# Estimasi Kalori pada makanan melalui Citra Makanan menggunakan Model *SSD (Single Shot Detector)* pada *Mobile Device*


Disusun Oleh:

N a m a : Randy Efan Jayaputra

NIM : 15523177

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**2020**

**HALAMAN PENGESAHAN DOSEN PEMBIMBING**

# Estimasi Kalori pada makanan melalui *Citra Makanan* menggunakan Model *SSD (Single Shot Detector)* pada *Mobile Device*

## TUGAS AKHIR

Disusun Oleh:


N a m a : Randy Efan Jayaputra

NIM : 15523177


Yogyakarta, 26 Oktober 2020

| Pembimbing I | Pembimbing II |
|---|---|
| **(Dr. Ing. Ridho Rahmadi, M.Sc)** | **(Arrie Kurniawardhani, S.Si, M.Kom)** |

**HALAMAN PENGESAHAN DOSEN PENGUJI**

# Estimasi Kalori pada makanan melalui Citra Makanan menggunakan Model *SSD (Single Shot Detector)* pada *Mobile Device*

## TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Teknik Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 26 Oktober 2020

Tim Penguji

Dr. Ing. Ridho Rahmadi, M.Sc

**Anggota 1**

Ahmad Fathan Hidayatullah, S.T., M.CS.

**Anggota 2**

Taufiq Hidayat S.T., M.C.S.

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

( Dr. Raden Teduh Dirgahayu, S.T., M.Sc. )

## HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Randy Efan Jayaputra

NIM : 15523177

Tugas akhir dengan judul:

**Estimasi Kalori pada makanan melalui Citra Makanan menggunakan Model *SSD (Single Shot Detector)* pada *Mobile Device***

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 7 November 2020


(Randy Efan Jayaputra)

# HALAMAN PERSEMBAHAN

***Assalamu'alaikum Warrahmatullah Wabarakatuh***

Dengan mengucapkan syukur *Alhamdulillahirrabbil alamin*, saya persembahkan Tugas Akhir untuk orang-orang tercinta atas kasih, cinta dan perhatian terhadap saya.

**Kedua orang tua tersayang**

(Bapak Junaedi Juddin dan Ibu Dewi Ratnawati)

Saya persembahkan kepada kedua orang tua saya Tugas Akhir ini, yang dapat diselesaikan karena do'a, nasihat, motivasi serta arahan hingga saya berada pada tahap ini.

# HALAMAN MOTO

*"Maka nikmat Tuhan mana lagi yang kamu dustakan?"*
(Q.S. Ar-Rahman: 13)

"*it always seems impossible until it's done"*
(Nelson Mandela)

"*Be kind to one and antoher"*
(Ellen)

# KATA PENGANTAR

***Assalamu'alaikum Warrahmatullah Warabakatuh,***

*Alhamdulillahirabbil'alamin,* puji syukur atas kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga penyusunan laporan Tugas Akhir dengan judul "Estimasi Kalori pada makanan melalui *Image* menggunakan model *deep learning* pada *mobile devices*" dapat diselesaikan dengan baik dan lancar. Sholawat serta salam semoga selalu dilimpahkan kepada junjungan kita Nabi Muhammad SAW dalam memperjuangkan kaum muslimin menuju jalan kebenaran.

Laporan ini dibuat sebagai syarat dalam memperoleh gelar sarjana di jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Dalam proses pengerjaan tugas akhir ini penulis diberi dukungan dan bantuan dari berbagai pihak. Oleh karenanya, pada kesempatan ini penulis ingin menyampaikan rasa terima kasih kepada:

1. Kedua orang tua Bapak Junaedi Juddin dan Ibu Dewi Ratnawati juga Adik Farhah dan Adik Fahreza Darul Aqsa Aldi Ramadhani, serta keluarga penulis atas doa dan dukungan selama penulis menempuh Pendidikan di Universitas Islam Indonesia.
2. Bapak Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
3. Bapak Prof. Dr. Ir. Hari Purnomo, M.T., selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Hendrik, S.T., M.Eng., selaku Ketua Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Program Studi Informatika – Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Ibu Arrie Kurniawardhani, selaku Dosen Pembimbing Tugas Akhir yang telah meluangkan waktu dan membimbing penulis dalam pengerjaan tugas akhir.
7. Bapak Dr. Ing. Ridho Rahmadi, M.Sc., selaku Dosen Pembimbing Tugas Akhir yang telah meluangkan waktu dan membimbing penulis dalam pengerjaan tugas akhir.
8. Seluruh dosen di Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
9. Teman-teman Angkatan 2015 Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

10. Teman-teman KKN Unit 205 Angkatan 59 yang telah memberi dukungan dan bantuan kepada penulis.
11. Teman-teman UKM Futsal FTI SPARTA ’06 dan Minaret yang selalu mendukung dan memberi nasihat kepada penulis.
12. Teman-teman seperjuangan di tim Meno FC, yang selalu menjadi motivasi saya untuk terus memberikan yang terbaik kapanpun dan dimanapun saya berada.
13. Semua pihak yang telah membantu dalam penyelesaian tugas akhir ini yang tidak bisa saya tulis satu-persatu.

Penulis menyadari bahwa tugas akhir ini masih perlu disempurnakan lagi dan tidak terlepas dari kekurangan karena keterbatasan penulis. Oleh karena itu penulis mengharapkan kritik dan saran yang membangun demi kesempuranaan tugas akhir ini. Semoga dengan tugas akhir ini dapat memberi manfaat bagi semua pihak yang bersangkutan.

***Wassalamu’alaikum Warahmatullah Wabarakatuh.***

Yogyakarta, 26 Oktober 2020

(Randy Efan Jayaputra)

# SARI

Saat ini, penyakit diabetes dan obesitas menjadi penyakit yang cukup banyak diderita manusia di dunia. Berdasarkan data yang dihimpun *WHO Global Report*, pada tahun 2000, di Indonesia sudah terdapat 8,4 juta jiwa penderita Diabetes. Diabetes juga diperkirakan pada tahun 2030 nanti akan sampai pada 21,3 juta jiwa penderitanya di Indonesia. Penyakit-penyakit seperti diabetes dan obesitas sendiri, disebabkan oleh salah satunya adalah kelebihan asupan kalori pada tubuh.

Kelebihan asupan kalori pada tubuh terjadi disebabkan oleh manusia yang belum mampu mengatur kadar kalori yang masuk ke dalam tubuh melalui makanan dan minuman yang dikonsumsinya (Tirasirichai, Thanomboon, Soontorntham, Kusakunniran, & Robinson, 2018).

Pada penelitian ini, diusulkan sebuah *mobile application* yang dapat melakukan estimasi kalori pada makanan yang akan dimakan, melalui kamera *mobile devices* menggunakan model kecerdasan buatan. Model kecerdasan buatan yang dibangun adalah model *deep learning*, yang memanfaatkan algoritma *SSD (Single Shot Multibox Detector)* untuk melakukan deteksi pada objek makanan.

Berdasarkan hasil pengujian performa model dan juga pengujian estimasi kalori dengan nilai sebenarnya, *Mobile Application* yang dibangun sudah mampu untuk melakukan estimasi kalori pada makanan yang akan dimakan. Aplikasi juga diharapkan mampu untuk mengurangi resiko terkena penyakit-penyakit seperti diabetes dan obesitas yang saat ini dialami oleh banyak manusia.

Kata kunci: *SSD, deep learning,* kalori, *mobile application*.

# GLOSARIUM

| | |
|---|---|
| iOS | iPhone Operating System. |
| AI | Artificial Intelligence. |
| Waterfall | metode pengembangan perangkat lunak |

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Diabetes, Obesitas, dan penyakit-penyakit degeneratif lain merupakan penyakit yang saat ini banyak ditemukan di masyarakat Indonesia bahkan dunia. Berdasarkan data yang dihimpun *WHO Global Report*, pada tahun 2000, di Indonesia sudah terdapat 8,4 juta jiwa penderita Diabetes. Diabetes juga diperkirakan pada tahun 2030 nanti akan sampai pada 21,3 juta jiwa penderitanya di Indonesia. Diabetes dan Obesitas sendiri salah satunya disebabkan oleh kelebihan asupan kalori pada makanan daripada yang dibakar oleh tubuh.

Jumlah kalori yang masuk ke dalam tubuh (biasanya dari makanan dan minuman yang dikonsumsi) sering kali lebih banyak daripada kalori yang dibakar (melakukan olahraga, dan aktifitas tubuh yang lain). Jumlah kalori itu seringkali tidak sesuai dengan yang dibutuhkan oleh tubuh masing-masing. Manusia belum mampu mengatur kadar kalori yang masuk ke dalam tubuh melalui makanan dan minuman yang dikonsumsinya (Tirasirichai, Thanomboon, Soontorntham, Kusakunniran, & Robinson, 2018).

Pada penelitian ini, sebuah aplikasi *mobile devices* dan model kecerdasan buatan (*Artificial* Intelligence) dibangun untuk mengatasi masalah manusia yang belum mampu mengestimasi kalori yang masuk ke dalam tubuh dari makanan. Aplikasi ini akan memanfaatkan model kecerdasan buatan (*Artificial Intelligence*) untuk melakukan deteksi pada makanan hanya dengan menggunakan kamera dari *mobile devices*. Aplikasi juga akan mampu melakukan estimasi kalori dari makanan yang telah berhasil dideteksi.

Model kecerdasan buatan yang dibangun merupakan model *Deep Learning* yang memanfaatkan algoritma *SSD (Single Shot Multibox Detector*) untuk bisa mengenali jenis makanan melalui citra makanan. Jenis makanan yang sudah berhasil dikenali oleh model di dalam aplikasi akan diasumsikan porsinya untuk dapat diestimasi kandungan kalori dari jenis makanan tersebut.

Dengan memanfaatkan aplikasi yang dibangun di *mobile devices*, akan mempermudah setiap orang untuk mengestimasi asupan kalori yang masuk melalui makanan. Harapannya, orang-orang dapat mengurangi resiko terkena penyakit seperti Diabetes dan Obesitas dengan menjaga asupan kalori yang masuk dan keluar dari tubuhnya hanya menggunakan *mobile device* yang dimiliki.

## 1.2 Rumusan Masalah

Adapun rumusan masalah pada penelitian tugas akhir ini adalah sebagai berikut:

a. Bagaimana membangun *mobile application* yang memanfaatkan model kecerdasan buatan untuk mampu mengestimasi jumlah kalori dari makanan?
b. Bagaimana performa model dan *mobile application* saat melakukan deteksi jenis makanan?

## 1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian tugas akhir ini adalah sebagai berikut:

a. Jenis makanan Indonesia yang mampu dikenali oleh model *SSD* (*Single Shot Detector*) ada sembilan jenis makanan berbeda, yaitu dada ayam goreng, kerupuk putih, tahu goreng, tempe goreng, nasi putih, telur dadar, sambal, tumis kangkung, dan ikan lele goreng.
b. Estimasi kalori dilakukan dengan asumsi kadar kalori makanan per porsi.
c. Estimasi porsi makanan yang menjadi dasar dalam mengestimasi jumlah kalori mengacu kepada porsi makanan yang diberikan oleh *FatSecret Indonesia*.
d. Karena sedang terjadi pandemi, pengambilan gambar untuk *training set* dan *test set* dalam pembangunan model *SSD* sebagian besar diambil di Internet dan sebagian kecil lain diambil menggunakan kamera handphone.
e. Aplikasi yang dibangun untuk mengimplementasikan model *deep learning* berbasis iOS (Iphone Operating System).

## 1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah sebagai berikut:

a. Membangun model *deep learning* untuk mendeteksi jenis-jenis makanan Indonesia.
b. Melakukan estimasi kalori pada jenis-jenis makanan yang sudah berhasil dideteksi.
c. Mengimplementasikan model yang sudah dibangun ke dalam *mobile application* dengan baik.

## 1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memudahkan orang Indonesia untuk mengestimasi asupan kalori yang masuk ke dalam tubuh melalui makanan.

## 1.6 Sistematika Penulisan

Sistematika penelitian bertujuan untuk memudahkan dalam melakukan pembahasan tugas akhir Estimasi Kalori pada makanan melalui citra makanan menggunakan model *deep learning* pada *mobile devices*. Penelitian ini terbagi dalam lima bab. Adapun rincian dari masing-masing bab tersebut adalah:

**Bab I Pendahuluan,** berisi latar belakang, batasan masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan yang dijadikan bahan acuan dan materi mengenai penelitian yang dilakukan penulis.

**Bab II Landasan Teori,** berisi teori-teori yang berkaitan dan menjadi pedoman dasar dalam penelitian ini. Teori-teori tersebut berhubungan dengan seperti apa Kalori, sampai ke pembahasan teknikal tentang *Deep Learning* yang akan digunakan di dalam penelitian ini.

**Bab III Metodologi,** berisi analisis kebutuhan yang digunakan dalam penelitian ini, juga alur perancangan model, sampai perancangan implementasi ke *mobile application*.

**Bab IV Hasil dan Pembahasan,** berisi hasil penelitian yang sudah dilakukan, dan implementasi dari perancangan yang sudah di siapkan di bab sebelumnya.

**Bab V Kesimpulan dan Saran,** berisi kesimpulan dari hasil pembahasan penelitian dan saran terhadap penelitian yang sudah dilakukan.

# BAB II
# TINJAUAN PUSTAKA

## 2.1 Penelitian Terkait

Penelitian *Calories Prediction From Food Image* (Chokr & Elbassuoni, 2017) melakukan studi tentang bagaimana membangun sebuah pemodelan untuk mengestimasi kalori berdasarkan citra makanan. Pendekatan yang digunakan dalam penelitian itu adalah *Feature Extraction*, yang digunakan untuk mengekstrak fitur dari citra makanan, lalu melakukan *Data Compression*, setelah itu *Food Type Classification* untuk mengklasifikasi citra makanan tersebut ke beberapa kelas. Pada penelitian tersebut, makanan diklasifikasi ke dalam enam kelas (*burger*, *chicken*, *doughnut*, *pizza*, *salad,* dan *sandwich*). Lalu masuk di tahapan *Food Size Prediction*, yang mana pada tahapan ini digunakan untuk mengestimasi ukuran/berat dari citra makanan yang diberikan, dengan mengambil data ukuran dan berat makanan dari yang tersedia online setelah gambar makanan diklasifikasi. Salah satu contohnya, untuk ukuran *McDonald's* item, ada *tool* yang tersedia online yang memberikan informasi mengenai ukuran/berat termasuk kalori yang terkandung didalam jenis makanan tersebut.

Sebuah penelitian (Ege & Yanai, 2017) telah dilakukan untuk mengestimasi kalori pada citra makanan yang diberikan. Pada penelitian tersebut, estimasi kalori pada makanan dilakukan melalui pendekatan *Multi-Task CNN (Convolutional Neural Network)*; prediksi kategori makanan dilakukan serentak dengan prediksi kalori yang ada pada makanan. Takumi Ege & Keiji Yanai pada penelitiannya ini membangun sebuah jaringan *convolutional* untuk melakukan dua tugas tersebut secara bersamaan. Di dalam dataset yang digunakan terdapat informasi jenis makanan dan juga kandungan kalori yang ada pada makanan tersebut, yang kemudian digunakan untuk melatih jaringan *multi-task CNN* yang dibangun. Namun pada penelitian ini masih ditemukan beberapa kekurangan, salah satunya kalori dan kategori makanan hanya bisa dideteksi pada gambar yang hanya terdapat satu jenis makanan (*single-dish*).

Sebuah studi (Miyazaki, De Silva, & Aizawa, 2011) menggunakan foto makanan harian yang diunggah oleh berbagai *user* di web dengan nama "*FoodLog*". Foto-foto tersebut didapat dari *user* untuk kemudian diestimasi oleh *expert nutrition*; jumlah kandungan kalori yang dijadikan sebuah dataset. Setiap gambar yang diberikan dibandingkan dengan *ground truth* dari berbagai macam fitur-fitur di gambar (*multiple image features*), seperti, *Color Histogram,*

*Color Correlograms* dan *SURF Features*. Lalu, *ground truth* akan diurutkan berdasarkan kemiripan dengan gambar yang diberikan. Kandungan kalori dari sebuah gambar yang diberikan dihitung oleh *Linear Estimation* menggunakan peringkat kalori tertinggi di *multiple image features*. Metode ini menunjukkan bahwa 79% dari estimasi benar dalam kurang lebih 40% *error*, dan 35% benar di dalam kurang lebih 20% *error*.

Sebuah studi lainnya (Liang & Li, 2017) mengestimasi kalori dengan cara mengambil gambar pada dua sisi pada makanan dan menggunakan objek kalibrasi. Penelitian ini menggunakan *Faster R-CNN* untuk mendeteksi objek dan *Grab-Cut* sebagai algoritma segmentasi. Penelitian ini berfokus pada jenis buah-buahan karena bentuknya mudah untuk digolongkan (*ellipsoid, collumn, irregular*).

Sebuah penelitian (Wasif & Pereira, 2019) melakukan estimasi kalori pada makanan melalui beberapa tahapan. Pertama, adalah melakukan deteksi pada makanan, lalu setelahnya melakukan *segmentation* pada hasil deteksi objek itu, lalu dikalkulasi *volume* dari objek dengan menggolongkan hasil segmentasi ke dalam 2 bentuk; *ellipsoid* atau *other*. Setelah volume didapatkan, barulah kalori bisa diestimasi menggunakan hitungan kalori per gram.

Pada penelitian ini, pendekatan yang digunakan untuk melakukan estimasi pada kalori adalah, pemanfaatan *mobile devices* yang digunakan sehari-hari dan juga memanfaatkan model kecerdasan buatan. Model kecerdasan buatan digunakan untuk dapat melakukan deteksi pada objek makanan yang ada di dalam piring, lalu *mobile application* akan dimanfaatkan untuk melakukan estimasi kalori dengan mengambil informasi kalori per porsi dari FatSecret. Model kecerdasan buatan yang dibangun di penelitian adalah model *SSD (single shot multibox detector)* sebagaimana diketahui merupakan model *object-detection* dengan performa dan waktu komputasi yang sangat baik.

Tabel 2.1 Referensi Penelitian Sebelumnya

| No. | Penulis, Tahun Penelitian | Judul Penelitian | Metode Penelitian | Pembahasan |
|---|---|---|---|---|
| 1. | Chokr & Elbassuoni, 2017 | *Calories Prediction from Food Image* | *Image Processing* | Melakukan klasifikasi pada citra makanan yang diberikan, lalu sistem akan mengambil ukuran dan berat makanan dari yang tersedia di internet, untuk kemudian diestimasi. |
| 2. | Ege & Yanai, 2017 | *Simultaneous Estimation of Food Categories and Calories with Multi-task CNN* | *Multi-Task CNN* | *Multi task CNN*, melakukan klasifikasi dan estimasi kalori di dalam satu arsitektur CNN, dengan menyiapkan dataset yang sudah terdapat kandungan kalorinya di dalam dataset tersebut. |
| 3. | Miyazaki, De Silva, & Aizawa, 2011 | *Image-based Calorie Content Estimation for Dietary Assessment* | *Image Processing* | Menggunakan *Color Histogram, Color Correlograms* dan *SURF Features*. Lalu untuk kandungan kalori dihitung oleh *Linear Estimation* menggunakan peringkat kalori tertinggi di *multiple image features* |
| 4. | Liang & Li, 2017 | *Deep Learning-Based Food Calorie Estimation Method in Diatery Assessment* (Liang & Li, 2017) | *Image Processing* | penelitian ini mengharuskan untuk mengambil gambar pada dua sisi pada makanan, dan menggunakan objek kalibrasi, untuk mengestimasi ukuran dari makanan. Penelitian ini menggunakan *Faster R-CNN* untuk mendeteksi objek dan *Grab-* |

| No. | Penulis, Tahun Penelitian | Judul Penelitian | Metode Penelitian | Pembahasan |
|---|---|---|---|---|
| | | | | *Cut* sebagai algoritma segmentasi. Penelitian ini baru dilakukan pada beberapa makanan dan paling banyak pada buah, karena bentuk dari buah mudah untuk digolongkan kedalam tiga tipe yang sudah ditentukan di penelitian ini, *ellipsoid, collumn, irregular*. |
| 5. | Wasif & Pereira, 2019 | *Food Calorie Estimation using Machine Learning and Image Processing* | *Image Processing* | untuk melakukan estimasi kalori pada makanan, metode yang digunakan pada penelitian ini melalui beberapa tahapan. Pertama, adalah melakukan deteksi pada makanan, lalu setelahnya melakukan *segmentation* pada hasil deteksi objek itu, lalu dikalkulasi *volume* dari objek dengan menggolongkan hasil segmentasi ke dalam 2 bentuk; *ellipsoid* atau *other*. Setelah volume didapatkan, barulah kalori bisa diestimasi menggunakan hitungan kalori per gram. |

## 2.2 Kalori

Kalori merupakan takaran energi dalam makanan yang dibutuhkan oleh manusia untuk bisa bertahan hidup (beraktifitas). Namun, kelebihan asupan kalori yang dikonsumsi oleh tubuh manusia juga dapat berakibat fatal, contohnya adalah, penyakit Diabetes dan Obesitas yang ditemukan di masyarakat saat ini.

Dalam makanan, terdapat tiga jenis makronutrien, yaitu lemak, protein, dan karbohidrat. Per 1 gram lemak mengandung 9 kalori, 1 gram protein mengandung 4 kalori, dan 1 gram karbohidrat sendiri mengandung 4 kalori. Satuan kalori biasanya disingkat sebagai 'kal'.

Secara umum, asupan kalori yang dibutuhkan oleh manusia berbeda-beda. Sebagai contoh, rata-rata pria memerlukan 2.000 – 3.000 kalori, dan rata-rata wanita, memerlukan 1.600 – 2.400 kalori. Kalori yang dibutuhkan manusia tergantung daripada aktivitas keseharian yang dimiliki (jumlah energi yang dikeluarkan oleh setiap manusia saat beraktivitas setiap harinya), juga tinggi badan, berat badan, dan faktor-faktor lain seperti usia.

Penyebab besar dari penyakit-penyakit seperti obesitas dan diabetes terjadi pada tubuh seseorang adalah karena orang tersebut tidak bisa menyesuaikan konsumsi kalori yang masuk ke dalam tubuh, dan juga aktivitas yang dilakukannya itu membakar berapa kalori. Hal ini berdampak pada tubuh yang akan kekurangan/kelebihan asupan kalori untuk melakukan aktivitas.

## 2.3 *FatSecret Indonesia*

*FatSecret Indonesia* adalah sebuah *tools online* informasi diet, nutrisi, dan program penurunan berat badan, juga menjadi komunitas yang mengizinkan *user* untuk menentukan profil yang unik, memantau aktifitas, makanan, dan berat badan dari tiap user. *FatSecret* berkontribusi untuk memberikan informasi dan konten untuk membantau orang mencapai diet mereka. *FatSecret Indonesia* memiliki data-data yang cukup lengkap mengenai nutrisi (termasuk kalori) yang ada di dalam banyak makanan di Indonesia.

