksitas gambar yang terlalu besar, ukuran gambar yang terlalu besar dapat mengganggu proses komputasi, sedangkan ukuran gambar yang terlalu kecil dapat mempercepat proses, tetapi dapat menyebabkan informasi dari gambar hilang. Akibatnya, pengukuran gambar disesuaikan dengan ukuran objek yang ingin dideteksi. Setelah gambar yang dihasilkan dari segmentasi kemasan dikompresimenjadi gambar dengan ukuran yang lebih kecil. Gambar dengan ukuran ini bukan gambar yang besar, jadi objek kemasan masih dapat dilihat secara visual.

**- Labeling Gambar**

Dengan menggunakan *software Lobe,* anotasi gambar atau label gambar dilakukan untuk mengidentifikasi objek yang ingin dideteksi pada gambar. Ini dilakukan dengan menambahkan teks atau label ke setiap bagian gambar data yang telah dikumpulkan. Label "Bagus" dan "Rusak" digunakan di sini.



* + 1. ***Training Model***

Model diberikan data pelatihan, dan *processing* dilakukan untuk mengubah bobot dan parameter model agar lebih sesuai dengan data yang akan dilakukan dalam pelatihan. Proses ini melibatkan perhitungan *loss* atau *error* antara output model dan label yang benar, dan kemudian melakukan penyesuaian bobot agar *error* ini diminimalkan. Berikut tabel hasil training yang dilakukan dengan bantuan aplikasi Lobe.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Data | Waktu Pelatihan | *Correct* | *Loss / error* |
| 30 | ~ 3 menit | 90 % | 10 % |
| 80 | ~ 4 menit | 89 % | 11 % |
| 130 | ~ 5 menit | 95 % | 5 % |
| 200 | ~ 5 menit | 100% | 0% |

Tabel 4.1. Hasil Percobaan dengan jumlah dataset yang berbeda

* + 1. **Penyiapan *Server Backend***

Setelah melakukan proses training pada aplikasi Lobe, pada proses selanjutnya adalah mengekspor hasil *model training* untuk dikonsumsi *library* *Tensorflow* pada kode *Python* yang digunakan untuk melakukan validasi gambar inputan dari pengguna. Kode Python ini berfungsi untuk memberikan respon data persentase dari kemiripan gambar yang dikirim pengguna melalui *web frontend* menggunakan *Restful Api.* Dalam proses *backend web* yang akan digunakan adalah menggunakan Google Collab. Google Collab dipilih menjadi opsi karena fitur spesifikasi *server* yang mampu untuk melakukan komputasi *image processing* karena membutuhkan *Graphic Processing Unit* dengan performa yang bagus. Selain itu digunakannya Google Collab ini karena ada fitur penyimpanan untuk menyimpan data *model training* yang akan digunakan, serta karena Google Collab gratis untuk digunakan. Berikut tahapan dalam penyiapan *server backend* menggunakaan Google Collab.

1. **Mengunggah Dataset ke Google Drive**

Google Colab harus memiliki akses ke program dan dataset yang diperlukan. Selanjutnya, berkas diunggah ke Google Drive, yang dapat terhubung secara langsung dengan Google Colab. Ini dilakukan dengan mengunggah berkas ke folder yang sudah ditentukan untuk pelatihan model, atau workspace. Ini memudahkan proses mounting Google Drive ke Google Colab, sehingga tidak perlu mengubah alamat folder setiap kali diperlukan.

1. **Menghubungkan Google Colab dan Google Drive**

Untuk menghubungkan Google Collab dan Google Drive, pengguna harus memberikan hak akses untuk kedua layanan tersebut. Ini diperlukan saat proses mounting folder ke Google Colabs. Google akan meminta kode token otorisasi dari pengguna untuk memberikan izin untuk mengakses alamat Google Drive. Jika pengguna memberikan izin, Google akan memberikan kode hak akses untuk membiarkan Google Colabs mengakses berkas-berkas yang didaur ulang.

1. **Instalasi *Library Python***

Lembar kerja Google Collab menggunakan bahasa pemrograman *Python*. Untuk menghindari *error* program, sebelum memulai proses pelatihan model, pastikan bahwa *library* yang digunakan sudah terinstal. Daftar *library* *Python* yang digunakan ditunjukkan pada Tabel 4.2 berikut.

