SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

IMPLEMENTASI *DEEP LEARNING* MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN) DALAM KLASIFIKASI GAMBAR MAKANAN UNTUK MENENTUKAN KANDUNGAN KALORI PADA MAKANAN

TUGAS AKHIR

PRETTI NADA CAHAYA IRAWAN
0110220246

PROGRAM STUDI TEKNIK INFORMATIKA
DEPOK
AGUSTUS 2024

SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

IMPLEMENTASI *DEEP LEARNING* MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN) DALAM KLASIFIKASI GAMBAR MAKANAN UNTUK MENENTUKAN KANDUNGAN KALORI PADA MAKANAN

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana

PRETTI NADA CAHAYA IRAWAN
0110220246

PROGRAM STUDI TEKNIK INFORMATIKA
DEPOK
AGUSTUS 2024

HALAMAN PERNYATAAN ORISINALITAS

Skripsi/Tugas Akhir ini adalah hasil karya penulis, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama : Pretti Nada Cahaya Irawan

NIM : 0110220246

Depok, 23 Juli 2024

Tanda Tangan

Pretti Nada Cahaya Irawan

HALAMAN PENGESAHAN

Skripsi/Tugas Akhir ini diajukan oleh:
Nama : Pretti Nada Cahaya Irawan
NIM : 0110220246
Program Studi : Teknik Informatika
Judul Skripsi : Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Dalam Klasifikasi Gambar Makanan untuk Menentukan Kandungan Kalori pada Makanan

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri.

DEWAN PENGUJI

Pembimbing	Penguji
(Ahmad Rio Adriansyah, S.Si., M.Si.)	(Dr. Sirojul Munir, S.Si., M.Kom.)

Ditetapkan di : Depok
Tanggal : 23 Juli 2024

KATA PENGANTAR

Puji syukur penulis ucapkan ke hadirat Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan Tugas Akhir yang berjudul "Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) dalam Klasifikasi Gambar Makanan untuk Menentukan Kandungan Kalori pada Makanan". Penulisan Tugas Akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana komputer Program Studi Teknik Informatika pada Sekolah Tinggi Teknologi Terpadu Nurul Fikri. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, akan sangat sulit bagi penulis untuk menyelesaikan Tugas Akhir ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

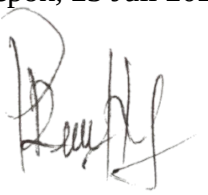
1. Allah SWT.
2. Kepada diri sendiri yang atas segala usaha, kerja keras, dan komitmen yang telah dicurahkan dalam menyelesaikan Tugas Akhir ini dengan baik. Tidak bisa dipungkiri bahwa proses penyelesaian ini penuh dengan tantangan dan rintangan. Namun, dengan semangat dan komitmen yang terus saya berikan, saya berhasil mencapai titik ini.
3. Orang tua, terutama mama, dan semua anggota keluarga (terutama adik-adik saya, Ghina, dan Luthfi) yang telah memberikan dorongan baik secara moril maupun materiil dalam penyelesaian tugas ini.
4. Bapak Dr. Lukman Rosyidi, S.T., M.M., M.T. selaku Ketua Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
5. Ibu Tifanny Nabarian, S.Kom, M.T.I. selaku Ketua Program Studi Teknik Informatika Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
6. Bapak Dr. Lukman Rosyidi, S.T., M.M., M.T selaku Dosen Pembimbing Akademik yang telah membimbing penulis selama berkuliah di Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
7. Bapak Ahmad Rio Adriansyah, S.Si., M.Si. selaku Dosen Pembimbing Tugas Akhir penulis dalam menyelesaikan penulisan ilmiah ini.

8. Bapak Dr. Sirojul Munir, S.Si., M.Kom, selaku Dosen Penguji Tugas Akhir penulis.
9. Para Dosen di lingkungan Sekolah Tinggi Teknologi Terpadu Nurul Fikri yang telah membimbing penulis dalam menuntut ilmu yang telah diberikan.
10. Bangkit beserta karyawan dan mentor yang telah memberikan banyak ilmu dan bimbingan kepada penulis selama mengikuti program studi independen di Bangkit Academy.
11. Teman-teman yang selalu menemani dan memberikan dukungan kepada penulis dalam menyelesaikan penulisan tugas akhir ini. Terutama anggota "*UIUX Team*", yaitu Farah, Fikri, Bambang, Mutia, dan Eka.