## 2.4 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) merupakan paradigma pemrosesan informasi yang meniru cara kerja sistem sel saraf biologi yang dimiliki manusia, seperti cara kerja otak manusia ketika memproses suatu informasi yang didapat oleh dunia luar. Jaringan syaraf tiruan belajar dari suatu contoh. Biasanya, pemodelan jaringan syaraf tiruan dibentuk untuk memecahkan suatu

masalah tertentu seperti pengenalan pola atau klasifikasi. JST memungkinkan suatu sistem belajar dan melakukan generalisasi sehingga diharapkan sistem tidak hanya mengenali data-data yang sudah pernah diberikan, tetapi juga data baru (Staub, Karaman, Kaya, Karapınar, & Güven, 2015)

## 2.5 *Deep Learning*

*Deep Learning* merupakan pengembangan dari jaringan saraf tiruan yang menggunakan metadata sebagai *input* dan mengolahnya menggunakan sejumlah lapisan tersembunyi *(hidden layer)* yang ditata berlapis-lapis dan mendalam. *Deep Learning* disebut sebagai *Deep* karena struktur dan jumlah jaringan saraf pada algoritmanya sangat banyak bisa mencapai puluhan hingga ratusan lapisan.

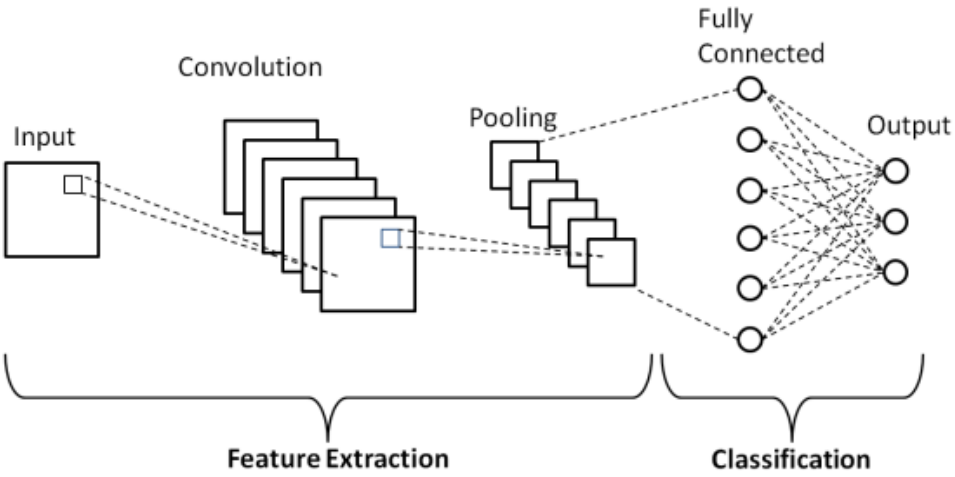
## 2.6 *Convolutional Neural Network (CNN)*

*Convolutional Neural Network (ConvNet/CNN)* adalah sebuah *multilayer perceptron* yang didesain spesial untuk mengidentifikasi informasi dari citra (LeCun, Haffner, Bottou, & Bengio, 1999). *CNN* merupakan salah satu implementasi dari *deep learning*, karena dalamnya tingkat jaringan dan banyak diimplementasikan di dalam citra. CNN merupakan algoritma pengenal yang efisien dan banyak sekali digunakan dalam pengenalan pola dan pastinya pengolahan citra.

Secara teknik, *convolutional network* adalah arsitektur yang bisa dilatih untuk mengenali objek-objek yang terdapat di dalam sebuah gambar, dalam prosesnya yang terdiri dari beberapa tahap. Masukan dan keluaran dari masing-masing tahap adalah beberapa *array* yang disebut *feature map* atau peta fitur.

### 2.6.1 Arsitektur Jaringan CNN

Ilustrasi model CNN ditampilkan pada Gambar 2.1.




Gambar 2.1 Arsitektur Jaringan CNN

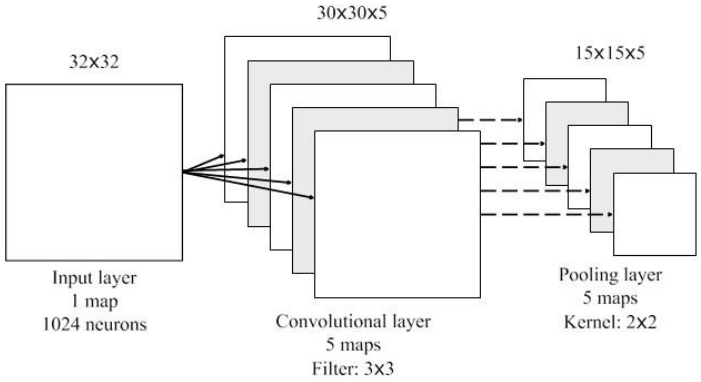
Sumber: (Phung & Rhee, 2019)

Secara garis besar, arsitektur jaringan CNN terbagi menjadi dua bagian; *Feature Learning* dan *Fully-Connected Layer (MLP)*. Untuk *Feature Learning* sendiri di dalamnya terdiri dari tiga layer yaitu konvolusi, *activation layer,* dan *pooling layer*, sedangkan untuk *Fully-Connected Layer (MLP)* merupakan *multilayer perceptron*.

### 2.6.2 *Feature Learning Layer*

Layer ini merupakan sekumpulan proses "*encoding*" sebuah *inputan* gambar ke *features* yang berupa angka-angka yang merepresentasikan gambar tersebut.

***Convolutional Layer (Convolutional Layer)***




Gambar 2.2 Ilustrasi *convolutional layer*

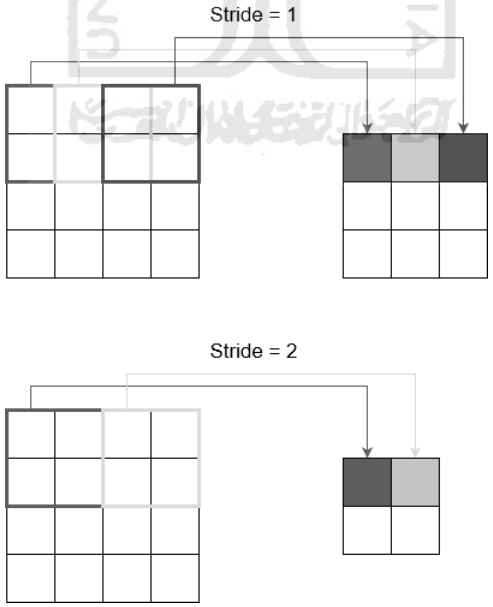
Sumber: https://socs.binus.ac.id/2017/02/27/convolutional-neural-network/

*Convolutional Layer* sendiri terdiri dari *neurons* yang tersusun sedemikian rupa sehingga membentuk sebuah *filter*. Sebagai contoh, layer pertama pada feature extraction layer biasanya adalah *convolutional layer* dengan ukuran 3x3x3. Panjang 3 piksel, tinggi 3 piksel, dan biasanya ada 3 buah sesuai dengan channel dari *image* tersebut, seperti ditampilkan di ilustrasi pada Gambar 2.2.

Ketiga *filter* ini akan digeser ke seluruh bagian dari gambar. Setiap pergeseran akan dilakukan operasi "dot" antara *input* dan nilai dari filter tersebut sehingga menghasilkan sebuah *output* atau biasa disebut sebagai *Activation Map* atau *Feature Map*.

***Stride***

*Stride* merupakan parameter yang menentukan berapa jumlah pergeseran filter ketika dilakukan konvolusi pada sebuah gambar. Jika nilai *stride* adalah 2, maka *convolutional filter* akan terus bergeser sebanyak 2 piksel secara *horizontal* lalu *vertical.* Dalam beberapa penelitian mengatakan, semakin kecil nilai stride yang digunakan maka akan semakin detail informasi yang berhasil didapatkan dari sebuah *input*. Namun, semakin kecil nilai stride itu, membutuhkan komputasi yang semakin besar. Dalam beberapa penilitian juga mengemukakan, perlu diperhatikan bahwa penggunaan nilai *stride* yang lebih kecil tidak selalu menghasilkan performa yang lebih bagus. Ilustrasi *stride* digambarkan pada Gambar 2.3




Gambar 2.3 Ilustrasi *stride*

Sumber: https://medium.com/@nadhifasofia/1-convolutional-neural-network-convolutional-neural-network-merupakan-salah-satu-metode-machine

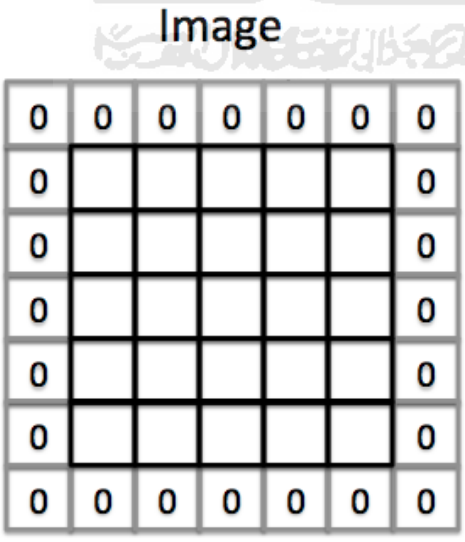
***Padding***

*Padding* adalah parameter yang menentukan jumlah piksel (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi *output* dari *convolutional layer* (*Feature Map*).

Tujuan dari penggunaan *padding* adalah dimensi *output* dari *convolutional layer* selalu lebih kecil dari *inputnya* (kecuali penggunaan 1x1 filter dengan *stride* 1). *Output* ini akan digunakan kembali sebagai input dari convolutional layer selanjutnya, sehingga sudah pasti makin banyak informasi yang terbuang.

Dengan menggunakan *padding*, kita dapat mengatur dimensi *output* agar tetap sama seperti dimensi *input* atau setidaknya tidak berkurang secara drastis. Sehingga kita bisa menggunakan *convolutional layer* yang lebih dalam sehingga lebih banyak *features* yang berhasil di-*extract*.

Penggunaan *padding* akan meningkatkan performa dari model karena *convolutional filter* akan fokus pada informasi yang sebenarnya, yaitu yang berada diantara *zero padding* tersebut. Pada Gambar 2.4, dimensi dari *input* sebenarnya adalah 5x5, jika dilakukan *convolution* dengan filter 3x3 dan *stride* sebesar 2, maka akan didapatkan *feature map* dengan ukuran 2x2. Namun jika kita tambahkan *zero padding* sebanyak 1, maka *feature map* yang dihasilkan berukuran 3x3 (lebih banyak informasi yang dihasilkan). Pada Gambar 2.4 ditampilkan ilustrasi dari *padding*




Gambar 2.4 Ilustrasi *zero padding*

Sumber: https://deepai.org/machine-learning-glossary-and-terms/padding

**Fungsi Aktivasi**

Fungsi Aktivasi adalah tahapan yang dilakukan sebelum masuk ke *pooling layer* dan setelah melakukan proses *convolution layer*. Di tahap ini, hasil dari proses konvolusi akan dikenakan fungsi aktivasi atau *activation function*. Terdapat beberapa fungsi aktivasi yang banyak sekali digunakan di *convolutional network* saat ini, di antaranya adalah *sigmoid*, t*anh* dan *reLU.*

Fungsi aktivasi sigmoid berguna untuk mengubah nilai aktivasi menjadi nilai antara 0 sampai 1. Hal ini berguna untuk masalah Klasifikasi Biner dan memang sebagian besar digunakan pada lapisan output dari masalah tersebut. Pada persamaan 2.1, merupakan persamaan matematika dari fungsi *sigmoid.*

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.1)$$

Fungsi aktivasi *Tanh* ini berguna untuk mengubah nilai aktivasi menjadi nilai antara -1 sampai 1. fungsi aktivasi ini lebih baik dari fungsi aktivasi sigmoid dalam banyak kasus karena nilai outputnya dinormalisasi. Persamaan matematika dari fungsi aktifasi *tanh* ditampilkan pada persamaan 2.2

$$tanh(z) = \frac{e^{z}- e^{-z}}{e^{z}+ e^{-z}} \quad (2.2)$$

Pada fungsi *reLU* nilai *output* dari neuron bisa dinyatakan sebagai 0 jika inputnya adalah negatif. Jika nilai input dari fungsi aktivasi adalah positif, maka output dari neuron adalah nilai input aktivasi itu sendiri.

Fungsi aktivasi *reLU* ini sering digunakan karena sifatnya lebih berfungsi dengan baik untuk memfilter informasi pada gambar. Persamaan matematika dari fungsi aktivasi reLu ditampilkan pada persamaan 2.3

$$f(z) = \max(0, z) \quad (2.3)$$

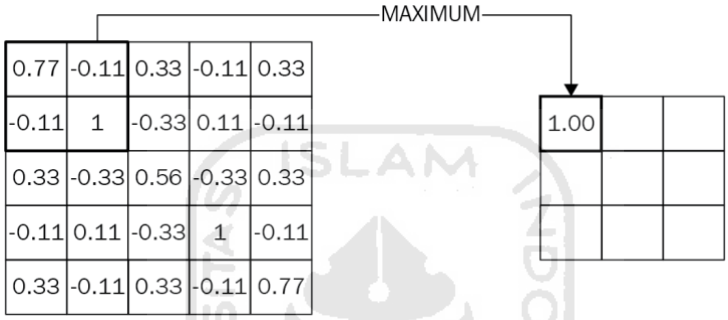
***Pooling Layer***

*Pooling layer* adalah hal yang dilakukan untuk mengurangi ukuran matriks yang didapatkan setelah melewati proses *convolutional layer*, dengan melakukan operasi *pooling.* Dua macam *pooling* yang sering digunakan adalah *average pooling* dan *max-pooling*. *Average*

*pooling* mengambil nilai rata-rata, dan pada *max pooling* nilai yang diambil adalah nilai yang paling besar.

Pada konsepnya, *pooling layer* ini terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan bergeser pada seluruh area *feature map*, dan menerapkan pooling pada tiap pergeserannya.

Tujuannya adalah tentu mengurangi dimensi dari *feature map* atau lebih dikenal dengan *downsampling*, sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan mengatasi *overfitting*. Pada Gambar 2.5, merupakan ilustrasi dari *max pooling layer*.




Gambar 2.5 Ilustrasi dari *pooling layer*

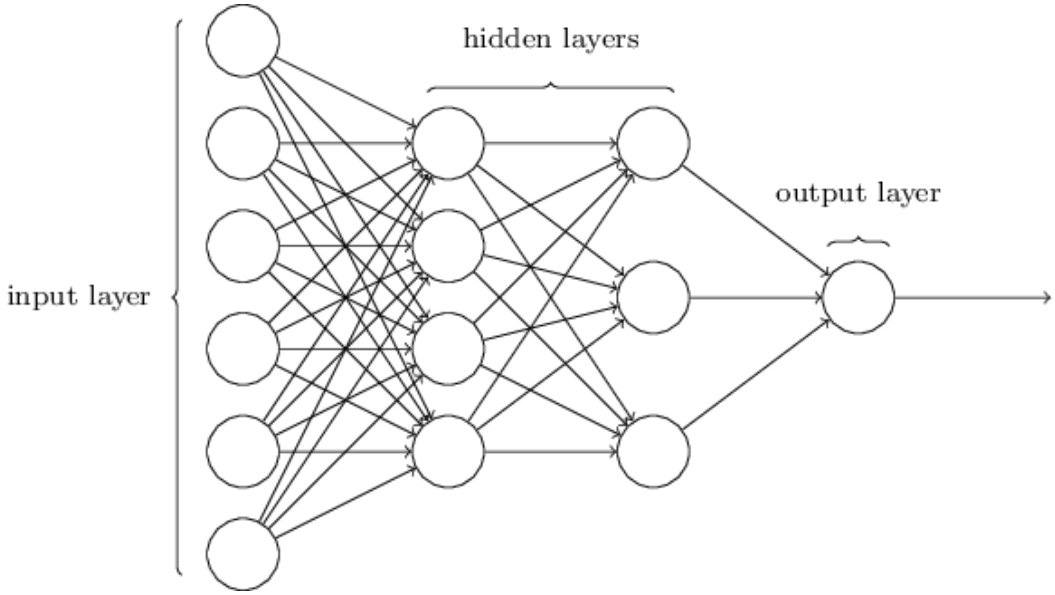
Sumber: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

### 2.6.3 *Fully-Connected Layer*

Setelah melewati banyak tahapan di *Feature Extraction Layer*, akan dihasilkan *multidimensional array*, sehingga harus dilakukan "*flatten*" atau *reshape feature map* menjadi sebuah vector agar bisa digunakan sebagai input dari *fully-connected layer*.

*Fully-Connected Layer* sendiri adalah lapisan di mana semua neuron aktivasi dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti halnya jaringan saraf tiruan biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua neuron di *Fully-Connected Layer*, itu kenapa perlu dilakukan proses "*Flatten*".

*Fully-Connected Layer* biasanya digunakan untuk mengolah data sehingga data tersebut bisa diklasifikasikan. Pada Gambar 2.6, merupakan ilustrasi dari *fully-connected layer*



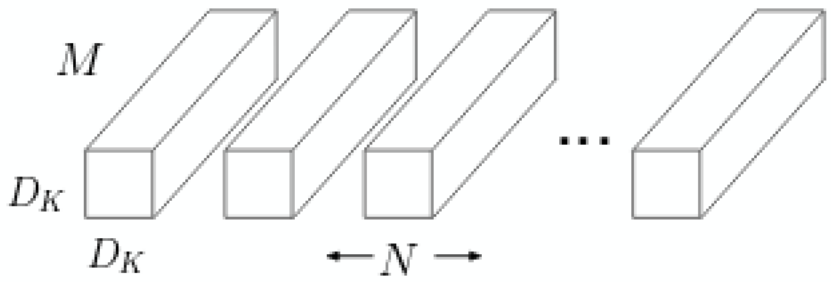

Gambar 2.6 Ilustrasi dari *fully-connected layer*

Sumber: (Saracoglu & Altural, 2010)

## 2.7 *Mobile-Net*

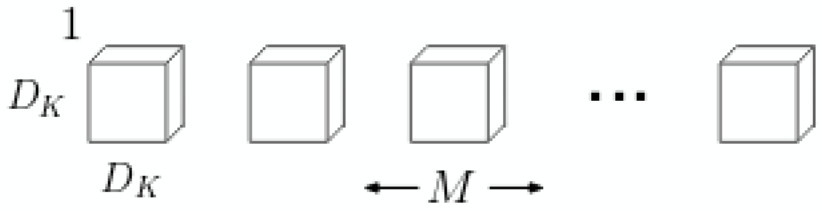
*Mobile-Net* merupakan salah satu arsitektur *convolutional neural network* (*CNN*) yang dapat digunakan untuk mengatasi masalah kebutuhan akan *computing resource* berlebih. *Mobile-Net* ini dikembangkan oleh para peneliti dari *Google*, dengan tujuan membuat arsitektur *CNN* yang dapat digunakan untuk ponsel.

Ada beberapa hal yang mendasari arsitektur *mobile-net* ini dengan arsitektur *convolutional neural network* pada umumnya, seperti pada *MobileNet V1*. Pada arsitektur tersebut, proses konvolusi dibagi menjadi *depthwise convolutionalolution* dan *pointwise convolutionalolution*.

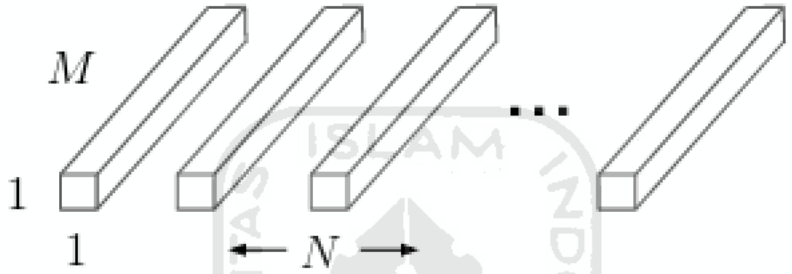



Gambar 2.7 *Standard Convolutional Filters*

Sumber: (Howard et al., 2017)




Gambar 2.8 *Depthwise Convolutional Filters*
Sumber*:* (Howard et al., 2017)




Gambar 2.9 1x1 *Convolutional Filters* atau bisa disebut *Pointwise Convolution*
Sumber: (Howard et al., 2017)

Pada *mobilenet V1*, satu konvolusi, dibagi menjadi dua bagian seperti yang ditunjukkan pada Gambar 2.8 dan Gambar 2.9.

Sedikit perubahan pada *mobilenet V2*, yang dirilis bulan April 2017, yang masih juga menggunakan *depthwise convolutional* dan *pointwise convolutional*, tetapi ditambahkan dua fitur tambahan, yaitu; *Linear Bottleneck* dan *Shortcut Connections* antar *Bottleneck*.

Sehingga penggunaan arsitektur ini pada resource komputasi yang lebih kecil seperti pada *Mobile Device,* dan lain-lain, akan sangat cocok, apalagi untuk *task-task* seperti *object-detection*.

Di tahun 2019, versi terbaru dari *mobilenet* dirilis, dengan nama *mobilenet V3,* juga dengan desain arsitektur baru. *Mobilenet V3* disetel ke *processor ponsel* melalui kombinasi *hardware-aware network architecture search (NAS)* dilengkapi dengan *NetAdapt* algoritma, *mobilenet V3* akan lebih cepat dan lebih akurat daripada *Mobilenet V2* di *classification task*. Tapi itu tidak sepenuhnya benar untuk *task-task* lain seperti *Object-Detection*.

## 2.8 *Object-Detection*

*Object-Detection* adalah teknik komputer vision untuk menetapkan lokasi dari objek di dalam sebuah gambar atau video. Di dalam *Deep Learning, task object-detection* akan menggunakan *Convolutional Neural Network* sebagai *backbone* dan menggunakan algoritma seperti *R-CNN*, Yolo v2, SSD, untuk mendeteksi lokasi dari objek yang dikenali oleh *CNN*.

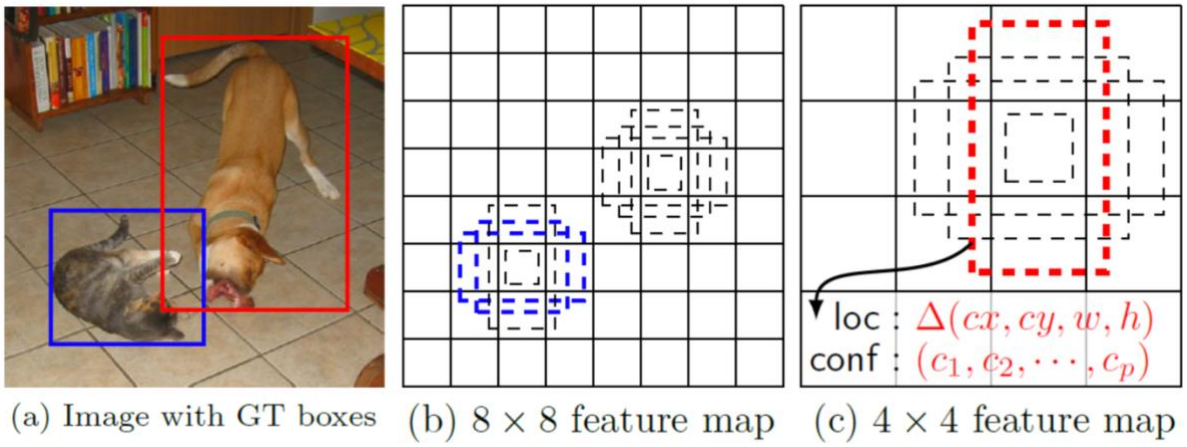
*Object-Detection* juga merupakan sebuah *task* yang ada di dalam *deep learning* yang bertujuan menetapkan sebuah *bounding box* di sekitar objek yang dideteksi pada suatu gambar dan mengasosiasikannya dengan kategori objek dari setiap *bounding box*.