Sebuah gambar berisi teks, cuplikan layar, tampilan, Font

Deskripsi dibuat secara otomatis

Gambar 4.1. Memasang Pustaka yang digunakan

|  |  |
| --- | --- |
| *Flask* | Berguna untuk menghandel *request* API dari halaman utama *website* |
| *Pillow* | Digunakan untuk memanipulasi gambar |
| *Tensorflow* | *Library* untuk pelatihan model *deep learning* |
| *Flask\_cors* | *Library* yang diperlukan supaya akses data dari luar *domain* yang dipakai dapat dijalankan tanpa terkena *blocking data* |
| *Flask-ngrok* | *Library* yang diperlukan untuk menjalankan layanan Ngrok supaya mendapat *public ip* agar bisa diakses diluar lingkungan lokal |
| *Pyngrok* | *Library* yang diperlukan untuk menjalankan layanan Ngrok supaya mendapat *public ip* agar bisa diakses diluar lingkungan lokal |

Tabel 4.2. Tabel fungsi dari *library* yang digunakan

1. **Menjalankan kode *python***

Kode *backend* yang berisi kode python untuk melakukan validasi gambar yang diupload lewat halaman *frontend.* Ada banyak opsi untuk menjalankan kode *backend* ini. Bisa lewat *Virtual Private Server* (VPS) untuk menjalankan, tetapi membutuhkan biaya yang cukup banyak untuk menyewa sebuah VPS. Jadi untuk alternatif agar bisa diakses secara publik bisa menggunakan Google Collabs.

A screen shot of a computer program

Description automatically generated with medium confidence

Gambar 4.2. Kode Pada Google Collab

Jalankan Kode diatas maka akan muncul URL publik yang bisa diakses. Setelah itu copas url tersebut untuk dimasukkan ke halaman utama w*ebsite frontend*.

* + 1. **Penyiapan *Web Frontend***

Halaman ini berguna untuk mengambil gambar Nextar dari pengguna untuk diupload ke *backend python* untuk dilakukan cek data apakah gambar tersebut merupakan gambar Nextar yang rusak atau bagus. Untuk menyiapkan website ini supaya bisa diakses oleh publik bisa menggunakan Vercel. Vercel adalah *website* yang menyediakan fasilitas untuk melakukan *build Framework Javascript* secara gratis supaya dapat diakses oleh publik. Setelah melakukan *build* dan mendapatkan url publik dari Vercel segera akses url tersebut. Setelah muncul tampilan *web*nya berikan akses video karena *website* ini memerlukan akses untuk mengambil video atau gambar untuk melakukan validasi gambar secara *realtime.* Dan masukkan url yang didapatkan dari *Backend* pada menu url. Untuk pengiriman gambar yang dilakukan pada *frontend* ke *backend* menggunakan gambar *base-64* dari hasil capture camera yang berformatkan JPEG

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 4.3. Halaman Utama *Website*

* + 1. **Validasi**

Proses ini dilakukan untuk mendapatkan label antara bagus dan rusak dari inputan gambar yang dilakukan pengguna menggunakan kamera video secara *realtime*. Untuk mengambil dan mengirim gambar bisa menggunakan tombol enter secara otomatis akan mengambil gambar dari video lalu dikirim ke *backend* untuk divalidasi dan mendapatkan label.

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 4.4. Hasil Validasi dari *website* apabila rusak

Sebuah gambar berisi teks, cuplikan layar, Situs, Laman internet

Deskripsi dibuat secara otomatis

Gambar 4.5. Hasil Validasi dari *website* apabila bagus

## Pembahasan

Pembahasan perhitungan dan analisis metrik evaluasi untuk mengevaluasi performa model yang telah dilatih dalam bentuk scenario menggunakan metrik evaluasi sebagai berikut.

## Metrik Evaluasi

Dalam evaluasi model, kami menggunakan beberapa metrik evaluasi yang umum digunakan dalam tugas klasifikasi, yaitu Akurasi, Presisi, *Recall*, dan *F1-Score*.

* + - 1. **Akurasi**

Akurasi mengukur sejauh mana model dapat mengklasifikasikan dengan benar seluruh data dalam dataset. Kami menghitung akurasi menggunakan rumus: Akurasi = (TP + TN) / (TP + TN + FP + FN).