## 2.9 SSD (*Single Shot Detector*)

SSD atau singkatan dari *Single Shot Detector* adalah salah satu dari beberapa algoritma *object-detection* yang ada saat ini. SSD bekerja dengan algoritma yang relatif sederhana karena tidak melalui tahapan pembuatan proposal dan tahap feature resampling. Pendekatan-pendekatan lain dalam lingkup *object-detection,* seperti seri R-CNN masih membutuhkan 2 kali pembacaan pada gambar untuk *task object-detection*. Pertama, R-CNN akan men-*generate region proposal*, dan lalu melakukan deteksi objek pada tiap *proposal*.

*SSD* disebut sebagai *single shot detector* karena semua proses deteksi yang dilakukan di algoritma ini merangkum semua perhitungan dalam satu jaringan. Pada model *SSD,* arsitektur *CNN* seperti *mobile-net*, VGG-16, dan lain sebagainya dimanfaatkan untuk melakukan proses ekstraksi ciri. Setelah proses ekstraksi ciri selesai, akan menghasilkan *feature layer* berukuran M x N (jumlah lokasi) dengan P *channel*, seperti 8x8 atau 4x4. Lalu, konvolusi 3x3 diterapkan pada lapisan fitur M x N x P ini. Untuk tiap lokasi, kita akan mendapatkan beberapa kotak pembatas (*bounding box*), beberapa *bounding box* ini punya ukuran dan rasio aspek yang berbeda-beda untuk tiap lokasinya.

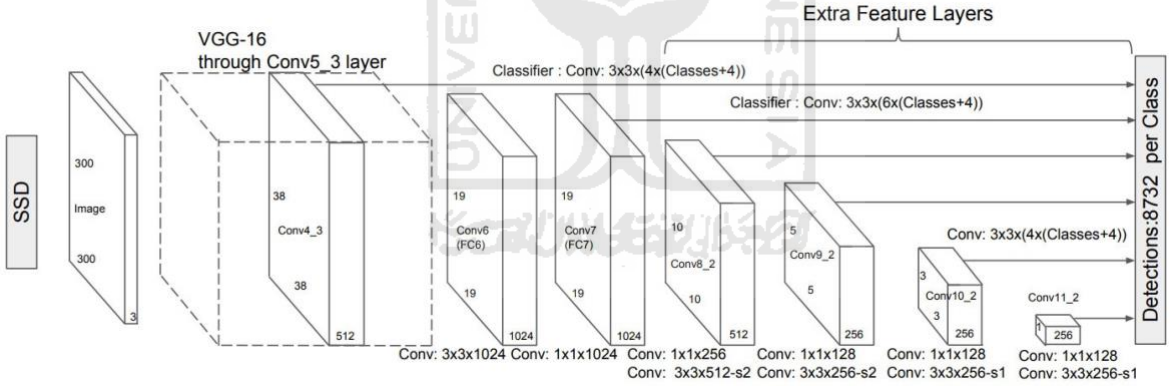
Setiap *bounding box* pada tiap lokasi nya akan dikalkulasikan skor kelas dan juga 4 *offsets* yang relatif terhadap bentuk *bounding box* asli, Seperti digambarkan pada Gambar 2.10




Gambar 2.10 Proses *SSD*

Sumber: https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection

Pada Gambar 2.11, merupakan salah satu contoh arsitektur lengkap dari jaringan *SSD*, yang disana, memanfaatkan arsitektur *VGG-16* untuk melakukan ekstraksi ciri di lapisan-lapisan awal. Lalu, setelah melewati ekstraksi ciri akan melakukan algoritma *SSD* yang sudah dijelaskan sebelumnya.




Gambar 2.11 Arsitektur jaringan *SSD*

Sumber: https://medium.com/jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing

Pada penggunaan model SSD di perangkat dengan *resource* yang lebih kecil, lapisan-lapisan awal biasanya akan menggunakan arsitektur *CNN* seperti *Mobile-Net* untuk ekstraksi ciri, sebelum dilakukan deteksi menggunakan algoritma SSD.

*SSD300* sendiri mencapai performa 74.3% mAP (*Mean Average Precision)* di 59 FPS (*Frame per Second)*. Adapun, pada *SSD500* mencapai performa 76.9% mAP di 19 FPS,

sedangkan algoritma pendekatan lain seperti, *Faster R-CNN* menghasilkan 73.2% mAP di 7 FPS dan *YOLOv1* menghasilkan 66.4% mAP di 21FPS.

| Method | mAP | FPS | batch size | # Boxes | Input resolution |
|---|---|---|---|---|---|
| Faster R-CNN (VGG16) | 73.2 | 7 | 1 | ~ 6000 | ~ 1000 × 600 |
| Fast YOLO | 52.7 | 155 | 1 | 98 | 448 × 448 |
| YOLO (VGG16) | 66.4 | 21 | 1 | 98 | 448 × 448 |
| SSD300 | 74.3 | 46 | 1 | 8732 | 300 × 300 |
| SSD512 | 76.8 | 19 | 1 | 24564 | 512 × 512 |
| SSD300 | 74.3 | 59 | 8 | 8732 | 300 × 300 |
| SSD512 | 76.8 | 22 | 8 | 24564 | 512 × 512 |

Gambar 2.12 Tabel perbandingan beberapa pendekatan *object-detection*

Sumber: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

## 2.10 *Confusion Matrix*

*Confusion Matrix* pada dasarnya berguna untuk memberikan informasi perbandingan hasil klasifikasi yang dilakukan oleh sistem (model) dengan hasil klasifikasi sebenarnya. Bentuk dari *confusion matrix* adalah tabel matriks yang menggambarkan kinerja model pada serangkaian data uji yang nilai sebenarnya sudah diketahui.

Proses perhitungan matriks untuk *multi-class classification* dan *binary classification* tentu saja berbeda. Salah satu contoh ditampilkan pada Gambar 2.13. Teks dengan warna hijau pada Gambar 2.13 merupakan nilai *True Positive* (TP) yang berhasil model deteksi.

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | $P_{GG}$ | $P_{MG}$ | $P_{SG}$ |
| | Mastiff | $P_{GM}$ | $P_{MM}$ | $P_{SM}$ |
| | Samoyed | $P_{GS}$ | $P_{MS}$ | $P_{SS}$ |

Gambar 2.13. Salah satu contoh tabel *confusion matrix* untuk *multi-class clasification*

Sumber: https://dev.to/overridedeveloper/understanding-the-confusion-matrix-264i

## 2.11 Tensorflow

Tensorflow merupakan pustaka perangkat lunak, untuk keperluan seperti *machine learning*, dan *deep neural network*. Tensorflow dikembangkan oleh tim Google Brain beserta

teknologi canggih dibalik fitur *image-recognition*-nya *Google Photos* atau *voice-*recognitionnya *Google Now*.

Tensorflow sendiri diklaim memiliki banyak keunggulan dalam pengembangan *machine* learning dan juga *deep neural network*. Keunggulan tensorflow diantaranya; dengan Tensorflow dapat mampu mendefinisikan, menghitung, dan mengoptimalkan secara efisien ekspresi matematis yang melibatkan *array* multi dimensi. Tensorflow juga memiliki kelebihan penggunaan GPU yang transparan (manajemen komputasi), dan skalabilitas komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar.

Tensorflow saat ini banyak digunakan perusahaan-perusahaan pada *task-task* dalam lingkup *machine learning* atau *deep learning*, dan mempermudah riset-riset terbaru mengenai pengembangan *deep learning* dalam masalah keseharian manusia.

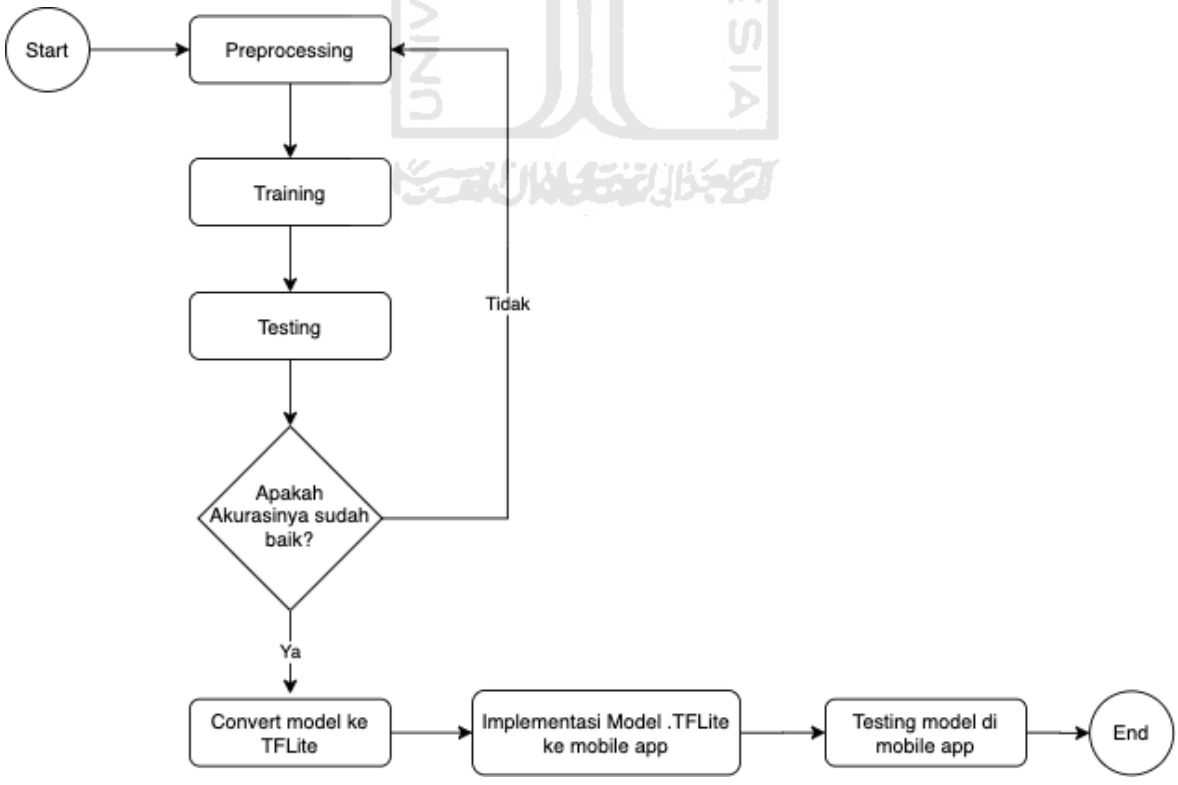
Tensorflow Lite adalah *library machine learning* yang dirancang khusus untuk menjalankan model di *mobile* atau perangkat-perangkat dengan *resource* komputasi yang cukup kecil.

# BAB III
# METODOLOGI

## 3.1 Perancangan

Terdapat enam tahap pada penelitian ini, yaitu pengumpulan data, *preprocessing*, *training*, *testing*, perancangan *user interface*, pengimplementasian model untuk estimasi kalori pada *mobile application*, pengujian model dan estimasi kalori di *mobile application*. Tahapan *preprocessing* terdiri atas mengganti tipe file gambar, menganotasi *bounding box* tiap makanan di gambar, melakukan augmentasi, serta *split* data *train* dan *test*. Sedangkan untuk tahapan *training* dan *testing* model sendiri akan dilakukan secara otomatis menggunakan bahasa pemrograman, dan juga tahapan-tahapan lain.

Secara garis besar, dalam pembangunan model sistem yang dapat mengestimasi kalori dari citra makanan ini akan menjadi dua tahapan besar, yaitu, tahapan pembangunan model *SSD*, dan tahapan implementasi model ke *mobile app* untuk mengestimasi kalori. *Flowchart* untuk tahapan pembangunan model *SSD* digambarkan pada Gambar 3.1.




Gambar 3.1 *Flowchart* Perancangan Model

### 3.1.1 Pengambilan Data

Pada tahapan pengambilan data sendiri, dibagi menjadi dua tahapan: pengambilan data kalori dan pengambilan data gambar.

**Pengambilan Data Kalori**

Di tahapan ini, karena estimasi kalori pada citra makanan masih menggunakan asumsi porsi makanan orang Indonesia, oleh karena itu, sumber kalori pada penelitian ini merujuk pada estimasi kalori per porsi yang diberikan oleh *FatSecret Indonesia*. Semua data-data kalori per porsi dari jenis makanan yang ditampilkan di Tabel 3.1 akan diambil langsung dari website publik milik *FatSecret Indonesia,* "www.fatsecret.co.id".

Tabel 3.1 Data makanan yang digunakan pada penelitian

| No | Nama Objek |
|---|---|
| 1 | Telur Dadar |
| 2 | Nasi Putih |
| 3 | Ikan Lele Goreng |
| 4 | Tumis Kangkung |
| 5 | Dada Ayam Goreng |
| 6 | Tempe Goreng |
| 7 | Tahu Goreng |
| 8 | Kerupuk Putih |
| 9 | Sambal |

**Pengambilan Data Gambar**

Pengambilan data gambar akan dilakukan menggunakan bantuan *search engine* untuk tiap objek makanan yang sudah dituliskan pada Tabel 3.1. Citra-citra yang diambil pun, harus memenuhi beberapa ketentuan yang digunakan di penelitian ini, diantaranya:

a. Objek makanan harus cukup tampak jelas di dalam gambar
b. Objek makanan tidak saling menumpuk satu sama lain di dalam gambar
c. Objek makanan pada gambar banyaknya harus sesuai perkiraan porsi makanan orang Indonesia

Beberapa gambar pada objek makanan sebagian kecil diambil menggunakan kamera handphone pribadi.

### 3.1.2 Mengganti Tipe File

Tahapan ini dilakukan karena gambar yang dikumpulkan dari berbagai sumber di internet akan menghasilkan tipe file yang berbeda-beda. Mengganti tipe file dilakukan untuk mempermudah tahapan-tahapan *pre-processing* kedepannya. Tipe file dari tiap gambar akan diselaraskan dengan file berformat "*JPEG*".

### 3.1.3 Anotasi Gambar

Setelah mengumpulkan gambar untuk sembilan jenis makanan (kelas) dan melaraskan tipe file, selanjutnya adalah proses pelabelan gambar atau anotasi gambar. Anotasi gambar ini adalah memberi label pada tiap jenis makanan dengan *bounding box*. Anotasi gambar ini menggunakan salah satu *library* berbahasa *Python*, yaitu *LabelImg*, dan *output*nya adalah file .xml untuk tiap *file* gambar. Di dalam *file* tersebut, terdapat informasi kelas dan lokasi *bounding box* dari kelas tersebut, pada kasus ini, format yang digunakan adalah PASCAL VOC untuk *bounding box*nya.

### 3.1.4 Augmentasi Gambar

Gambar yang sudah dikumpulkan dan sudah dianotasi *bounding box* untuk tiap kelasnya akan diaugmentasi untuk memperbanyak variasi data gambar, tujuannya agar model bisa lebih baik lagi untuk mengenali kelas nantinya dalam berbagai kondisi. Pada tahapan ini, akan menggunakan *library* bernama *ImgAug*, *library* berbahasa *Python* ini sangat mempermudah tahapan augmentasi. *ImgAug* bisa menyesuaikan anotasi *bounding-box* dari gambar sebelumnya dengan gambar yang sudah diaugmentasi.

Untuk tahapan augmentasi pada penelitian ini, digunakan beberapa macam teknik augmentasi seperti yang ditampilkan pada Tabel 3.2. Skema augmentasi yang digunakan adalah tiap gambar yang sudah dikumpulkan, akan melalui semua jenis augmentasi juga setiap *value* yang sudah dituliskan di Tabel 3.2.

Tabel 3.2 Teknik Augmentasi yang digunakan

| **Teknik Augmentasi** | ***Value*** | **Jumlah Augmentasi** |
|---|---|---|
| *Gamma Contrast* | 0.5, 0.6, 0.7, 0.8, 0.9 | 5 |
| *Average Blur* | 2×2, 3×3, 4×4 (Kernel) | 3 |
| *Horizontal Flip* | 1 | 1 |
| *Vertical Flip* | 1 | 1 |
| *Multiply pixel* | 0.5, 0.6, 0.7, 0.8, 0.9 | 5 |
| *Motion Blur* | 5×5 (Kernel) | 1 |
| *Affine* | 50%, 60%, 70%, 80%, 90%, 110%, 120% | 7 |
| *Additive Poisson Noise* | 15 | 1 |
| *Coarse Dropout* | 1% | 1 |
| **Total Augmentasi** | | **25** |

Pemilihan teknik augmentasi sendiri berdasar pada beberapa kondisi yang menurut penulis mungkin bisa terjadi saat model melakukan deteksi pada jenis makanan, juga menghindari kesalahan model dalam mendeteksi objek makanan, di antaranya:

a) Pencahayaan (terlalu terang atau terlalu gelap)

Kondisi ini bisa saja terjadi saat *user* mendeteksi makanan di luar atau dalam ruangan yang cukup gelap atau cukup terang. Teknik augmentasi yang digunakan untuk meminimalisir kesalahan model ketika mendeteksi dalam kondisi ini adalah, *Gamma Contrast & Multiply Pixel.*

***Gamma Contrast***

Teknik augmentasi yang menyesuaikan kontras gambar dengan melakukan kalkulasi pada piksel dengan nilai *Gamma* yang ditambahkan, yang dalam penelitian ini, value nya adalah 0.5 sampai dengan 0.9, ditampilkan pada gambar Gambar 3.2.

**(a)** **(b)** **(c)**

Gambar 3.2 **(a)** gambar asli, **(b)** gambar dengan augmentasi *gamma contrast* 0.7, **(c)** gambar dengan augmentasi *gamma contrast* 0.5

***Multiply Pixel***

Teknik augmentasi yang mengalikan seluruh *value pixel* di dalam gambar dengan *value* yang diberikan saat augmentasi, dan akan memberikan efek yang lebih gelap atau lebih terang pada gambar, contoh ditampilkan pada Gambar 3.3

**(a)** **(b)** **(c)**

Gambar 3.3 **(a)** gambar asli, **(b)** dengan *multiply* 0.7, **(c)** dengan *multiply* 0.5

b) Posisi Objek

Kondisi yang bisa terjadi apabila posisi dari tiap objek gambar yang diambil oleh *user*, berbeda dengan posisi objek dalam gambar yang diberikan. Teknik augmentasi yang digunakan untuk meminimalisir kesalahan model dalam mendeteksi di kondisi ini adalah, *Flip Horizontal dan Flip Vertical.*

***Flip Horizontal***

Teknik augmentasi yang membalik semua gambar secara *horizontal*, contoh ditampilkan pada Gambar 3.4

**(a)** **(b)**

Gambar 3.4 **(a)** gambar asli, **(b)** gambar dengan augmentasi *flip horizontal*

***Flip Vertical***

Teknik augmentasi yang membalik semua gambar secara *vertical*, contoh ditampilkan pada Gambar 3.5

**(a)** **(b)**

Gambar 3.5 **(a)** gambar asli, **(b)** gambar dengan augmentasi *flip vertical*

c) Jarak Pengambilan Gambar

Kondisi yang bisa terjadi apabila jarak pengambilan gambar yang berbeda-beda, seperti pengambilan yang terlalu dekat atau terlalu jauh dari objek makanan yang ingin dideteksi. Teknik augmentasi yang digunakan untuk meminimalisir kesalahan model dalam mendeteksi di kondisi ini adalah, *Affine*.

***Affine***

Teknik augmentasi yang melakukan *scale* pada gambar dengan *value* yang diberikan, yang dalam penelitian ini value nya adalah, 50%, 60%, 70%, 80%, 90%, 110%, dan 120%. Contoh ditampilkan pada Gambar 3.6

**(a)** **(b)** **(c)**

Gambar 3.6 **(a)** gambar asli, **(b)** gambar dengan augmentasi *affine* 60%, **(c)** gambar dengan augmentasi *affine* 120%

d) Kualitas Kamera

Kondisi yang bisa terjadi apabila kualitas gambar dari tiap user (*mobile devices*) yang nantinya bisa berbeda-beda, yang membuat model akan lemah pada kualitas gambar dari *mobile devices* tertentu. Teknik augmentasi yang digunakan untuk meminimalisir kesalahan model dalam mendeteksi di kondisi ini adalah, *Additive Poisson Noise.*

***Additive Poisson Noise***

Teknik augmentasi yang memberikan *poisson noise* pada tiap piksel di dalam gambar senilai *value* yang diberikan, dalam penelitian ini, *value* nya adalah 15. Ditunjukkan pada Gambar 3.7

**(a)** **(b)**

Gambar 3.7 **(a)** gambar asli, **(b)** gambar dengan augmentasi *Additive Poisson Noise*

e) *Blurring* Gambar

Kondisi yang bisa terjadi apabila pengambilan gambar dari *user* nantinya sedikit kabur karena bergerak. Juga untuk meningkatkan kinerja model menjadi lebih baik lagi saat melakukan deteksi secara *real-time* di *mobile application*, maka digunakan beberapa teknik augmentasi, diantaranya, *Average Blur* dan *Motion Blur* pada setiap gambar.

***Average Blur***

Teknik augmentasi yang memberikan efek blur merata pada keseluruhan gambar, hal ini membuat model nantinya masih bisa mengenali gambar jika gambar terlihat sedikit *blur*. *value* dalam augmentasi ini menggunakan ukuran kernel, diantaranya 2x2, 3x3, dan 4x4. Contoh terlihat pada Gambar 3.8

**(a)** **(b)** **(c)**

Gambar 3.8 **(a)** gambar asli, **(b)** gambar dengan augmentasi *average blur,* ukuran kernel 2x2, **(c)** gambar dengan augmentasi *average blur,* ukuran kernel 3x3

***Motion Blur***

Teknik augmentasi yang mem*blur*kan gambar dengan cara memalsukan gerakan kamera atau objek dengan ukuran kernel tertentu, yang digunakan pada penelitian ini adalah kernel ukuran 5x5. Contoh dapat dilihat pada Gambar 3.9

**(a)** **(b)**

Gambar 3.9 **(a)** gambar asli, **(b)** gambar dengan augmentasi *motion blur*

f) Derau pada Objek Gambar

Kondisi yang bisa saja terjadi apabila objek makanan yang akan dideteksi terdapat sedikit bagian yang berbeda dari objek biasanya, dan untuk mencegah model yang tidak bisa mengenali itu, maka ditambahkan satu teknik augmentasi, *Coarse Dropout* pada seluruh gambar.

***Coarse Dropout***

Teknik augmentasi yang melakukan *drop* pada piksel dalam value (persen) dari keseluruhan piksel. Yang dalam kasus ini melakukan *drop* piksel sebanyak 1% dari seluruh piksel pada gambar yang diberikan. Contoh ditunjukkan pada Gambar 3.10

**(a)** **(b)**

Gambar 3.10 **(a)** gambar asli, **(b)** gambar dengan augmentasi *coarse dropout*

Berdasarkan macam-macam teknik augmentasi yang sudah dijelaskan dan akan digunakan pada penelitian ini, setiap gambar yang berhasil dikumpulkan di subbab 3.1.1, total akan dilakukan augmentasi sebanyak 25 kali. Dengan menggunakan jenis-jenis augmentasi yang ada pada Tabel 3.2, sehingga nantinya keseluruhan gambar yang sudah dikumpulkan dan diaugmentasi menjadi *dataset* yang digunakan untuk membangun model *deep learning*.

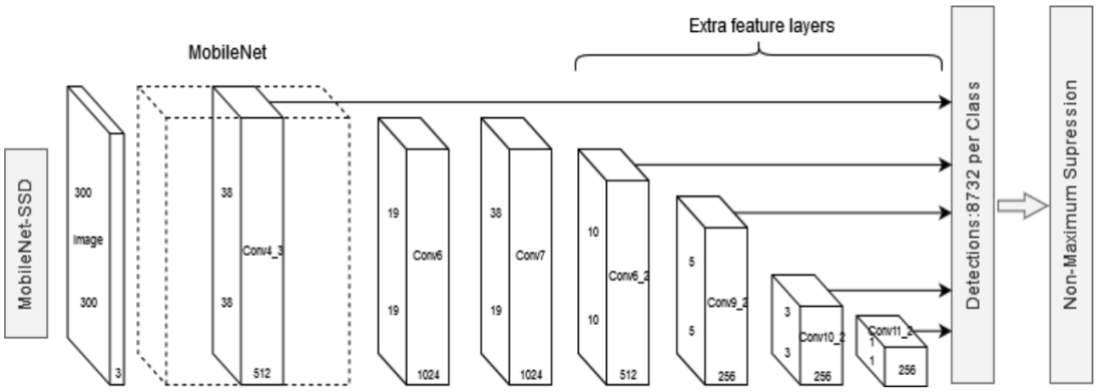
### 3.1.5 *Split Dataset*

Setelah seluruh gambar diaugmentasi, lalu tahapan selanjutnya adalah melakukan *split dataset*, membagi *dataset* gambar yang sudah dikumpulkan dan diaugmentasi menjadi 2 bagian, *Train set* dan *Test set*. Tujuannya adalah membagi data *train* yang akan digunakan untuk *training* model, dan data *test* yang nantinya akan digunakan untuk *test* performa dari model. Untuk pembagiannya sendiri, penelitian ini menggunakan rasio pembagian, yaitu, data *Train* 80% dan data *Test* 20% dari keseluruhan *dataset* gambar yang dimiliki.

### 3.1.6 *Training*

Penelitian ini menggunakan arsitektur *Mobile-Net-V2* sebagai *Backbone* jaringan syaraf tiruan (*feature extraction*), dan menggunakan algoritma *SSD (Single Shot Detector)* untuk melakukan deteksi lokasi dari objek yang dikenali pada gambar.

**Arsitektur Jaringan**




Gambar 3.11 Arsitektur Jaringan yang digunakan di penelitian ini

Seperti ditampilkan pada Gambar 3.11, adalah keseluruhan arsitektur jaringan yang digunakan. Penelitian ini menggunakan "*MobileNet-V2"* sampai pada "*conv_6*" untuk melakukan ekstraksi ciri lalu memisahkan semua konvolusi layer setelahnya. Setiap *feature map* terkoneksi langsung dengan layer pengenalan terakhir, hal ini yang memungkinkan deteksi dan lokalisasi objek dari skala yang berbeda.

Pada tahapan *training* dan *test* ini akan menggunakan *library machine learning* yang saat ini banyak sekali digunakan, yaitu *Tensorflow*.

### 3.1.7 *Testing*

Pada tahap *testing*, dilakukan dengan memasukkan dataset yang sudah dibagi ke dalam data *test*, dan akan menggunakan *TensorBoard* untuk melihat performa model dari hasil test yang sudah dilakukan.

### 3.1.8 Convert model ke TFLite

Setelah akurasi disimpulan cukup baik untuk mengenali jenis-jenis makanan yang diberikan, selanjutnya model akan dibuat "*frozen graph*"-nya, sebelum dikonversi ke *Tensorflow Lite (TFLite) Flatbuffer Format*. Setelah hal itu dilakukan, akan dihasilkan file berekstensi .pb dan .pbtxt, kedua file tersebut yang nantinya akan dikonversi lagi ke file berekstensi .tflite, dan dapat dijalankan di aplikasi *mobile devices*.

Semua proses barusan menggunakan API dari Tensorflow yang memudahkan dalam konversi model ke *frozen-graph* sampai menjadi file berekstensi .tflite yang sudah siap dijalankan di aplikasi *mobile device.*

### 3.1.9 Menjalankan Model ke *Mobile Application*

Di tahapan ini, akan dibangun sebuah aplikasi menggunakan IDE (*Integrated Development Environment)* yang biasa digunakan untuk membangun aplikasi pada sistem operasi iOS (*iphone operating system),* yaitu *XCode*. Bahasa pemrograman yang digunakan untuk membangun aplikasi di dalam sistem operasi iOS pada penelitian ini adalah bahasa *Swift.*

Lalu, tahap implementasi paling awal yang harus dilakukan untuk menjalankan model tflite di *mobile application* adalah aplikasi harus meng*import* modul tensorflowlite ke dalam aplikasi. Modul Tensorflowlite akan dimasukkan melalui *cocoapods* (*dependency manager*) yang ada saat membangun aplikasi. Selanjutnya, penulis menambahkan file .tflite model ke dalam memory aplikasi. Penulis juga harus membuat program untuk *transforming data* dari kamera ke ukuran yang bisa diterima oleh model, salah satu contohnya dalam penelitian ini adalah melakukan *resize image* sebelum diberikan ke model untuk dideteksi. Hal itu dilakukan karena ukuran yang didapatkan dari kamera *mobile device* harus disesuaikan dengan ukuran gambar yang bisa diterima oleh model.

Langkah selanjutnya, menggunakan TensorflowLite API untuk menjalankan model, langkah ini melibatkan beberapa tahapan lain, seperti membangun *Intrepreter*, dan mengalokasikan tensor. Lalu masuk ke langkah terakhir, adalah, *Intrepreting Output*, yaitu memanfaatkan kembali hasil (output) dari model tersebut.

### 3.1.10 Testing Model di *Mobile Application*

Pada tahapan testing model di *Mobile Application*, penelitian ini akan menggunakan tabel "*Confusion Matrix*". Bentuk *confession matrix* yang akan digunakan di penelitian ini adalah *confusion matrix* pada *multiclass-classification*.

Pada tahapan ini, akan diambil 30 Gambar dari *test set* yang sudah disiapkan. Model akan dijalankan menggunakan *mobile application* yang sudah dibangun, lalu hasil prediksi model pada tiap gambar akan dimasukkan ke dalam format tabel seperti ditampilkan di Tabel 3.3.

Tabel 3.3 Format tabel deteksi objek

| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Prediction Object* |
|---|---|---|---|---|
| 1 | Senin, 20 Juli 2020, 18.00 WIB | | • Nasi Putih<br>• Dada Ayam Goreng<br>• Tumis Kangkung | • Nasi Putih<br>• Dada Ayam Goreng<br>• Tumis Kangkung |

Setelah itu seluruh hasil deteksi objek pada Tabel 3.3 akan dimasukkan ke dalam Tabel 3.4, yang sudah disiapkan.

Tabel 3.4 Format Tabel *Confusion Matrix* yang digunakan

| | | **True Class** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Dada Ayam | Nasi Putih | Tahu Goreng | Tempe Goreng | Kerupuk Putih | Tumis Kangkung | Lele Goreng | Telur Dadar | Sambal |
| **Predicted Class** | Dada Ayam | **$TP_1$** | $F_{21}$ | $F_{31}$ | $F_{41}$ | $F_{51}$ | $F_{61}$ | $F_{71}$ | $F_{81}$ | $F_{91}$ |
| | Nasi Putih | $F_{12}$ | **$TP_2$** | $F_{32}$ | $F_{42}$ | $F_{52}$ | $F_{62}$ | $F_{72}$ | $F_{82}$ | $F_{92}$ |
| | Tahu Goreng | $F_{13}$ | $F_{23}$ | **$TP_3$** | $F_{43}$ | $F_{53}$ | $F_{63}$ | $F_{73}$ | $F_{83}$ | $F_{93}$ |
| | Tempe Goreng | $F_{14}$ | $F_{24}$ | $F_{34}$ | **$TP_4$** | $F_{54}$ | $F_{64}$ | $F_{74}$ | $F_{84}$ | $F_{94}$ |
| | Kerupuk Putih | $F_{15}$ | $F_{25}$ | $F_{35}$ | $F_{45}$ | **$TP_5$** | $F_{65}$ | $F_{75}$ | $F_{85}$ | $F_{95}$ |
| | Tumis Kangkung | $F_{16}$ | $F_{26}$ | $F_{36}$ | $F_{46}$ | $F_{56}$ | **$TP_6$** | $F_{76}$ | $F_{86}$ | $F_{96}$ |
| | Lele Goreng | $F_{17}$ | $F_{27}$ | $F_{37}$ | $F_{47}$ | $F_{57}$ | $F_{67}$ | **$TP_7$** | $F_{87}$ | $F_{97}$ |
| | Telur Dadar | $F_{18}$ | $F_{28}$ | $F_{38}$ | $F_{48}$ | $F_{58}$ | $F_{68}$ | $F_{78}$ | **$TP_8$** | $F_{98}$ |
| | Sambal | $F_{19}$ | $F_{29}$ | $F_{39}$ | $F_{49}$ | $F_{59}$ | $F_{69}$ | $F_{79}$ | $F_{89}$ | **$TP_9$** |

Tabel 3.4 merupakan tabel *confusion matrix* yang akan digunakan untuk mengukur performa dari model ketika dijalankan di dalam *mobile application.* Seperti yang ditampilkan pada Tabel 3.4, terdapat $TP_1$, $TP_2$, $TP_3$, $TP_4$, $TP_5$, $TP_6$, $TP_7$, $TP_8$, dan $TP_9$ yang akan diisi dengan jumlah data *true positive*. Salah satu contohnya adalah, kolom $TP_1$, yang akan diisi jumlah gambar dada ayam goreng (*true object*) yang diprediksi oleh model juga sebagai dada ayam goreng (*prediction*), itu yang dimaksud dengan data *true positive*.

Terdapat juga kolom-kolom di luar *true positive*, yang digolongkan sebagai kolom *false*, sebagai contoh, kolom $F_{51}$, yang akan diisi dengan jumlah gambar kerupuk putih (*true object*) yang diprediksi oleh model sebagai dada ayam goreng (*prediksi*).

Untuk mengukur performa dari model sendiri, penelitian ini akan menggunakan beberapa acuan seperti Presisi, *Recall*, *F1-Score*, dari tiap kelas yang bisa dideteksi oleh model, juga mengkalkulasikan *Accuracy*, *Macro Average*, dan *Weighted Average*. Tabel lengkapnya di tampilkan pada Tabel 3.5

Tabel 3.5 Tabel Performa Model

| No | Nama Kelas | Presisi | *Recall* | F1-Score | Support |
|---|---|---|---|---|---|
| 1 | Dada Ayam | | | | |
| 2 | Nasi Putih | | | | |
| 3 | Tahu Goreng | | | | |
| 4 | Tempe Gor. | | | | |
| 5 | Kerupuk Putih | | | | |
| 6 | Tumis Kangk. | | | | |
| 7 | Lele Goreng | | | | |
| 8 | Telur Dadar | | | | |
| 9 | Sambal | | | | |
| **Macro Average** | | | | | |
| **Weighted Average** | | | | | |

**Presisi**

Presisi sendiri adalah perhitungan mengenai berapa persen model bisa memprediksi suatu objek benar dari seluruh prediksi model tentang objek itu. Sebagai contoh, model

memprediksi sepuluh dada ayam goreng (prediction) dari lima belas gambar, tetapi hanya ada lima gambar ayam (*true object*) yang ada dari sepuluh prediksi tersebut. Maka model sendiri memiliki presisi sebesar 50% pada dada ayam goreng. Sebagai contoh, untuk mendapatkan presisi dada ayam, rumus yang akan digunakan dari Tabel 3.4, ditampilkan pada persamaan 3.1. yang dalam penelitian ini akan dihitung pada tiap kelas

$$Presisi = \frac{(TP_1 + F_{21} + F_{31} + F_{41} + F_{51} + F_{61} + F_{71} + F_{81} + F_{91})}{TP_1} \quad (3.1)$$

***Recall***

Recall adalah perhitungan mengenai berapa persen model mampu memprediksi suatu objek dari seluruh *true object* yang diberikan. Sebagai contoh, model memprediksi sepuluh kerupuk putih (*prediction*) dari sepuluh kerupuk putih yang diujikan. Maka model memiliki *recall* 100% untuk kelas dada ayam goreng. Sebagai contoh, untuk mendapatkan *recall* dada ayam goreng, rumus yang akan digunakan dari Tabel 3.4, ditampilkan pada persamaan 3.2. yang dalam penelitian ini akan dilakukan pada tiap kelas.

$$Recall = \frac{(TP_1 + F_{12} + F_{13} + F_{14} + F_{15} + F_{16} + F_{17} + F_{18} + F_{19})}{TP_1} \quad (3.2)$$

***F1-Score***

*F1-Score* dihitung menggunakan *mean* (rata-rata), tetapi bukan mean aritmatika biasa. Biasanya *F1-Score* ini digunakan untuk meringkas performa model ke dalam satu metrik. Untuk mendapatkan nilai *F1-Score* pada setiap kelas digunakan persamaan 3.3

$$F1Score = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)} \quad (3.3)$$

***Support***

*Support* adalah jumlah sampel objek yang diujikan di dalam tabel *confusion matrix*, sehingga dapat diketahui di dalam data uji itu terdapat berapa objek masing-masing di tiap kelasnya.

***Macro Average***

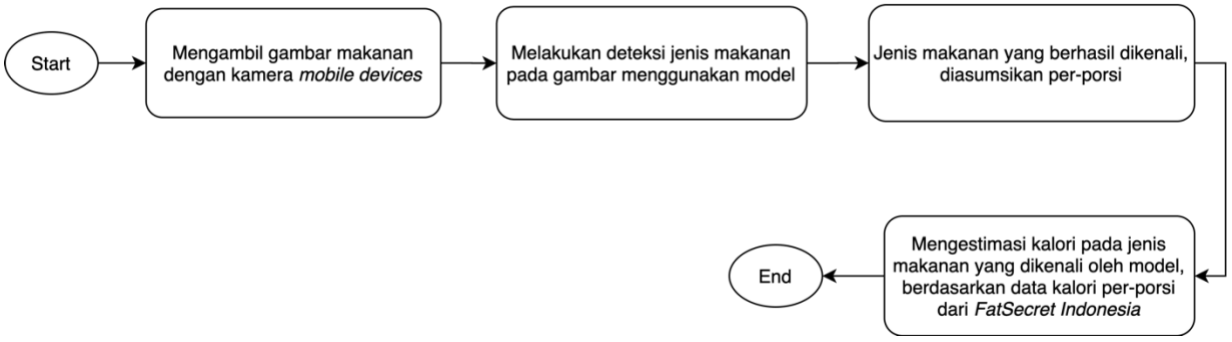
*Macro Average* adalah perhitungan nilai rata-rata yang dalam penelitian ini akan dihitung rata-rata dari presisi, *recall*, dan *F1-Scores*.

***Weighted Average***

*Weighted Average akan* menambahkan jumlah F1-Scores dengan jumlah objek yang di tes (*support*), pada penelitian ini juga dilakukan dengan menghitung *weighted-precision* dan *weighted-recall*.

## 3.2 Perancangan *Mobile Application* untuk mengestimasi kalori pada makanan

Seperti yang tertulis di batasan masalah, bahwa pada penelitian ini, kalori akan diestimasi menggunakan asumsi kalori per porsi berdasarkan data dari *FatSecret Indonesia.* Model *SSD* yang dibangun di penelitian ini akan digunakan di *mobile application* sebagai pendeteksi jenis makanan. Setelah makanan berhasil dikenali oleh model, model akan memberikan jenis-jenis makanan yang berhasil dikenali tersebut untuk di ambil data kalori per-porsinya di *FatSecret Indonesia*, untuk dapat diestimasi kalorinya. *Flowchart* alur estimasi kalori makanan digambarkan pada Gambar 3.12

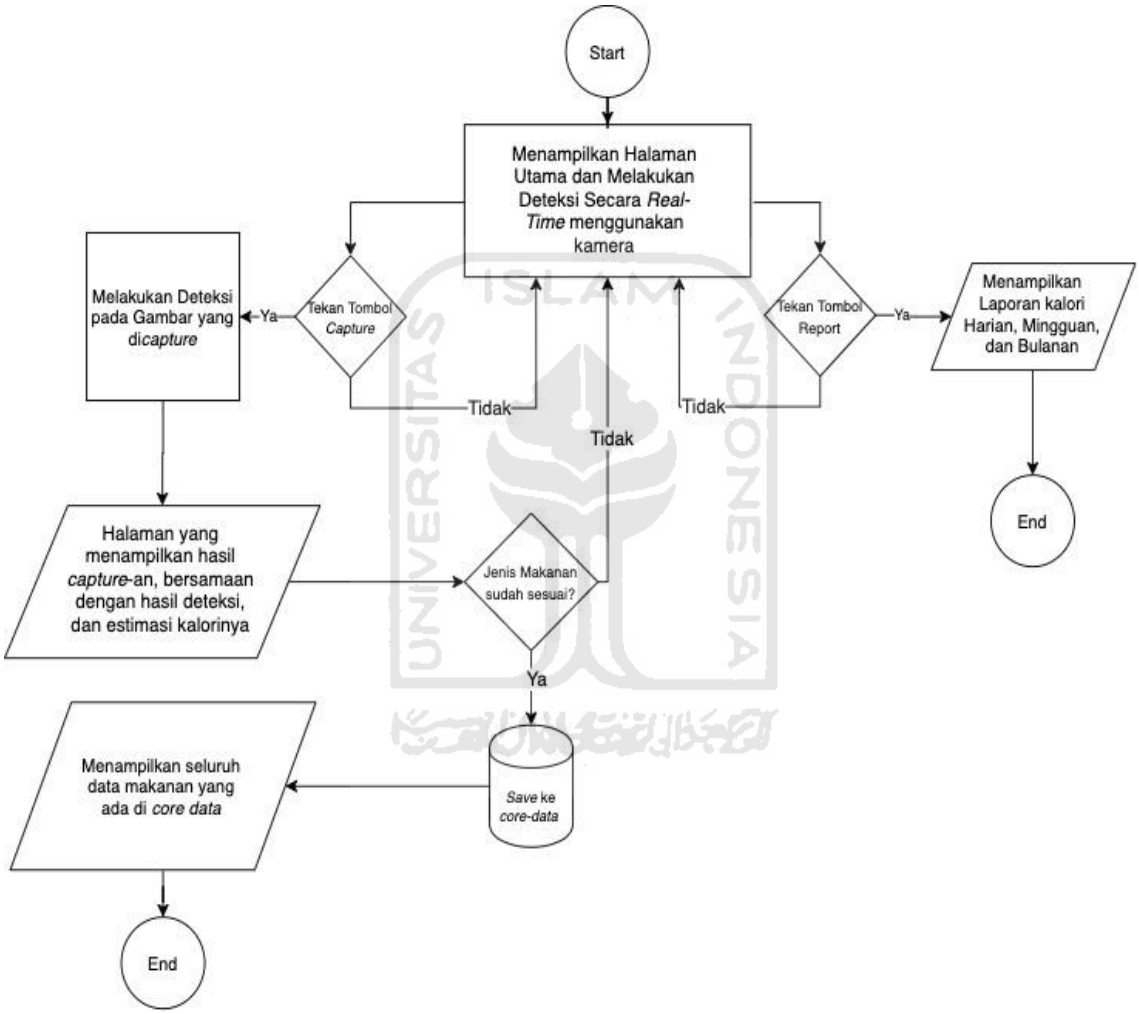



Gambar 3.12 *Flowchart* alur estimasi kalori makanan

Pada perancangan *mobile application*, bertujuan untuk membangun aplikasi *mobile* yang dapat menjalankan model *deep learning* yang sudah di-*training* untuk mengenali makanan.

Lalu, *mobile application* harus dapat melakukan estimasi kalori, menyimpan, dan melihat data makanan yang dimakan oleh *user* beserta estimasi kalorinya.

*Mobile Application* juga akan mampu melihat asupan kalori harian, mingguan, dan bulanan, yang dikonsumsi oleh *user*. *Mobile Application* pun dapat menghitung kalori yang dibakar dengan kalkulasi langkah yang dilakukan. Alur utama penggunaan *mobile application* sendiri digambarkan di Gambar 3.13.




Gambar 3.13 *Flowchart* pengguna *mobile application*

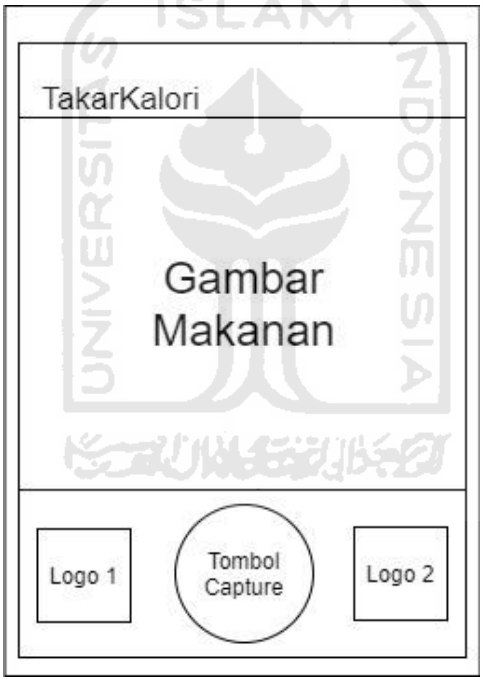
## 3.3 Perancangan Antarmuka

Perancangan antarmuka merupakan bagian yang akan berinteraksi dan dilihat oleh para *user mobile application* ketika melakukan estimasi kalori.

a. Rancangan Halaman Utama

Gambar 3.14 merupakan tampilan halaman utama aplikasi, juga yang akan *user mobile application* temukan saat pertama kali membuka aplikasi. Di halaman ini, aplikasi akan langsung mengambil *input* dari kamera dan diberikan ke dalam model untuk dideteksi. Hasil deteksi dari model akan ditampilkan secara *real-time* di bagian 'gambar makanan'.

Di halaman ini juga, terdapat satu tombol *capture* yang berfungsi untuk *user* melakukan deteksi kalori apabila hasil deteksi model secara *real-time* sudah dapat mengenali makanan dari kamera. Tombol lain yang ada di halaman ini ditampilkan di 'Logo 1' dan 'Logo 2', dimana 'logo 1' dapat digunakan oleh user untuk mengecek riwayat makanan yang sudah dideteksi oleh model. 'Logo 2' dapat ditekan oleh user untuk masuk ke halaman 'Statistik Kalori'.