TP = *True* Postif adalah label bagus dengan hasil yang benar

TN = *True* Negatif adalah label rusak dengan hasil yang benar

FP = *False* Postif adalah label bagus dengan hasil yang salah

FN = *False* Negatif adalah label rusak dengan hasil yang salah

* + - 1. **Presisi**

Presisi mengukur sejauh mana prediksi positif yang dibuat oleh model adalah benar. Kami menghitung presisi menggunakan rumus: Presisi = TP / (TP + FP)

* + - 1. ***Recall***

*Recall* mengukur sejauh mana model mampu mendeteksi dengan benar data positif. Kami menghitung recall menggunakan rumus: Recall = TP / (TP + FN)

* + - 1. ***F1-Score***

*F1-Score* merupakan nilai rata-rata harmonik antara presisi dan recall, memberikan ukuran keseluruhan performa model. Kami menghitung *F1-Score* menggunakan rumus: *F1-Score* = 2 \* (Presisi \* Recall) / (Presisi + *Recall*).

## Perhitungan

Berikut hasil perhitungan metrik evaluasi dari hasil dari training model yang dilakukan dengan beberapa jumlah data yang berbeda. Sebagai berikut:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Jumlah Data | TN | FN | TP | FP | Akurasi | Presisi | Recall | F1-Score |
| 30 | 13 | 2 | 13 | 2 | 0,933333 | 0,866667 | 0,866667 | 0,866667 |
| 80 | 30 | 3 | 40 | 7 | 0,9125 | 0,851064 | 0,930233 | 0,888889 |
| 134 | 69 | 2 | 60 | 1 | 0,992424 | 0,983607 | 0,967742 | 0,97561 |
| 215 | 69 | 2 | 144 | 1 | 0,99537 | 0,993103 | 0,986301 | 0,989691 |

Analisis hasil perhitungan metrik evaluasi menunjukkan bahwa model yang telah dilatih dengan menggunakan jumlah dataset yang lebih banyak mampu mengklasifikasikan kemasan Nextar yang rusak dengan tingkat akurasi, *presisi*, *recall* dan *f1-score* yang tinggi menunjukkan bahwa dari semua data yang diprediksi mampu melakukan validasi inputan data yang lebih efektif. *F1-Score* sebesar 98.96% menunjukkan keseimbangan antara *presisi* dan *recall* dalam memprediksi kemasan biskuit Nextar yang rusak. Selain itu dilakukan juga hasil uji coba pada aplikasi yang dilakukan selama beberapa kali dengan jumlah data yang berbeda dan dilakukan perhitungan metrik evaluasi terhadap hasil uji coba yang dilakukan sebagai berikut :

# BAB V

# KESIMPULAN

## Kesimpulan

Penelitian ini menggunakan teknik *Deep Learning* dengan metode *Tensorflow* dan CNN sebagai metode dalam pengembangan pendeteksian kemasan biskuit nextar yang rusak. Pada tahapannya peneliti melakukan pengumpulan pembuatan data, pengujian data, penerapan data, dan validasi metrik.

Dalam pengembangannya, aplikasi yang dikembangkan yaitu berbasis web. Dimana pengguna dapat mengakses aplikasi tersebut dengan menggunakan web browser pada *smartphone* maupun komputer. Sistem kerja dari aplikasi yang dikembangkan yaitu aplikasi melakukan pengambilan gambar dari kemasan produk, setelah itu gambar akan dikirim ke *backend* untuk mendapatkan label dari kemasan bagus atau kemasan rusak.

hasil analisis yang didapatkan dari perhitungan metrik evaluasi menunjukkan bahwa model yang telah dilatih mampu mengklasifikasikan kemasan Nextar yang rusak dengan tingkat akurasi yang tinggi, yaitu 95.83%. Presisi sebesar 93.33% menunjukkan bahwa dari semua data yang diprediksi sebagai rusak, 93.33% benar-benar rusak. Recall sebesar 98.25% menunjukkan bahwa model mampu mendeteksi 98.25% dari semua data yang sebenarnya rusak. F1-Score sebesar 95.60% menunjukkan keseimbangan antara presisi dan recall dalam memprediksi kemasan biskuit Nextar yang rusak.

## Saran

Adapun untuk saran dari hasil penelitian ini, yaitu:

1. Pengembangan dari berbasis web diubah kedalam bentuk aplikasi software android, ios, maupun windows.
2. Penambahan fitur riwayat.
3. Penambahan fitur yang menampilkan hasil data scan dalam beberapa dekade atau waktu tertentu.
4. Penerapan aplikasi ini disatukan dengan alat – alat yang ada di lingkungan pabrik sehingga sistem menjadi lebih efisien.
5. Pengintregasian aplikasi ini ke sistem informasi yang sudah berjalan di pabrik supaya dapat dianalisa data yang didapat