Gambar 3.14 Tampilan Halaman Utama Aplikasi

b. Rancangan Halaman Estimasi Kalori

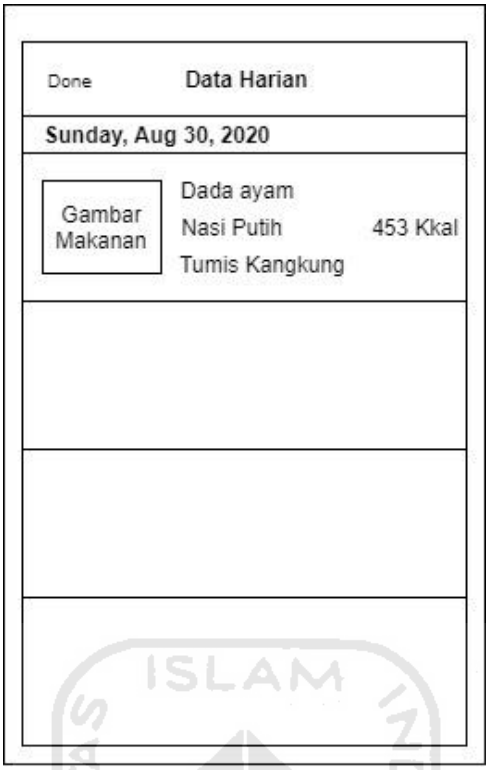
Setelah model sudah mampu mengenali makanan secara *real-time* di halaman utama, user dapat menekan 'Tombol *Capture'*, dan aplikasi akan melakukan pengambilan gambar. Gambar yang diambil itu, akan dimasukkan kembali ke dalam model dan akan dideteksi jenis makanannya. Lalu, aplikasi akan menampilkan halaman Estimasi Kalori, seperti yang ditampilkan di Gambar 3.15


Gambar 3.15 Rancangan Halaman Estimasi Kalori

Di halaman ini, *user* dapat melihat jenis makanan yang sudah berhasil dikenali oleh model beserta asupan kalori per-porsinya. Terdapat juga tombol 'Deteksi Ulang' apabila user ingin melakukan deteksi ulang & 'Simpan Hasil' apabila jenis makanan sudah dirasa sesuai.

c. Rancangan Halaman Data Harian

Halaman data harian merupakan halaman yang menampilkan makanan-makanan yang disimpan oleh *user* setelah melakukan estimasi kalori. Halaman ini akan menampilkan jenis makanan dan total kalori yang sudah dikonsumsi oleh *user mobile application*. Ditampilkan di




Gambar 3.16 Rancangan Halaman Data Harian

d. Rancangan Halaman Statistik Kalori (Penghitung kalori dalam tubuh)

Halaman ini merupakan halaman pendukung di dalam aplikasi. Halaman yang menampilkan data kalori yang sudah dikonsumsi oleh *user mobile application*, dalam rentang hari, minggu, hingga bulan. Halaman ini juga menampilkan defisit kalori masuk (dari makanan) dan kalori keluar (dari langkah kaki) yang diambil menggunakan bantuan aplikasi *HealthKit* milik Apple. Tampilan halaman ditampilkan pada Gambar 3.17


Gambar 3.17 Rancangan Halaman Statistik Kalori

### 3.4 Pengujian Estimasi Kalori pada Aplikasi dengan Nilai Kalori Sebenarnya

Pada tahapan ini, penulis akan melakukan pengujian estimasi kalori pada aplikasi yang sudah dibangun. Pengujian yang dilakukan adalah dengan membandingkan estimasi kalori yang diberikan oleh aplikasi dengan nilai kalori sebenarnya. Dalam hal ini, nilai kalori sebenarnya mengacu kepada data kalori yang diambil di "*FatSecret Indonesia*".

Terkait teknis pengujian, aplikasi akan diberikan 10 gambar (citra) makanan, dan aplikasi akan melakukan estimasi kalori pada gambar-gambar tersebut, untuk menyimpulkan apakah aplikasi sudah dapat melakukan estimasi pada kalori makanan menggunakan citra. Tabel pengujian yang akan digunakan di tahapan ini ditampilkan di Tabel 3.6

Tabel 3.6 Contoh tabel pengujian estimasi kalori

| **No** | **Gambar Makanan** | **Kalori pada makanan (berdasarkan *FatSecret Indonesia*)** | |
|---|---|---|---|
| 1 | | Telur Dadar | 103 Kal |
| | | Nasi Putih | 100 Kal |
| | **Hasil Deteksi** | **Estimasi Kalori pada aplikasi** | |
| | | Nasi Putih | 100 Kal |
| | | Telur Dadar | 103 Kal |

# BAB IV
# HASIL DAN PEMBAHASAN

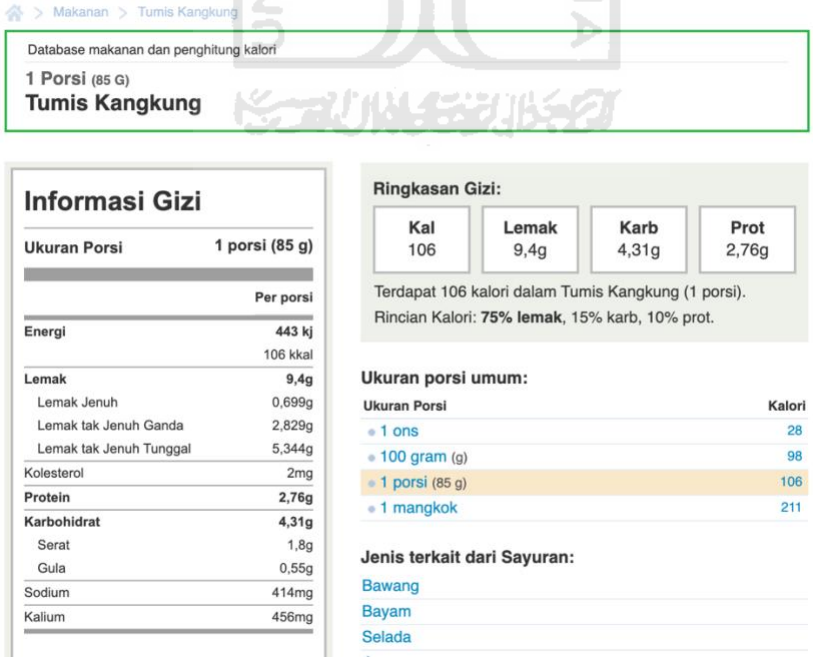
## 4.1 Implementasi

Implementasi merupakan tahap selanjutnya dari tahap perancangan yang sudah direncanakan di tahapan sebelumnya. Pada implementasi ini dilakukan tahapan-tahapan yang sebelumnya sudah dirancang, yaitu: Pengambilan Data, *Pre-Processing*, *Training, Testing,* sampai ke pengujian estimasi kalori pada *mobile application*.

### 4.1.1 Implementasi Pengambilan Data

**Pengambilan Data Kalori**

Seperti yang sudah disampaikan di Subbab 3.1.1, data kalori yang digunakan untuk estimasi kalori pada penelitian ini menggunakan asumsi porsi makanan orang Indonesia, karena itu penelitian ini mengacu pada data yang diberikan oleh *FatSecret Indonesia*. Data kalorinya sendiri bisa diakses di halaman *website* https://www.fatsecret.co.id/kalori-gizi/. Tampilan informasi dari *website FatSecret* Indonesia ditunjukkan di Gambar 4.1




Gambar 4.1 Tampilan Halaman Website *FatSecret Indonesia*

Mengacu pada data kelas yang sudah ditentukan sebelumnya di Tabel 3.1, sehingga data kalori yang dikumpulkan dari *FatSecret Indonesia*, dapat ditampilkan di Tabel 4.1

Tabel 4.1 Data Kalori yang Digunakan

| Nama Makanan | Porsi (Berat) | Estimasi Kalori |
|---|---|---|
| Telur Dadar | 1 Telur | 135 Kal |
| Nasi Putih | 1 Porsi (135 g) | 135 Kal |
| Ikan Lele Goreng | 1 Porsi (85 g) | 204 Kal |
| Tumis Kangkung | 1 Porsi (85 g) | 106 Kal |
| Dada Ayam Goreng | 1 Porsi (98 g) | 212 Kal |
| Tempe Goreng | 1 Buah | 34 Kal |
| Tahu Goreng | 1 Buah | 35 Kal |
| Kerupuk Putih | 1 Buah | 65 Kal |
| Sambal | 1 Porsi | 102 Kal |

**Pengambilan Data Gambar**

Pada implementasi pengambilan data gambar ini, dilakukan dengan menggunakan *search engine*. Penelitian ini juga menggunakan sedikit gambar yang diambil menggunakan *mobile phone*, juga beberapa gambar yang mengandung 2-3 objek sekaligus dalam satu gambar. Populasi gambar yang berhasil dikumpulkan sebagian besar ditampilkan di Gambar 4.2

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

Gambar 4.2 Contoh gambar yang diambil dari internet, (a) kerupuk Putih, (b) dada ayam, (c) tahu goreng, (d) tempe goreng, (e) lele goreng, (f) tumis kangkung, (h) telur dadar, (i) sambal, (j) tumis kangkung. Contoh gambar yang diambil menggunakan kamera handphone, (g) nasi putih, (k) lele goreng, (l) nasi putih, telur dadar, dan tumis kangkung

Total gambar yang berhasil dikumpulkan dari internet menggunakan beberapa ketentuan yang sudah dituliskan di Subbab 3.1.1 adalah 202 gambar, untuk sembilan kelas yang dalam tiap gambarnya bisa mengandung lebih dari satu objek dan satu kelas. Sebagian kecil gambar juga diambil menggunakan kamera *handphone*. Data objek yang berhasil dikumpulkan ditampilkan pada Tabel 4.2

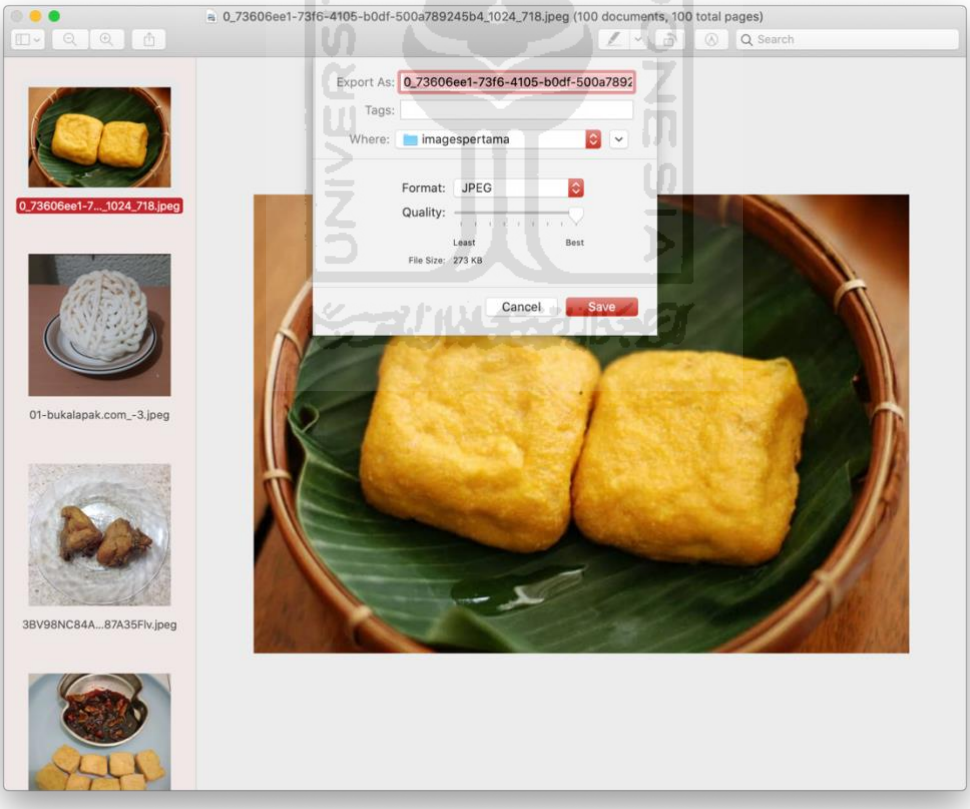
Tabel 4.2 Jumlah Objek Dari Gambar yang dikumpulkan

| **Nama Objek** | **Jumlah** |
|---|---|
| Telur Dadar | 41 |
| Nasi Putih | 42 |
| Ikan Lele Goreng | 41 |

| Nama Objek | Jumlah |
|---|---|
| Tumis Kangkung | 41 |
| Dada Ayam Goreng | 41 |
| Tempe Goreng | 41 |
| Tahu Goreng | 48 |
| Kerupuk Putih | 41 |
| Sambal | 42 |

### 4.1.2 Implementasi Mengganti Tipe File

Semua file yang sudah diambil di internet dan yang menggunakan kamera *handphone*, dikumpulkan ke dalam satu folder. Setelah itu masukkan seluruh gambar tersebut ke aplikasi *Preview* yang ada di sistem operasi Mac OS. Selanjutnya, *export* semua gambar tersebut ke dalam satu format "JPEG", seperti yang ditampilkan pada Gambar 4.3.

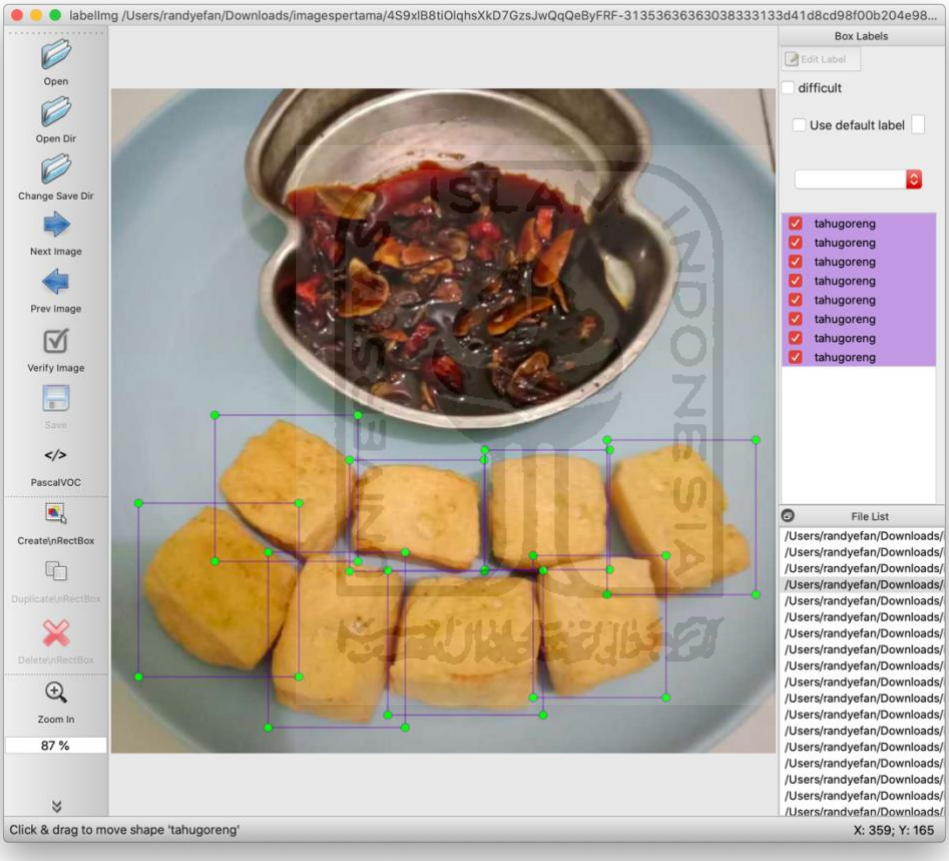



Gambar 4.3 Proses mengubah tipe file gambar

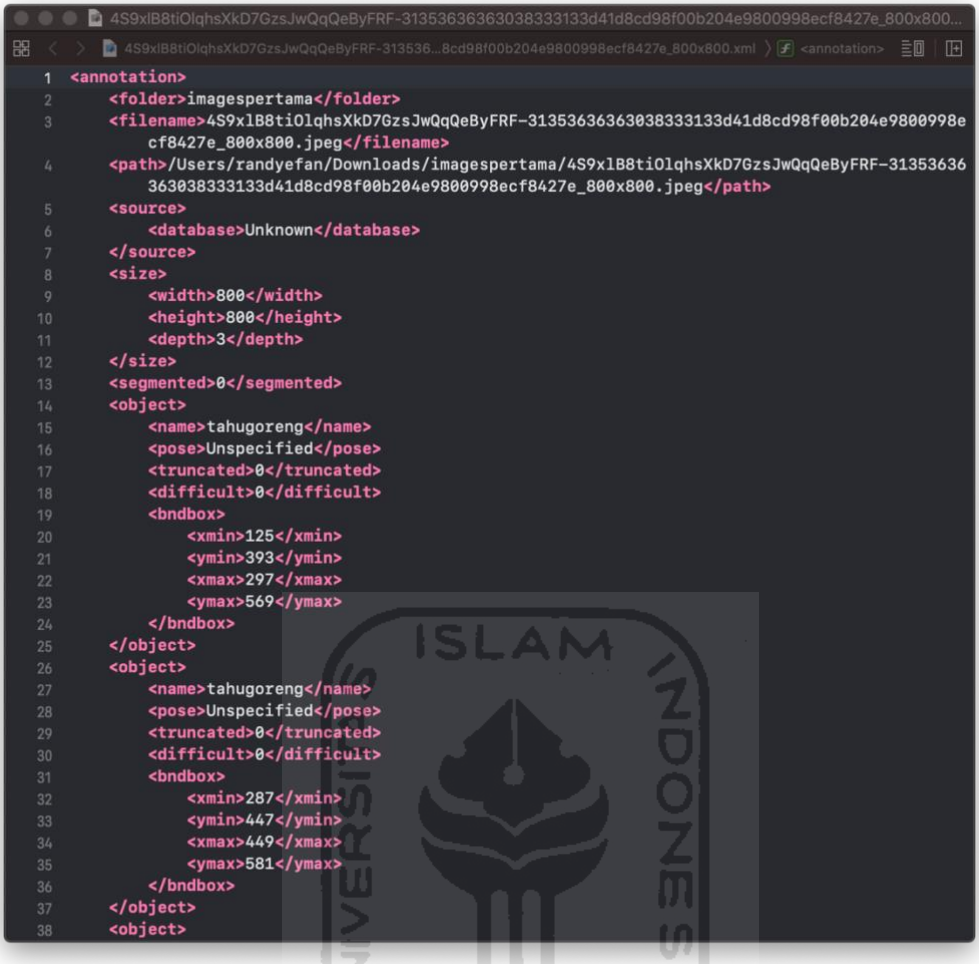
### 4.1.3 Implementasi Anotasi (Pelabelan) Objek

Tahapan ini akan dilakukan dengan memberi label nama objek dan *bounding box* pada tiap objek seperti yang ditampilkan di Gambar 4.4, sehingga menghasilkan file berekstensi .xml

Di dalam *file* berekstensi .xml itu, terdapat informasi-informasi dari gambar, seperti nama *file*, kelas dari objek-objek yang ada di dalam gambar, beserta lokasi *bounding box*. Format anotasi yang digunakan di dalam penelitian ini adalah format "PascalVOC", salah satu contohnya ditampilkan di Gambar 4.5 .




Gambar 4.4 Implementasi Anotasi Gambar



```xml
<annotation>
    <folder>imagespertama</folder>
    <filename>4S9xlB8tiOlqhsXkD7GzsJwQqQeByFRF-3135363636303833313 3d41d8cd98f00b204e9800998ecf8427e_800x800.jpeg</filename>
    <path>/Users/randyefan/Downloads/imagespertama/4S9xlB8tiOlqhsXkD7GzsJwQqQeByFRF-31353636363038333133d41d8cd98f00b204e9800998ecf8427e_800x800.jpeg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>800</width>
        <height>800</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>tahugoreng</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>125</xmin>
            <ymin>393</ymin>
            <xmax>297</xmax>
            <ymax>569</ymax>
        </bndbox>
    </object>
    <object>
        <name>tahugoreng</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>287</xmin>
            <ymin>447</ymin>
            <xmax>449</xmax>
            <ymax>581</ymax>
        </bndbox>
    </object>
    <object>
```

Gambar 4.5 File xml setelah dilakukan anotasi gambar

### 4.1.4 Implementasi Augmentasi pada Gambar

Setelah seluruh gambar dianotasi berdasarkan objek yang ada di dalam gambar, proses augmentasi dilakukan pada gambar tersebut. Proses augmentasi berfungsi untuk memperbanyak variasi dari gambar. Implementasi augmentasi gambar ini akan menggunakan *library* "imgAug" di bahasa pemrograman *Python*, dan akan menggunakan bantuan beberapa *package Python* lain, seperti Pandas, Xml.etree.ElementTree, dan lain-lain.

Pada penelitian ini, semua proses dari augmentasi dilakukan di Google Colaboratory. Google Colaboratory akan tersambung dengan Google Drive, oleh karena itu seluruh gambar yang sudah dikumpulkan dan dianotasi *bounding box* (.xml) nya di-*upload* ke google drive agar bisa dilakukan proses augmentasi.

Setelah semua gambar berhasil diunggah, sebelum memperbanyak gambar menggunakan augmentasi, semua gambar yang sudah dikumpulkan akan dilakukan dulu proses *resize image*.

Proses *resize image* dilakukan dengan ketentuan apabila gambar tingginya lebih dari 600, maka akan di-*resize height*-nya ke 600 dan apabila lebarnya lebih dari 600, maka akan di *resize width*-nya ke 600 juga. Lalu, apabila *width* atau *height* pada gambar kurang dari 600, maka tidak di-*resize width* atau *height*nya. Hal ini dilakukan untuk mempermudah model nantinya ketika memproses gambar saat *training* dilakukan.

Lalu masuk ke proses duplikasi gambar menggunakan augmentasi, macam-macam teknik augmentasi yang digunakan pada penelitian ini telah ditampilkan di Tabel 3.2. Salah satu implementasi kode augmentasi ditampilkan di dalam Gambar 4.6. Sintaks tersebut menggunakan sintaks 'OneOf' di dalam *library imgAug* yang berarti menjalankan augmentasi di dalam nya pada seluruh gambar yang melewati augmentasi ini.

```
aug = iaa.OneOf([
                  iaa.CoarseDropout(0.01, size_percent=0.5)
])
```

Gambar 4.6 Kode program untuk *set* teknik augmentasi yang digunakan

Sehingga dibuatlah sebuah fungsi untuk melakukan augmentasi, yang ditunjukkan pada Gambar 4.7

```
def image_aug(df, images_path, aug_images_path, image_prefix,
augmentor):
    aug_bbs_xy = pd.DataFrame(columns=
                              ['filename','width','height','class',
'xmin', 'ymin', 'xmax', 'ymax']
                             )
    grouped = df.groupby('filename')

    for filename in df['filename'].unique():
        group_df = grouped.get_group(filename)
        group_df = group_df.reset_index()
        group_df = group_df.drop(['index'], axis=1)
        image = imageio.imread(images_path+filename)
        bb_array = group_df.drop(['filename', 'width', 'height',
'class'], axis=1).values

        bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array,
shape=image.shape)
        image_aug, bbs_aug = augmentor(image=image,
bounding_boxes=bbs)
        bbs_aug = bbs_aug.remove_out_of_image()
        bbs_aug = bbs_aug.clip_out_of_image()

        if re.findall('Image...', str(bbs_aug)) == ['Image([]']:
            pass
        else:
            imageio.imwrite(aug_images_path+image_prefix+filename,
image_aug)
            info_df = group_df.drop(['xmin', 'ymin', 'xmax',
'ymax'], axis=1)
            for index, _ in info_df.iterrows():
                info_df.at[index, 'width'] = image_aug.shape[1]
                info_df.at[index, 'height'] = image_aug.shape[0]
            info_df['filename'] = info_df['filename'].apply(lambda
x: image_prefix+filename)
            bbs_df = bbs_obj_to_df(bbs_aug)
            aug_df = pd.concat([info_df, bbs_df], axis=1)
            aug_bbs_xy = pd.concat([aug_bbs_xy, aug_df])
    return aug_bbs_xy
```

Gambar 4.7 Kode Program untuk melakukan augmentasi

Setelah semua gambar dilakukan augmentasi dengan kode program di Gambar 4.7, dan diulang untuk setiap jenis augmentasi dan *value* yang sudah ditentukan, semua informasi di dalam gambar yang baru diaugmentasi akan disimpan ke dalam *Data Frame*. *Data Frame* sendiri adalah struktur data di dalam *Library Pandas*. Lalu, selanjutnya, menggabungkan *Data Frame* ketika melakukan *resizing image* dan *augmentasi* menjadi satu *Data Frame* dan menjadikannya file berekstensi .csv, menggunakan kode program di Gambar 4.8

```
all_labels_df = pd.concat([resized_images_df, augmented_images_df])
all_labels_df.to_csv('all_labels.csv', index=false)
```

Gambar 4.8 Kode program untuk menyimpan dataframe augmentasi ke csv

Setelah data gambar diaugmentasi, dataset gambar bertambah menjadi total 5252 hasil gambar yang dikumpulkan, ditambah dengan hasil augmentasi, dengan rincian tiap kelas (1 gambar bisa mengandung beberapa objek berbeda) bisa dilihat pada Tabel 4.3

Tabel 4.3 Data objek pada gambar setelah augmentasi

| **Nama Objek** | **Jumlah** |
|---|---|
| Telur Dadar | 1066 |
| Nasi Putih | 1092 |
| Ikan Lele Goreng | 1096 |
| Tumis Kangkung | 1066 |
| Dada Ayam Goreng | 1066 |
| Tempe Goreng | 1066 |
| Tahu Goreng | 1248 |
| Kerupuk Putih | 1066 |
| Sambal | 1092 |

### 4.1.5 Implementasi *Split Dataset*

Setelah seluruh gambar berhasil diaugmentasi, sebelum masuk ke *training model,* data harus dibagi menjadi dua bagian, data *train* dan data *test*. Data gambar akan dibagi ke dalam dua bagian, begitu juga dengan informasi dari gambar (.csv) tersebut.

### *Split* Data Gambar

Total gambar setelah diaugmentasi berjumlah 5252 Gambar dan 5252 gambar tersebut akan dibagi menjadi dua bagian, *test data* & *train data*. Gambar akan dibagi dengan rincian pembagian kurang lebih 20:80, kurang lebih 20% untuk *test data* & 80% untuk *train data*. Pembagian pada gambar akan dilakukan acak, sehingga data *test* berjumlah 997 gambar, dan data train terdapat 4255 gambar. Untuk kode program pembagiannya sendiri bisa dilihat pada kode program yang ditampilkan di Gambar 4.9

```
mkdir test_labels train_labels

ls *jpeg | sort -R | head -997 | xargs -I{} mv{} test_labels/

ls *jpeg | xargs -I{} mv{} train_labels/
```

Gambar 4.9 Kode program *split* data gambar

### *Split* Data ke dalam .csv

Setelah seluruh data gambar sudah berada di dalam folder masing-masing berdasarkan data *test* & *train*, selanjutnya adalah membuat file berekstensi .csv yang berisi nama objek yang ada dalam gambar dan lokasi *bounding box* tiap gambar pada *Test set* dan *Train set*. Cara yang digunakan di tahapan ini adalah dengan menuliskan fungsi untuk mengecek nama file gambar dari tiap folder. Setelah itu, penulis mengambil informasi objek dari file tersebut yang tersimpan di file "all_labels.csv" yang sudah dibuat pada Gambar 4.8. Kode program ditampilkan pada Gambar 4.10 dan Gambar 4.11

```
%cd /gdrive/My Drive/images/test_labels

labels_df = pd.read_csv('/gdrive/My Drive/images/all_labels.csv')
newcsv = pd.DataFrame(columns= ['filename','width','height','class',
'xmin', 'ymin', 'xmax', 'ymax'])
newcsv.to_csv('test_labels.csv', index=None)

grouped = labels_df.groupby('filename')

for i, index in enumerate(glob.glob('*.jpeg')):
  labels_df = grouped.get_group(index)
  labels_df = labels_df.reset_index()
  labels_df = labels_df.drop(['index'], axis=1)

  labels_df.to_csv('test_labels.csv', mode='a', header=False,
index=None)
```

Gambar 4.10 Kode program untuk membuat file .csv dari data test

```
%cd /gdrive/My Drive/images/train_labels


labels_df = pd.read_csv('/gdrive/My Drive/images/all_labels.csv')

newcsv = pd.DataFrame(columns= ['filename','width','height','class',
'xmin', 'ymin', 'xmax', 'ymax'])
newcsv.to_csv('train_labels.csv', index=None)

grouped = labels_df.groupby('filename')

for i, index in enumerate(glob.glob('*.jpeg')):
  labels_df = grouped.get_group(index)
  labels_df = labels_df.reset_index()
  labels_df = labels_df.drop(['index'], axis=1)

  labels_df.to_csv('train_labels.csv', mode='a', header=False,
index=None)
```

Gambar 4.11 Kode program untuk membuat file .csv dari data train

Pada Tabel 4.4 ditampilkan daftar jumlah objek untuk data *train* dan data *test*, beberapa objek makanan pada *training set* diseimbangkan dengan tujuan agar model mampu mengenali semua kelas (objek makanan) dengan baik.

Tabel 4.4 Daftar Jumlah Objek untuk data train dan data test

| No | Nama Kelas | Jumlah Objek | |
|---|---|---|---|
| | | ***Training Set*** | ***Test Set*** |
| 1 | Dada Ayam Goreng | 878 | 188 |
| 2 | Kerupuk Putih | 878 | 188 |
| 3 | Nasi Putih | 878 | 214 |
| 4 | Tumis Kangkung | 878 | 214 |
| 5 | Tahu Goreng | 878 | 370 |
| 6 | Tempe Goreng | 879 | 187 |
| 7 | Sambal | 878 | 214 |
| 8 | Lele Goreng | 878 | 188 |
| 9 | Telur Dadar | 877 | 189 |

### 4.1.6 *Training* Model

*Training* akan dilakukan menggunakan aplikasi "TeamViewer" yang berguna untuk melakukan remote komputer di Lab Informatika UII, dan memanfaatkan Tensorflow-gpu yang ada di sana.

Pada implementasi *training*, langkah pertama yang harus dilakukan adalah men-*generate* file *tfrecord*. File *tfrecord* merupakan file yang akan dibaca oleh Tensorflow ketika melakukan *training*, untuk mengarahkan lokasi data *test* dan data *train* yang dimiliki, dan juga lokasi *ground-truth bounding box* dari setiap gambar di dataset. Dibuatlah sebuah fungsi seperti yang ditampilkan pada Gambar 4.12 dan Gambar 4.13

```
data_base_url = 'c:/kalori/data/'

#location of images
image_dir = data_base_url +'images/'

def class_text_to_int(row_label):
  if row_label == 'dadaayam':
    return 1
  elif row_label == 'kerupukputih':
    return 2
  elif row_label == 'lelegoreng':
    return 3
  elif row_label == 'nasiputih':
    return 4
  elif row_label == ‘sambal’:
    return 5
  elif row_label == 'tahugoreng':
    return 6
  elif row_label == 'telurdadar':
    return 7
  elif row_label == 'tempegoreng':
    return 8
  elif row_label == 'tumiskangkung':
    return 9
  else:
    None

def split(df, group):
  data = namedtuple('data', ['filename', 'object'])
  gb = df.groupby(group)
  return [data(filename, gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]

def create_tf_example(group, path):
  with tf.io.gfile.GFile(os.path.join(path, '{}'.format(group.filename)),
'rb') as fid:
    encoded_jpg = fid.read()
  encoded_jpg_io = io.BytesIO(encoded_jpg)
  image = Image.open(encoded_jpg_io)
  width, height = image.size
  filename = group.filename.encode('utf8')
  image_format = b'jpeg'
```

Gambar 4.12 Kode Program generate TFRecord bagian 1

```
        classes_text = []
        classes = []

        for index, row in group.object.iterrows():
            xmins.append(row['xmin'] / width)
            xmaxs.append(row['xmax'] / width)
            ymins.append(row['ymin'] / height)
            ymaxs.append(row['ymax'] / height)
            classes_text.append(row['class'].encode('utf8'))
            classes.append(class_text_to_int(row['class']))

        tf_example = tf.train.Example(features=tf.train.Features(feature={
            'image/height': dataset_util.int64_feature(height),
            'image/width': dataset_util.int64_feature(width),
            'image/filename': dataset_util.bytes_feature(filename),
            'image/source_id': dataset_util.bytes_feature(filename),
            'image/encoded': dataset_util.bytes_feature(encoded_jpg),
            'image/format': dataset_util.bytes_feature(image_format),
            'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
            'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
            'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
            'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
            'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
            'image/object/class/label': dataset_util.int64_list_feature(classes),
            }))
        return tf_example
    #creates tfrecord for both csv's
    for csv in ['train_labels', 'test_labels']:
      writer = tf.io.TFRecordWriter(data_base_url + csv + '.record')
      path = os.path.join(image_dir)
      examples = pd.read_csv(data_base_url + csv + '.csv')
      grouped = split(examples, 'filename')
      for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

      writer.close()
      output_path = os.path.join(os.getcwd(), data_base_url + csv + '.record')
      print('Successfully created the TFRecords: {}'.format(data_base_url +csv +
'.record'))
```

Gambar 4.13 Kode Program generate TFRecord bagian 2

Metode *training* ini akan menggunakan arsitektur "ssd-mobilenet-V2" yang sudah tersedia di Tensorflow. Oleh karena itu, ada beberapa *hyperparameter* yang harus disesuaikan, agar model bisa di-*training* dan bekerja sesuai dengan yang diinginkan. Untuk beberapa perubahan *hyperparameter* yang sebelumnya adalah *default*, ditampilkan pada Tabel 4.5

Tabel 4.5 Set *hyperparameter* sebelum *training*

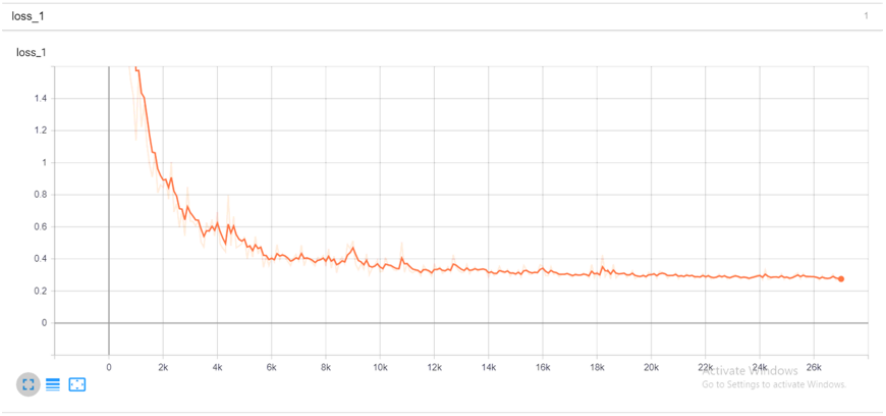
| ***Hyperparameter*** | ***Value*** |
|---|---|
| *Input Image* | 300x300 |
| *Use_Dropout* | *True* |
| *Learning Rate* | 0.004 |
| *Batch Size* | 22 |
| *Num steps* | 400.000 |
| *Dropout value* | 0.8 |
| *Num classes* | 9 |

Setelah *hyperparameter* sudah disesuaikan, maka *training* bisa dimulai pada *dataset* menggunakan kode program yang ditampilkan pada Gambar 4.14

```
python object_detection/model_main.py --
pipeline_config_path=C:\kalori\models\research\object_detection\sampl
es\configs\ssd_mobilenet_v2_coco.config --model_dir=training/ --
alsologtostderr
```

Gambar 4.14 Kode program untuk memulai *training* pada dataset

Jalannya *training model* pada dataset, dipantau menggunakan *tensorboard*, agar bisa dipantau nilai *loss*, yang dihasilkan selama proses *training* berlangsung. Lalu, setelah *step* 18000, nilai *total loss* yang dihasilkan dari proses *training* pada dataset sudah menunjukkan nilai yang cukup rendah yaitu, 0.36 *graph loss* selama *training,* ditunjukkan pada Gambar 4.15

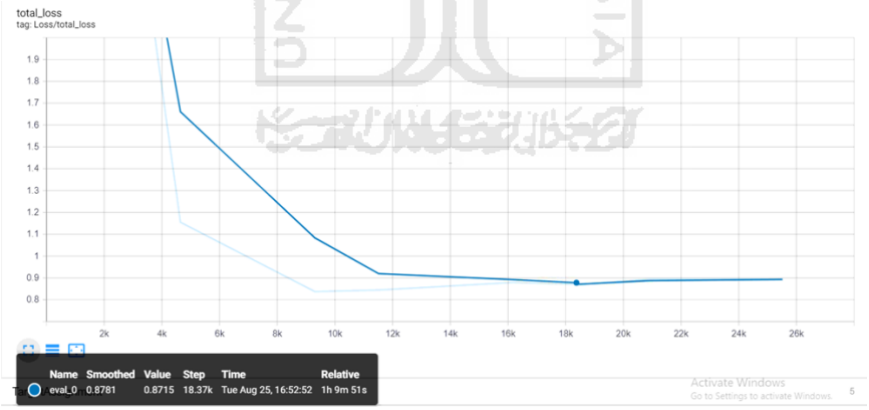



Gambar 4.15 Graph *loss* model pada *training set*

Nilai *loss* yang sudah tidak akan mengalami penurunan lagi membuat *training* model sudah harus dicukupkan, oleh karena itu, penulis memilih untuk menghentikan proses *training* yang sedang berlangsung.

### 4.1.7 *Testing* Model

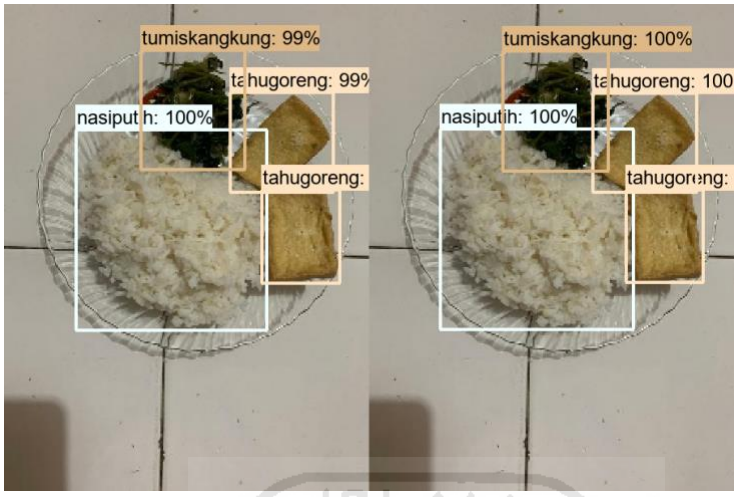
*Testing* Model ini adalah tahapan menguji model dengan memberikan data yang sudah disiapkan (*test set*), data yang belum pernah di-*training* oleh model, lalu melihat bagaimana model melakukan deteksi kelas, dan lokasi *bounding box*.




Gambar 4.16 Graph *loss* model pada *testing set*

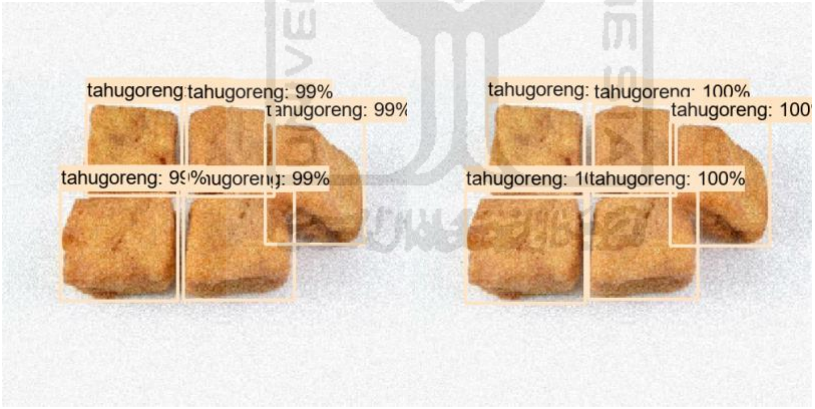
Seperti yang ditampilkan di Gambar 4.16, merupakan performa model setelah 18000 *step training* dilakukan dengan *value loss*nya adalah 0.87. Dari sini kita lihat dari *loss* yang didapatkan berdasarkan *training set* dan *test set,* ada sedikit kecenderungan *overfitting.* Performa model pada gambar pun sebagian dapat ditunjukkan di Gambar 4.17, Gambar 4.18, Gambar 4.19, Gambar 4.20, Gambar 4.21, Gambar 4.22, Gambar 4.23, Gambar 4.24 dan

Gambar 4.25 dimana gambar **(a)** merupakan hasil *detection* dari model, sedangkan gambar **(b)** merupakan *ground-truth* dari gambar tersebut.




**(a)** **(b)**

Gambar 4.17 Performa model pada step 18 K.

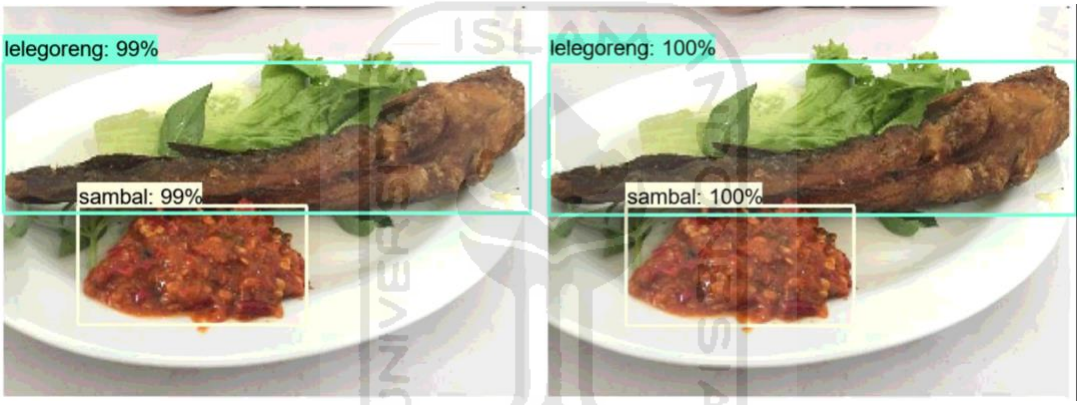



**(a)** **(b)**

Gambar 4.18 Performa model pada step 18 K


**(a)** **(b)**

Gambar 4.19 Performa model pada step 18 K

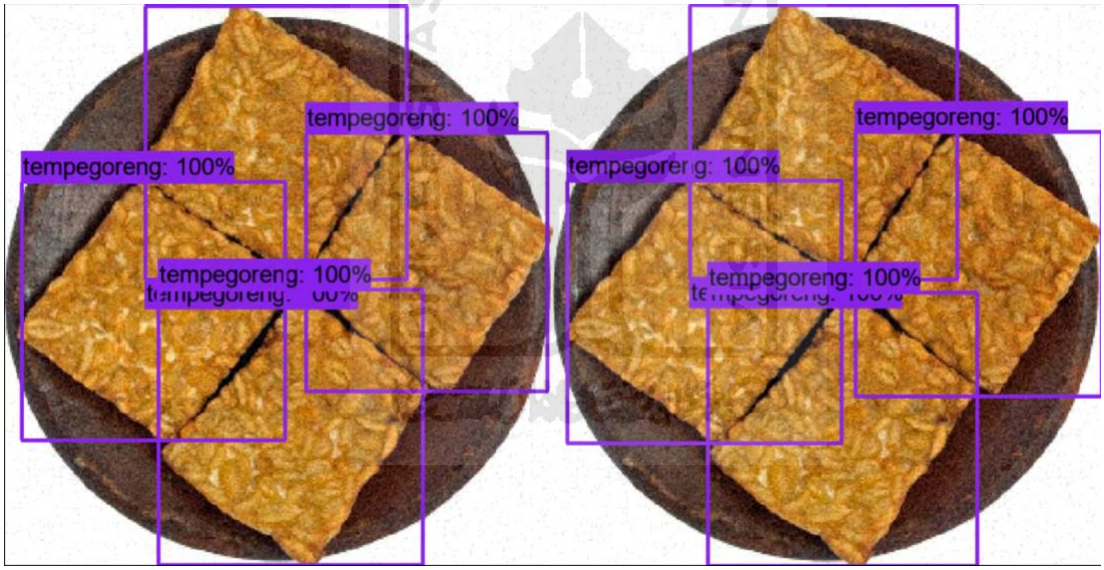



**(a)** **(b)**

Gambar 4.20 Performa model pada step 18 K

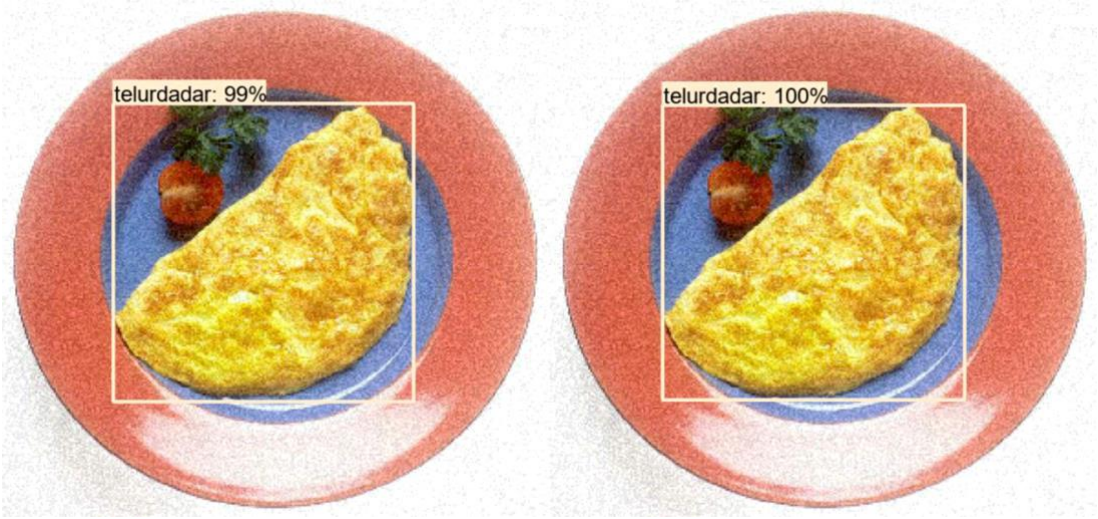

**(a)** **(b)**

Gambar 4.21 Performa model pada step 18 K



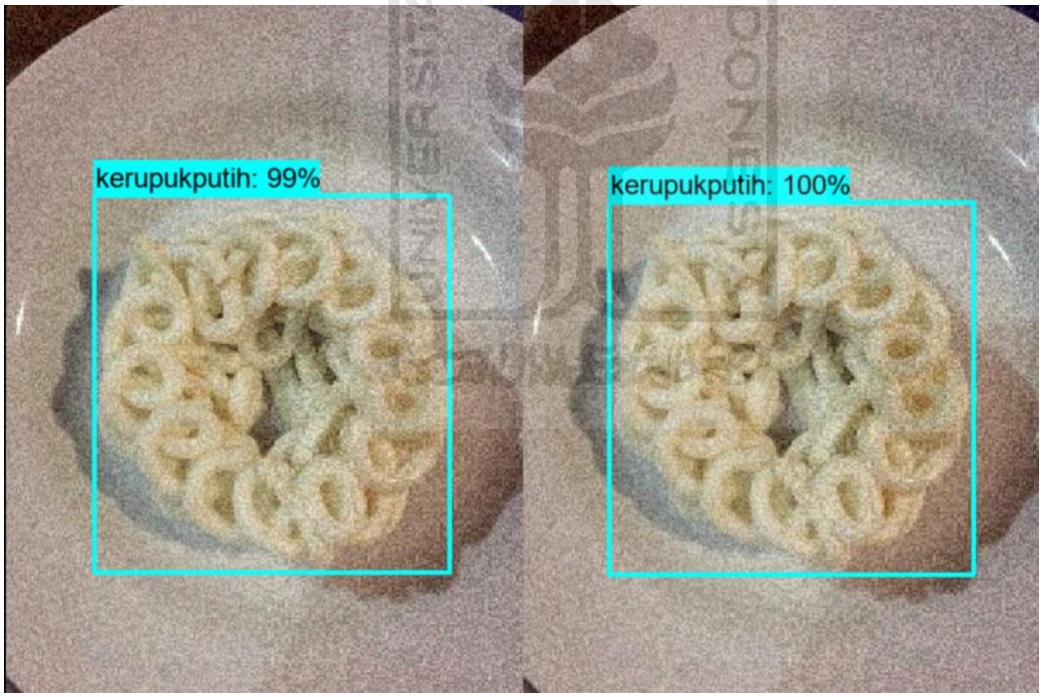

**(a)** **(b)**

Gambar 4.22 Performa model pada step 18 K



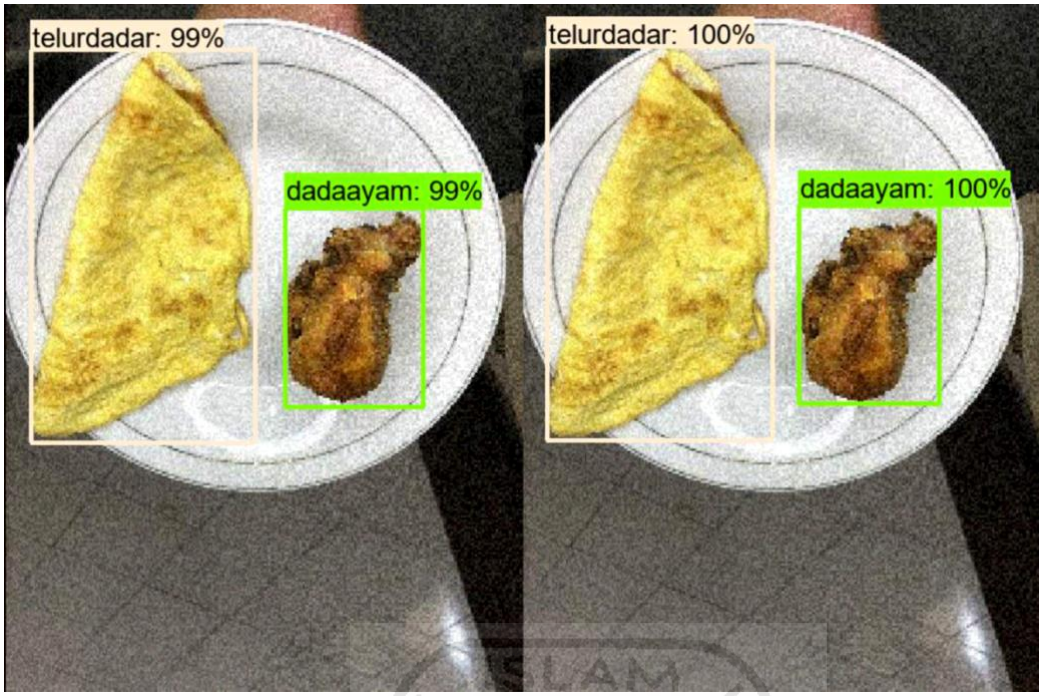

**(a)** **(b)**

Gambar 4.23 Performa model pada step 18 K




**(a)** **(b)**

Gambar 4.24 Performa model pada step 18 K




**(a)** **(b)**

Gambar 4.25 Performa model pada step 18 K

Sehingga dari hasil itu, bisa disimpulkan bahwa model sudah berjalan dengan cukup baik dan sudah siap untuk dibuat *frozen graph*-nya lalu dikonversi ke format .tflite agar bisa diimplementasikan ke *mobile application*.

### 4.1.8 Implementasi *convert* model ke tflite

Tahapan-tahapan yang digunakan untuk mengkonversi model ke tflite pada penelitian ini, pertama, akan membuat *frozen graph* pada *checkpoint* yang ditentukan, yang pada kasus ini, dilakukan pada *checkpoint training* ke *18370*. Kode programnya dapat dilihat pada Gambar 4.26

```
python object_detection/export_tflite_ssd_graph.py \
  --pipeline_config_path=
c:/kalori/models/research/object_detection/samples/configs/ssd_mobilenet_v2_coco
.config \
  --trained_checkpoint_prefix=c:/kalori/models/research/training5/model.ckpt-
18370 \
  --output_directory=tmp/tflite \
  --add postprocessing op=true
```

Gambar 4.26 Kode Program untuk melakukan *frozen graph* pada *checkpoint*

Setelah kode program dijalankan, akan ada 2 file baru yang berlokasi di dalam folder "c:/kalori/models/research/tmp/tflite" yaitu file dengan nama "tflite_graph.pb" & "tflite_graph.pbtxt". Tahapan selanjutnya adalah mengkonversi file "tflite_graph.pb" tersebut menjadi file berformat ".tflite" yang bisa diimplementasikan ke dalam *mobile application*, yang dalam kasus ini diberi nama "food.tflite". Kode program untuk melakukan konversi tersebut bisa dilihat pada Gambar 4.27

```
tflite_convert \
--output_file=c:kalori/models/research/tmp/tflite/food.tflite \
--graph_def_file= c:kalori/models/research/tmp/tflite/tflite_graph.pb
\
--output_format TFLITE \
--input_shape=1,300,300,3 \
--input_arrays=normalized_input_image_tensor \
--
output_arrays='TFLite_Detection_PostProcess','TFLite_Detection_PostPr
ocess:1','TFLite_Detection_PostProcess:2','TFLite_Detection_PostProce
ss:3'  \
--mean_values=128 \
--std_dev_values=128 \
--allow_custom_ops
```

Gambar 4.27 Kode program untuk melakukan konversi ke format .tflite

### 4.1.9 Implementasi Model ke *Mobile Application*

Setelah file food.tflite sudah berhasil dibuat, langkah selanjutnya adalah mengimplementasikan model tersebut ke dalam *mobile application*. *Mobile application* yang dibangun harus memasukkan *third-party library* "TensorflowLite" ke dalam aplikasi. Tujuan dari memasukkan *third-party library* adalah untuk dapat menjalankan file berformat .tflite di dalam *mobile application*.

Pada kasus ini, aplikasi dibangun di *iPhone operating system* (*iOS*), oleh karena itu pembuatan *mobile application* menggunakan tools *Xcode* dan akan menggunakan bahasa pemrograman *native* iOS, yaitu bahasa pemrograman Swift. Aplikasi juga menggunakan *dependency manager* untuk memanfaatkan *third-party library*, yang pada penelitian ini digunakan 'Cocoapods'. Lalu, di dalam file *pods* yang tersedia, ditambakan tulisan "pod TensorFlowLiteSwift" untuk memasukkan *library* tensorflowlite ke dalam aplikasi.

Lalu menambah satu file di dalam *project mobile application*, di penelitian ini diberi nama “ModelDataHandler.swift”. Lalu, penulis meng-*import* modul tensorflowlite di dalam file tersebut, seperti yang ditampilkan pada Gambar 4.28

```
import tensorflowlite
…
…
…
```

Gambar 4.28 Kode Program pada file “ModelDataHandler.swift”

Setelah itu, penulis memasukkan file “makanan3.tflite” yang sudah di-*generate* di tahapan sebelumnya ke dalam *project navigator* yang ada di-xcode seperti ditampilkan pada Gambar 4.30. Penulis juga membuat file “labelmap.txt” yang berisi informasi makanan yang sudah bisa dikenali oleh model, seperti ditampilkan pada Gambar 4.29


Gambar 4.29 Isi file “labelmap.txt”


Gambar 4.30 Tampilan *Project Navigator* di Xcode

Saat semua file dan *third-party library* untuk menjalankan model sudah siap, langkah selanjutnya adalah penulis menyiapkan kode program untuk melakukan *Transforming data*. *Transforming data* dilakukan untuk menyesuaikan ukuran gambar yang diambil oleh *camera* dengan ukuran gambar yang bisa diterima oleh model saat *training*. Pada penelitian ini, proses *resize image* di *training* dan *testing* model sudah dilakukan di dalam *code Tensorflow*. Oleh karena itu, saat kamera *mobile application* ingin melakukan deteksi pada citra, masukan dari kamera harus dikonversi ukurannya menjadi 300x300. Kode program tersebut menggunakan bahasa pemrograman swift yang ditulis di file "ModelDataHandler.Swift", ditampilkan pada Gambar 4.31.

```
…
//Model Parameters
let inputWidth = 300
let inputHeight = 300
…
…

let scaledSize = CGSize(width: inputWidth, height: inputHeight)
   guard let scaledPixelBuffer = pixelBuffer.resized(to: scaledSize)
else {
      return nil
    }
```

Gambar 4.31 Kode Program pada file "ModelHandlerData.swift"

Lalu setelah itu, penulis membuat kode program untuk menjalankan *inference* model pada aplikasi, untuk fungsi lengkap nya bisa dilihat pada Gambar 4.32

```
func runModel(onFrame pixelBuffer: CVPixelBuffer) -> Result? {
   …
   …
   let scaledSize = CGSize(width: inputWidth, height: inputHeight)
   guard let scaledPixelBuffer = pixelBuffer.resized(to: scaledSize)
else {
          return nil
     }
    …
    …
    do {
          let inputTensor = try interpreter.input(at: 0)

          guard let rgbData = rgbDataFromBuffer(
            scaledPixelBuffer,
            byteCount: batchSize * inputWidth * inputHeight *
inputChannels,
            isModelQuantized: inputTensor.dataType == .uInt8
          ) else {
            print("Failed to convert the image buffer to RGB data.")
            return nil
          }

          // Copy the RGB data to the input `Tensor`.
          try interpreter.copy(rgbData, toInputAt: 0)

          // Run inference by invoking the `Interpreter`.
          let startDate = Date()
          try interpreter.invoke()
          interval = Date().timeIntervalSince(startDate) * 1000

          …
          …
   }
```

Gambar 4.32 Kode Program untuk *running infrence model* pada aplikasi

Di dalam *function* tersebut, penulis juga menambahkan *intrepreter* untuk menerjemahkan hasil deteksi dari model yang sudah dijalankan, agar bisa digunakan kembali hasil deteksinya untuk ditampilkan, kode program bisa dilihat pada Gambar 4.33

```
func runModel(onFrame pixelBuffer: CVPixelBuffer) -> Result? {
   …
   let outputBoundingBox: Tensor
   let outputClasses: Tensor
   let outputScores: Tensor
   let outputCount: Tensor
   …
   …
   …
   outputBoundingBox = try interpreter.output(at: 0)
   outputClasses = try interpreter.output(at: 1)
   outputScores = try interpreter.output(at: 2)
   outputCount = try interpreter.output(at: 3)
   …
   …
}
```

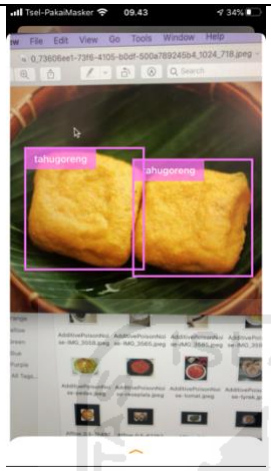
Gambar 4.33 Kode Program untuk *intrpreting* hasil deteksi model

Setelah melewati semua tahapan-tahapan di atas, model sudah dapat dijalankan di *mobile application*. Dengan begitu, *mobile application* sudah bisa digunakan untuk melakukan estimasi kalori pada makanan yang diberikan melalui citra.

## 4.2 Testing Model di *Mobile Application*

Pada tahapan kali ini, Penulis melakukan sebuah algoritma *testing* untuk mengukur kinerja dari model di *mobile application* menggunakan metode *confusion matrix*. Seluruh hasil pengujian sudah dapat dilihat pada Lampiran A. Pengujian model dilakukan dengan menguji 30 gambar yang ada pada *testing set* menggunakan *mobile application* yang sudah dibangun, contoh ditampilkan di Tabel 4.6. Lalu, hasil deteksi dari model akan dimasukkan ke dalam Tabel 3.4 yang sudah disiapkan sebelumnya. Langkah terakhir, seluruh hasil deteksi dimasukkan ke Tabel 4.8 dan akan dikalkulasikan nilai presisi, *recall*, *f1-score* dari tiap kelas, juga nilai *macro average* dan *weighted average* dari model pada Tabel 4.9.

Tabel 4.6 Salah satu contoh hasil pengujian model

| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 1 | Senin, 24 Agustus 2020, 08:30 |  | • Tahu Goreng<br>• Tahu Goreng | • Tahu Goreng<br>• Tahu Goreng |

Pada Tabel 4.6, kolom waktu pengambilan merupakan waktu pengujian ketika gambar dicoba ke dalam model menggunakan aplikasi. Kolom Gambar prediksi diisi dengan gambar hasil deteksi model menggunakan aplikasi, lalu terdapat kolom *true object* yang merupakan *list actual object* yang ada di gambar. Kolom *Predict Object* akan diisikan dengan list objek hasil deteksi model terhadap gambar. Ditampilkan juga di Tabel 4.7, daftar kelas dan jumlah objek pada data yang akan dimasukkan ke dalam tes di *confusion matrix.*

Tabel 4.7 Nama kelas dan jumlah objek pada data uji di *confusion matrix*

| Nama Kelas | Jumlah |
|---|---|
| Telur Dadar | 9 |
| Nasi Putih | 8 |
| Ikan Lele Goreng | 4 |
| Tumis Kangkung | 8 |
| Dada Ayam Goreng | 4 |
| Tempe Goreng | 6 |
| Tahu Goreng | 12 |

| Nama Kelas | Jumlah |
|---|---|
| Kerupuk Putih | 6 |
| Sambal | 5 |

Tabel 4.8 Tabel *Confusion Matrix*

| | | True Class | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Dada Ayam | Nasi Putih | Tahu Goreng | Tempe Goreng | Kerupuk Putih | Tumis Kangkung | Lele Goreng | Telur Dadar | Sambal |
| **Predicted Class** | Dada Ayam | **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Nasi Putih | 0 | **7** | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| | Tahu Goreng | 0 | 0 | **11** | 0 | 0 | 0 | 0 | 0 | 0 |
| | Tempe Goreng | 0 | 0 | 0 | **6** | 0 | 0 | 0 | 0 | 0 |
| | Kerupuk Putih | **1** | 0 | 0 | 0 | **6** | 0 | 0 | 0 | 0 |
| | Tumis Kangkung | 0 | 0 | 0 | 0 | 0 | **5** | 0 | 0 | 0 |
| | Lele Goreng | 0 | 0 | 0 | 0 | 0 | **1** | **4** | 0 | 0 |
| | Telur Dadar | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **8** | 0 |
| | Sambal | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **4** |

Pada Tabel 4.8 terdapat sedikit hasil pengujian dengan nilai *false positive*, yaitu, telur dadar yang dideteksi sebagai nasi putih, juga dada ayam yang dideteksi sebagai kerupuk putih, dan tumis kangkung yang dideteksi sebagai lele goreng.

Setelah Tabel 4.8 diisikan berdasarkan hasil pengujian 30 gambar dari *testing set*, selanjutnya adalah mengkalkulasikan nilai-nilai ke dalam presisi, recall, dan juga F1-Scores dari tiap kelas. Hal ini dilakukan untuk mengukur kinerja model terhadap tiap kelasnya. Untuk hasil perhitungan sendiri, dapat dilihat di Tabel 4.9

Tabel 4.9 Tabel Perhitungan Presisi, *Recall*, *F1-Scores*, *Macro-Average*, dan *Weighted Average*

| No | Nama Kelas | Presisi | *Recall* | F1-Score | Support |
|---|---|---|---|---|---|
| 1 | Dada Ayam | 100% | 75% | 85% | 4 |
| 2 | Nasi Putih | 87.5% | 100% | 93% | 8 |
| 3 | Tahu Goreng | 100% | 92% | 96% | 12 |
| 4 | Tempe Gor. | 100% | 100% | 100% | 6 |
| 5 | Kerupuk Putih | 86% | 100% | 92% | 6 |
| 6 | Tumis Kangk. | 100% | 62,5% | 76% | 8 |
| 7 | Lele Goreng | 80% | 100% | 89% | 4 |
| 8 | Telur Dadar | 100% | 88% | 94% | 9 |
| 9 | Sambal | 100% | 80% | 88% | 5 |
| **Macro Average** | | 95% | 89% | 90% | 62 |
| **Weighted Average** | | 96% | 89% | 90% | 62 |

## 4.3 Implementasi Rancangan Antarmuka Aplikasi & *Flow* Estimasi Kalori Menggunakan *Mobile Application*

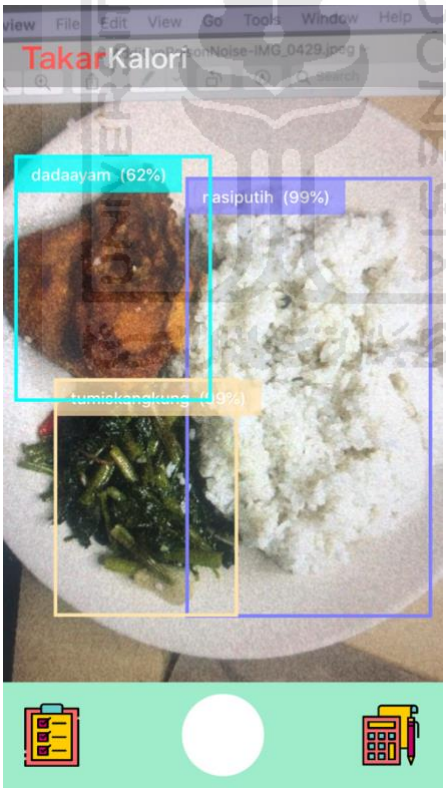
Saat aplikasi sudah bisa menjalankan model, untuk mendeteksi kalori pada makanan, aplikasi akan membuka kamera dan menampilkan apa yang dilihat oleh kamera secara *real-time.* Lalu di saat yang bersamaan, data dari kamera itu akan dikonversikan dan diberikan ke model yang sudah disiapkan di tahapan-tahapan sebelumnya. Dengan itu juga, aplikasi akan memunculkan *bounding box* dan *confidence class* dari hasil deteksi oleh model agar bisa dilihat oleh *user* aplikasi, seperti yang ditampilkan di Gambar 4.35. Setelah data makanan dirasa cukup sesuai, *user* dapat menekan Tombol *Capture*, di bawah tengah dari halaman utama, aplikasi akan melakukan *capture* dan melakukan deteksi kembali.

Lalu pada aplikasi, sudah dimasukkan data-data makanan yang sesuai dengan yang ada pada Tabel 4.1, dalam kasus ini, data makanan disiapkan menggunakan bahasa swift, kode program ditampilkan pada Gambar 4.34. Lalu, objek makanan yang berhasil dikenali, akan

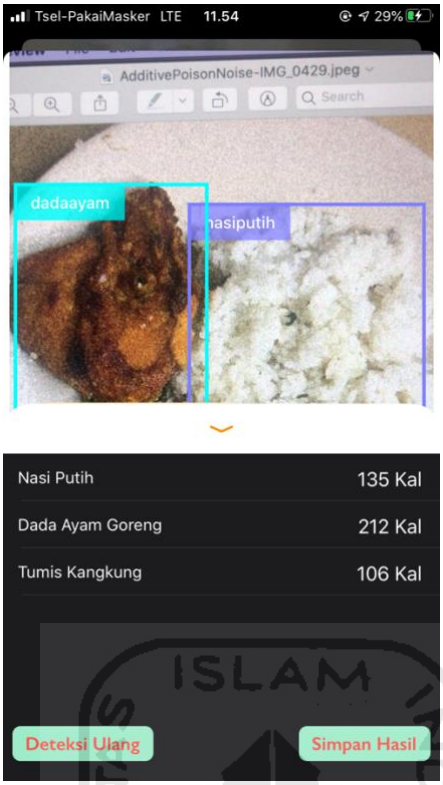
langsung dicocokkan dengan data makanan yang ada pada kode progam di Gambar 4.34. Sehingga, *user* bisa mengetahui estimasi kalori yang ada pada makanan, tampilan terlampir pada Gambar 4.35 dan Gambar 4.36

```
Var dataMakanan = [Makanan(namaMakanan: "Dada Ayam",kalori: 212),
                   Makanan(namaMakanan: "Lele Goreng",kalori: 204),
                   Makanan(namaMakanan: "Nasi Putih",kalori: 135),
                   Makanan(namaMakanan: "Telur Dadar",kalori: 135),
                   Makanan(namaMakanan: "Tumis Kangkung",kalori: 106),
                   Makanan(namaMakanan: "Tempe Goreng",kalori: 34),
                   Makanan(namaMakanan: "Tahu Goreng",kalori: 35),
                   Makanan(namaMakanan: "Kerupuk Putih",kalori: 65),
                   Makanan(namaMakanan: "Sambal",kalori: 102)]
```

Gambar 4.34. Tampilan kode program data kalori per-porsi dari jenis makanan

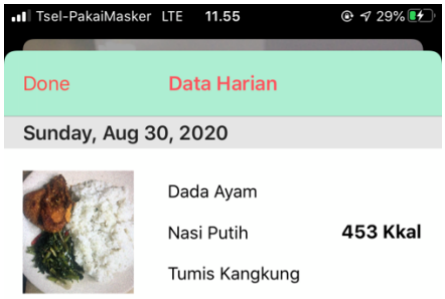



Gambar 4.35 Tampilan Halaman Utama Aplikasi "Takar Kalori-Mu"




Gambar 4.36 Tampilan halaman Estimasi Kalori Aplikasi "Takar Kalori-Mu"

Lalu, jika sudah berhasil dan sesuai, *user* dapat melakukan simpan pada data makanan beserta kalori yang sudah dimakan. Aplikasi juga akan melakukan penyimpanan menggunakan teknologi *core-data* yang diimplementasikan pada penelitian ini, sehingga tiap *user* bisa memantau asupan kalori harian mereka sendiri, seperti yang ditampilkan pada Gambar 4.37.




Gambar 4.37 Tampilan halaman "Data Harian"

Data makanan yang sudah dideteksi dan berhasil disimpan akan muncul seperti yang ditampilkan pada Gambar 4.37, jadi tiap kali *user* menyimpan data, akan dimunculkan di halaman data harian beserta informasi tanggal konsumsi makanan.

**Implementasi antarmuka "Halaman Statistik Kalori" (Pengitung kalori dalam tubuh)**

Pada aplikasi, terdapat juga halaman pendukung lain yaitu, halaman "Statistik Kalori" yang dapat menampilkan kalori harian, mingguan, dan bulanan yang masuk dari aplikasi. Di sana, juga terdapat, jumlah *steps* yang bisa dicek harian, mingguan, dan bulanan lalu dikalkulasikan menjadi kalori keluar, dan dikalkulasikan kembali untuk mendapatkan nilai defisit kalorinya. Halaman lengkap "Statistik Kalori" ditampilkan pada Gambar 4.38.


Gambar 4.38 Tampilan Halaman "Statistik kalori"

Pada halaman ini, akan ditampilkan kalori yang masuk ke tubuh dari makanan, dalam rentang, harian, mingguan, sampai bulanan. Halaman ini juga menampilkan *steps-counting* sampai kalori yang terbakar dari step tersebut. Data-data *steps-counting* dan *kalori* yang dibakar itu diambil dari aplikasi bawaan yang dimiliki oleh tiap *user* di sistem operasi iOS, *health-kit app*. Terdapat juga data defisit kalori yang dimana merupakan hasil pengurangan dari **kalori masuk** dikurangi dengan **kalori keluar.**

## 4.4 Pengujian Estimasi Kalori pada Aplikasi dengan Nilai Sebenarnya

Setelah melalui beberapa tahapan sebelumnya dari proses pembuatan model sampai menjalankan model di *mobile application*, tahapan berikutnya adalah melakukan pengujian pada aplikasi dalam mengestimasi kalori. Seperti yang sudah dijelaskan di subbab 3.4, pengujian ini adalah melakukan perbandingan estimasi kalori yang diberikan aplikasi dengan data kalori yang diberikan oleh *FatSecret Indonesia*. Pengujian dilakukan dengan memberikan aplikasi 10 gambar (citra) makanan beserta kalori dari makanan yang di dapatkan di *FatSecret Indonesia*. Salah satu hasil pengujian ditampilkan di dalam Tabel 4.10

Tabel 4.10 Tabel pengujian Estimasi Kalori

<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="5">1</td><td rowspan="2">  </td><td>Telur Dadar</td><td>135 Kal</td></tr>
<tr><td>Nasi Putih</td><td>135 Kal</td></tr>
<tr><td>Tumis Kangkung</td><td>106 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td> 
</td><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td>Nasi Putih</td><td>135 Kal</td></tr>
<tr><td>Telur Dadar</td><td>135 Kal</td></tr>
<tr><td>Tumis Kangkung</td><td>106 Kal</td></tr>
</table>

Seluruh hasil pengujian dapat dilihat pada Lampiran B. Dari 10 hasil pengujian yang sudah dilakukan, penulis tidak menemukan adanya data kalori pada makanan yang tidak sesuai dengan data kalori yang diberikan oleh *FatSecret Indonesia*. Berdasarkan hasil pengujian, akurasi *mobile application* dalam mengestimasi kalori setelah jenis makanan dideteksi adalah 100%.

Oleh karena itu, penulis dapat menyimpulkan bahwa aplikasi sudah berjalan dengan sangat baik, dan sudah siap untuk digunakan untuk mengestimasi kalori.

# BAB VI
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Dari penelitian yang sudah dilakukan, terdapat beberapa hasil yang dapat disimpulkan, yaitu:

*a.* *Mobile Application* dibangun dengan memanfaatkan model *SSD (Single Shot Multibox Detector)* untuk melakukan deteksi pada objek makanan, setelah itu, objek makanan yang berhasil dideteksi akan diasumsikan kalori nya oleh *Mobile Application*. Asumsi kalori akan memanfaatkan informasi kalori per-porsi yang diberikan oleh *FatSecret Indonesia*.

*b.* Performa model saat mendeteksi jenis makanan di *mobile application* menghasilkan nilai yang cukup baik, dengan nilai *Macro Average* pada *presisi* dan *recall* di atas 85% untuk 9 jenis makanan berbeda (kelas). *Mobile Application* juga mampu mengestimasi kalori setelah jenis makanan dideteksi.

## 5.2 Saran

Untuk pengembangan analisis lebih lanjut, terdapat beberapa saran yang diberikan, untuk kedepannya agar dapat menjadi lebih baik lagi. Berikut adalah beberapa saran tersebut:

a. Memperbanyak data gambar makanan agar model bisa mengenali lebih banyak lagi jenis makanan beserta asupan makanan.

b. Membangun dan mengimplementasikan model yang mampu untuk mengestimasi berat makanan dari sebuah gambar; sehingga estimasi kalori dari makanan dapat dilakukan akurasi lebih baik.

c. Mengembangkan aplikasi sehingga dapat mencatat aktivitas harian dengan lebih detil, sehingga perhitungan kalori masuk dan keluar dapat lebih presisi.

# DAFTAR PUSTAKA


Chokr, M., & Elbassuoni, S. (2017). Calories Prediction from Food Images. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 4664–4669. Retrieved from http://dl.acm.org/citation.cfm?id=3297863.3297871

Ege, T., & Yanai, K. (2017). Simultaneous estimation of food categories and calories with multi-task CNN. *Proceedings of the 15th IAPR International Conference on Machine Vision Applications, MVA 2017*, 198–201. https://doi.org/10.23919/MVA.2017.7986835

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. Retrieved from http://arxiv.org/abs/1704.04861

Khairani. (2019). Hari Diabetes Sedunia Tahun 2018. *Pusat Data Dan Informasi Kementrian Kesehatan RI*, 1–8.

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object recognition with gradient-based learning. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/3-540-46805-6_19

Liang, Y., & Li, J. (2017). *Deep Learning-Based Food Calorie Estimation Method in Dietary Assessment*. (Jianhua Li). Retrieved from http://arxiv.org/abs/1706.04062

Miyazaki, T., De Silva, G. C., & Aizawa, K. (2011). Image-based calorie content estimation for dietary assessment. *Proceedings - 2011 IEEE InternationalSymposium on Multimedia, ISM 2011*, 363–368. https://doi.org/10.1109/ISM.2011.66

Phung, V. H., & Rhee, E. J. (2019). A High-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences (Switzerland)*, *9*(21). https://doi.org/10.3390/app9214500

Saracoglu, Ö. G., & Altural, H. (2010). Color regeneration from reflective color sensor using an artificial intelligent technique. *Sensors*, *10*(9), 8363–8374. https://doi.org/10.3390/s100908363

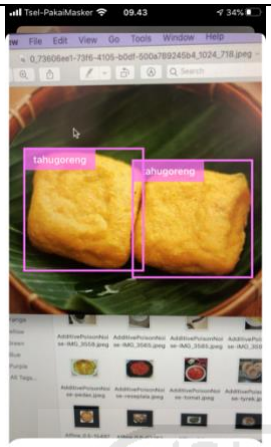
Staub, S., Karaman, E., Kaya, S., Karapınar, H., & Güven, E. (2015). Artificial Neural Network and Agility. *Procedia - Social and Behavioral Sciences*, *195*, 1477–1485. https://doi.org/10.1016/j.sbspro.2015.06.448

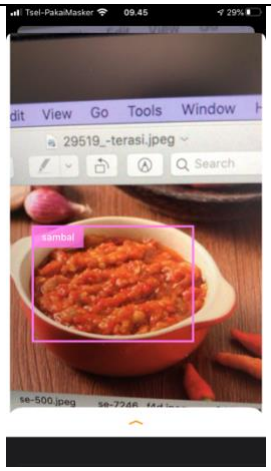
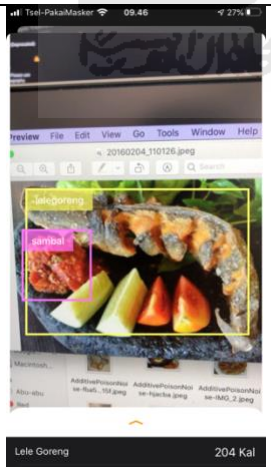
Tirasirichai, B., Thanomboon, P., Soontorntham, P., Kusakunniran, W., & Robinson, M. (2018). Bloom Balance: Calorie Balancing Application with Scientific Validation.

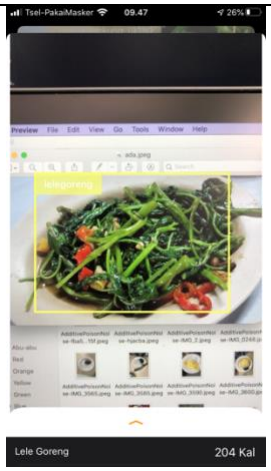
*Proceeding of 2018 15th International Joint Conference on Computer Science and Software Engineering, JCSSE 2018*. https://doi.org/10.1109/JCSSE.2018.8457177

Wasif, S. M., & Pereira, S. I. (2019). *Food calorie estimation using machine learning and image processing*. *5*(2), 1627–1630.

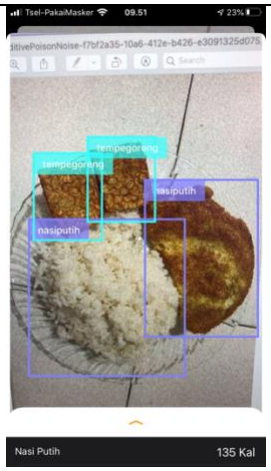
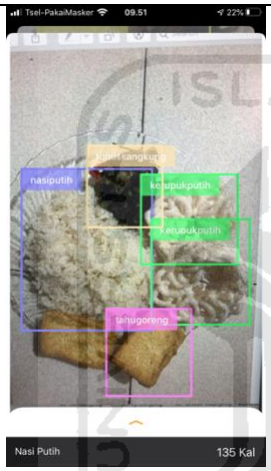
Lampiran A

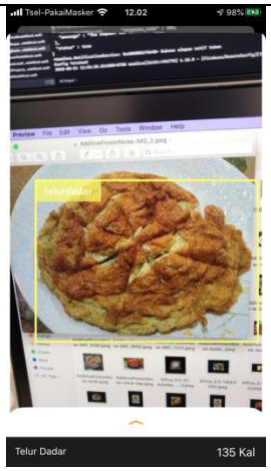
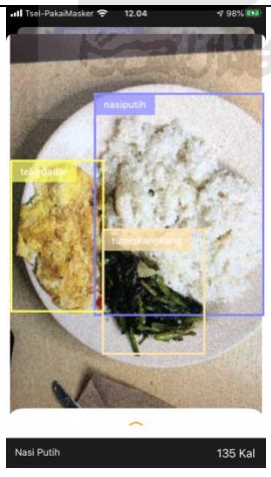
| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 1 | Senin, 24 Agustus 2020, 08:30 |  | • Tahu Goreng<br>• Tahu Goreng | • Tahu Goreng<br>• Tahu Goreng |
| 2 | Senin, 24 Agustus 2020, 08:31 |  | • Kerupuk Putih | • Kerupuk Putih |
| 3 | Senin, 24 Agustus 2020, 08:32 |  | • Tempe Goreng<br>• Tempe Goreng | • Tempe Goreng<br>• Tempe Goreng |

| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 4 | Senin, 24 Agustus 2020, 08:33 |  | • Sambal | • Sambal |
| 5 | Senin, 24 Agustus 2020, 08:34 |  | • Dada Ayam<br>• Sambal | • Kerupuk Putih<br>• Sambal |
| 6 | Senin, 24 Agustus 2020, 08:35 |  | • Lele Goreng<br>• Sambal | • Lele Goreng<br>• Sambal |

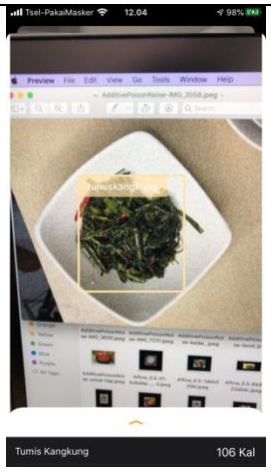
| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 7 | Senin, 24 Agustus 2020, 08:36 |  | • Tumis Kangkung | • Lele Goreng |
| 8 | Senin, 24 Agustus 2020, 08:37 |  | • Lele Goreng<br>• Sambal | • Lele Goreng<br>• Sambal |
| 9 | Senin, 24 Agustus 2020, 08:38 |  | • Kerupuk Putih | • Kerupuk Putih |

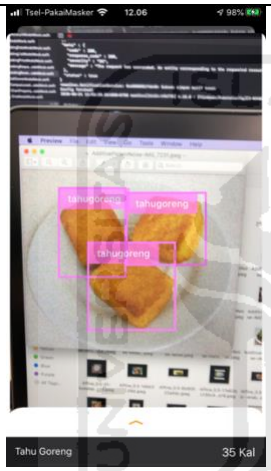
| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 10 | Senin, 24 Agustus 2020, 08:39 |  | • Tahu Goreng<br>• Tahu Goreng<br>• Tahu Goreng<br>• Tahu Goreng<br>• Tahu Goreng | • Tahu Goreng<br>• Tahu Goreng<br>• Tahu Goreng<br>• Tahu Goreng<br>• Tahu Goreng |
| 11 | Senin, 24 Agustus 2020, 08:40 |  | • Nasi Putih<br>• Tempe Goreng<br>• Tempe Goreng | • Nasi Putih<br>• Tempe Goreng<br>• Tempe Goreng |
| 12 | Senin, 24 Agustus 2020, 08:41 |  | • Telur Dadar | • Telur Dadar |

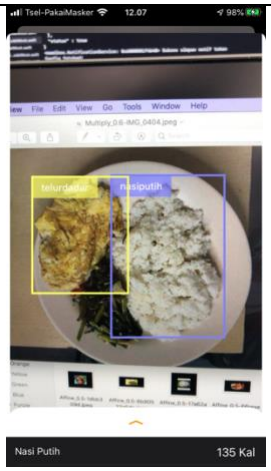
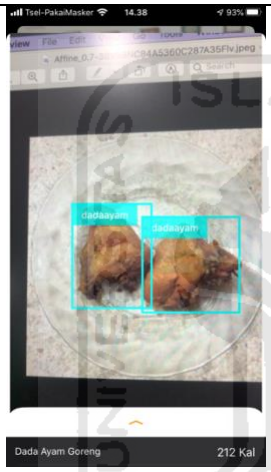
| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 13 | Senin, 24 Agustus 2020, 08:42 |  | • Nasi Putih<br>• Telur Dadar<br>• Tempe Goreng<br>• Tempe Goreng | • Nasi Putih<br>• Nasi Putih<br>• Tempe Goreng<br>• Tempe Goreng |
| 14 | Senin, 24 Agustus 2020, 08:43 |  | • Nasi Putih<br>• Tumis Kangkung<br>• Kerupuk Putih<br>• Kerupuk Putih<br>• Tahu Goreng<br>• Tahu Goreng | • Nasi Putih<br>• Tumis Kangkung<br>• Kerupuk Putih<br>• Kerupuk Putih<br>• Tahu Goreng |
| 15 | Senin, 24 Agustus 2020, 08:44 |  | • Lele Goreng | • Lele Goreng |

| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 16 | Senin, 24 Agustus 2020, 08:45 |  | • Telur Dadar | • Telur Dadar |
| 17 | Senin, 24 Agustus 2020, 08:46 |  | • Nasi Putih | • Nasi Putih |
| 18 | Senin, 24 Agustus 2020, 08:47 |  | • Telur Dadar<br>• Nasi Putih<br>• Tumis Kangkung | • Telur Dadar<br>• Nasi Putih<br>• Tumis Kangkung |

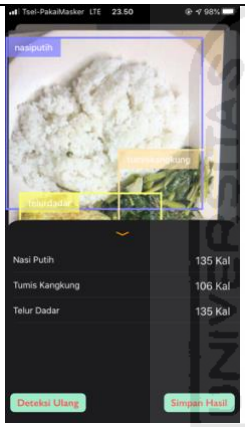
| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 19 | Senin, 24 Agustus 2020, 08:48 |  | • Kerupuk Putih | • Kerupuk Putih |
| 20 | Senin, 24 Agustus 2020, 08:49 |  | • Kerupuk Putih | • Kerupuk Putih |
| 21 | Senin, 24 Agustus 2020, 08:50 |  | • Telur Dadar | • Telur Dadar |

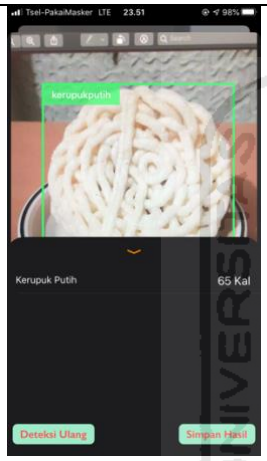
| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 22 | Senin, 24 Agustus 2020, 08:51 |  | • Tumis Kangkung | • Tumis Kangkung |
| 23 | Senin, 24 Agustus 2020, 08:52 |  | • Lele Goreng<br>• Tumis Kangkung<br>• Nasi Putih | • Lele Goreng<br>• Tumis Kangkung<br>• Nasi Putih |
| 24 | Senin, 24 Agustus 2020, 08:53 |  | • Telur Dadar | • Telur Dadar |

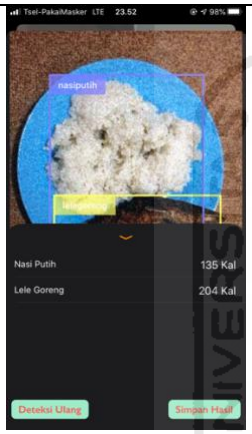
| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 25 | Senin, 24 Agustus 2020, 08:54 |  | • Telur Dadar<br>• Tumis Kangkung | • Telur Dadar |
| 26 | Senin, 24 Agustus 2020, 08:55 |  | • Tahu Goreng<br>• Tahu Goreng<br>• Tahu Goreng | • Tahu Goreng<br>• Tahu Goreng<br>• Tahu Goreng |
| 27 | Senin, 24 Agustus 2020, 08:56 |  | • Tumis Kangkung | • Tumis Kangkung |

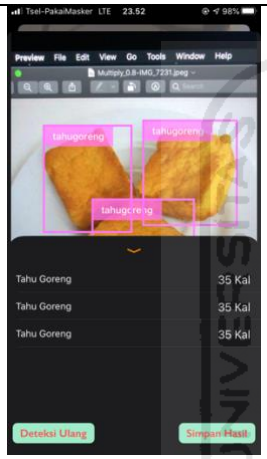
| No | Waktu Pengambilan | Gambar Prediksi | *True Object* | *Predict Object* |
|---|---|---|---|---|
| 28 | Senin, 24 Agustus 2020, 08:57 |  | • Nasi Putih<br>• Telur Dadar<br>• Tumis Kangkung | • Nasi Putih<br>• Telur Dadar |
| 29 | Senin, 24 Agustus 2020, 15:34 |  | • Dada Ayam<br>• Dada Ayam | • Dada Ayam<br>• Dada Ayam |
| 30 | Senin, 20 Juli 2020, 21:00 |  | • Dada Ayam<br>• Telur Dadar | • Dada Ayam<br>• Telur Dadar |

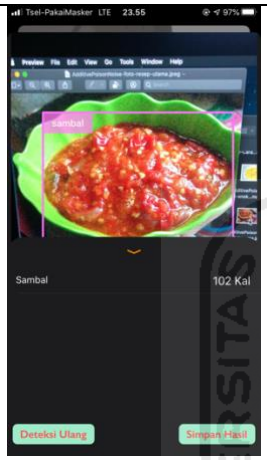
**Lampiran B**

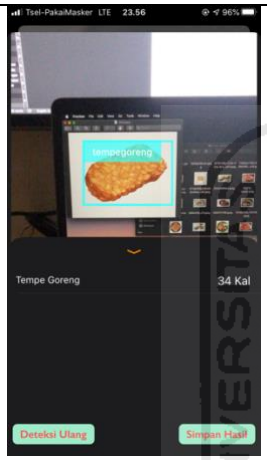
<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="6">1</td><td rowspan="2">  </td><td>Telur Dadar</td><td>135 Kal</td></tr>
<tr><td>Nasi Putih</td><td>135 Kal</td></tr>
<tr><td> 
</td><td>Tumis Kangkung</td><td>106 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td>Nasi Putih</td><td>135 Kal</td></tr>
<tr><td>Telur Dadar</td><td>135 Kal</td></tr>
<tr><td>Tumis Kangkung</td><td>106 Kal</td></tr>
</table>

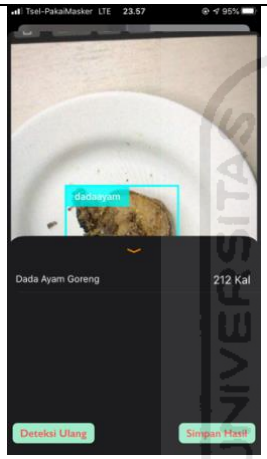
<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="4">2</td><td>  </td><td>Kerupuk Putih</td><td>65 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td> 
</td><td>Kerupuk Putih</td><td>65 Kal</td></tr>
</table>

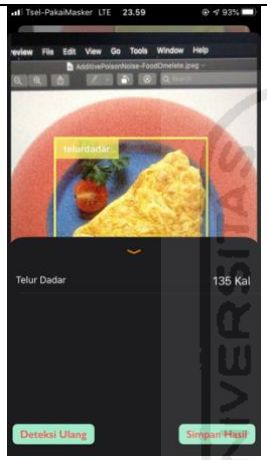
<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="5">3</td><td rowspan="2">  </td><td>Lele Goreng</td><td>204 Kal</td></tr>
<tr><td>Nasi Putih</td><td>135 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td rowspan="2"> 
</td><td>Nasi Putih</td><td>135 Kal</td></tr>
<tr><td>Lele Goreng</td><td>204 Kal</td></tr>
</table>

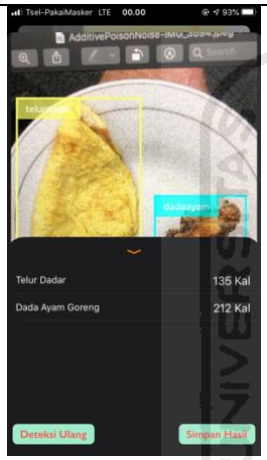
<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="6">4</td><td rowspan="2">  </td><td>Tahu Goreng</td><td>35 Kal</td></tr>
<tr><td>Tahu Goreng</td><td>35 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td>Tahu Goreng</td><td>35 Kal</td></tr>
<tr><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td rowspan="3"> 
</td><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td>Tahu Goreng</td><td>35 Kal</td></tr>
<tr><td>Tahu Goreng</td><td>35 Kal</td></tr>
<tr><td>Tahu Goreng</td><td>35 Kal</td></tr>
</table>

<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="4">5</td><td>  </td><td>Sambal</td><td>102 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td> 
</td><td>Sambal</td><td>102 Kal</td></tr>
</table>

<table>
<tr><td rowspan="2">No</td><td rowspan="2">Gambar Makanan</td><td colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td rowspan="4">6</td><td>  </td><td>Tempe Goreng</td><td>34 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td> 
</td><td>Tempe Goreng</td><td>34 Kal</td></tr>
</table>

<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="4">7</td><td>  </td><td>Dada Ayam</td><td>212 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td> 
</td><td>Dada Ayam</td><td>212 Kal</td></tr>
</table>

<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="4">8</td><td>  </td><td>Telur Dadar</td><td>135 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td> 
</td><td>Telur Dadar</td><td>135 Kal</td></tr>
</table>

<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="5">9</td><td rowspan="2">  </td><td>Telur Dadar</td><td>135 Kal</td></tr>
<tr><td>Dada Ayam</td><td>212 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td rowspan="2"> 
</td><td>Dada Ayam</td><td>212 Kal</td></tr>
<tr><td>Telur Dadar</td><td>135 Kal</td></tr>
</table>

<table>
<tr><th rowspan="2">No</th><th rowspan="2">Gambar Makanan</th><th colspan="2">Kalori pada makanan (berdasarkan FatSecret Indonesia)</th></tr>
<tr><th>Nama Makanan</th><th>Kalori</th></tr>
<tr><td rowspan="4">10</td><td>  </td><td>Tumis Kangkung</td><td>106 Kal</td></tr>
<tr><td rowspan="2">Hasil Deteksi</td><td colspan="2">Estimasi Kalori pada aplikasi</td></tr>
<tr><td>Nama Makanan</td><td>Kalori</td></tr>
<tr><td> 
</td><td>Tumis Kangkung</td><td>106 Kal</td></tr>
</table>