Dalam penulisan ilmiah ini tentu saja masih banyak terdapat kekurangan-kekurangan yang mungkin disebabkan oleh keterbatasan kemampuan dan pengetahuan yang penulis miliki. Walaupun demikian, penulis telah berusaha menyelesaikan penulisan ilmiah ini sebaik mungkin. Oleh karena itu apabila terdapat kekurangan di dalam penulisan ilmiah ini, dengan rendah hati penulis menerima kritik dan saran dari pembaca.

Akhir kata, penulis berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 23 Juli 2024



Pretti Nada Cahaya Irawan

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Sekolah Tinggi Teknologi Terpadu Nurul Fikri, saya yang bertanda tangan di bawah ini:

Nama : Pretti Nada Cahaya Irawan

NIM : 0110220246

Program Studi : Teknik Informatika

Jenis karya : Skripsi / Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada STT-NF **Hak Bebas Royalti Non eksklusif (*Non-exclusive Royalty - Free Right*)** atas karya ilmiah saya yang berjudul :

Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Dalam Klasifikasi Gambar Makanan untuk Menentukan Kandungan Kalori pada Makanan

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non eksklusif ini STT-NF berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok, Jawa Barat

Pada tanggal : 23 Juli 2024

Yang menyatakan

(Pretti Nada Cahaya Irawan)

ABSTRAK

(300 kata)

Nama : Pretti Nada Cahaya Irawan

NIM : 0110220246

Program Studi : Teknik Informatika

Judul : Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Dalam Klasifikasi Gambar Makanan untuk Menentukan Kandungan Kalori pada Makanan

Angka kekurangan gizi di Indonesia meningkat pasca pandemi. Berdasarkan hasil survei status gizi Indonesia pada tahun 2022, angka *underweight* di Indonesia naik sebesar 0.1%, menjadi 17,1% pada tahun 2022. Angka *wasting* juga naik sebesar 0.6%, menjadi 7.7% pada tahun 2022. Salah satu faktor penyebab permasalahan ini adalah pola makan yang buruk dan kurangnya pemahaman mengenai kebutuhan kalori harian yang terpenuhi dari makanan yang dikonsumsi. Oleh karena itu, untuk mengatasi permasalahan ini diperlukan sistem yang dapat membantu masyarakat memahami kandungan gizi dari makanan yang dikonsumsi. Penelitian ini dilakukan untuk membangun model *machine learning* dengan mengimplementasikan algoritma *Convolutional Neural Network* (CNN) yang akan mendukung fitur yang dibangun pada sistem yang dapat menampilkan kalori pada makanan. Metode analisis data yang digunakan adalah metode analisis kuantitatif dengan menggunakan metrik *accuracy*. Hasil penelitian ini adalah model *machine learning* yang dapat melakukan klasifikasi terhadap 20 kelas data dengan *accuracy* sebesar 75%. Dalam evaluasinya, masih terdapat beberapa kelas yang masih sering salah diprediksi oleh model. Oleh karena itu, disarankan untuk menggunakan *dataset* yang lebih besar lagi, terutama pada kelas yang masih sering salah prediksi, seperti “belimbing” dan “pisang”.

Kata kunci : *Convolutional Neural Network* (CNN), *deep learning*, klasifikasi gambar, *machine learning*,

ABSTRACT

Name : *Pretti Nada Cahaya Irawan*

NIM : *0110220246*

Study Program : *Informatics*

Title : *Implementation of Deep Learning Using Convolutional Neural Network (CNN) Algorithm in Food Image Classification to Determine the Calorie Content of Foods*

The malnutrition rate in Indonesia has increased after the pandemic. Based on the results of the Indonesian nutritional status survey in 2022, the underweight rate in Indonesia increased by 0.1%, to 17.1% in 2022. The wasting rate also increased by 0.6%, to 7.7% in 2022. One of the factors that contribute to this problem is unhealthy diet and a lack of understanding of the daily calorie requirements that are met from the food consumed. To overcome this problem, a system is needed by the community that can help them understand the nutritional content of the food they consume. This research was conducted to build a machine learning model by implementing the Convolutional Neural Network (CNN) algorithm that will support the features that were developed in the system that can present the calorie content of food. The data analysis method used is quantitative analysis method with accuracy metric. The result of this research is a machine learning model that can classify 20 classes of data with an accuracy of 75%. However, during the evaluation, there are still some classes that are still often mispredicted by the model. Therefore, it is recommended to use a larger dataset, especially for classes that are still often mispredicted, such as "belimbing" and "pisang".

Key words : Convolutional Neural Network (CNN), deep learning, image classification ,machine learning

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bagian pendahuluan dijelaskan mengenai permasalahan yang melatarbelakangi dilakukannya penelitian ini. Sub-bagian yang terdapat pada bab I ini yaitu terdiri dari latar belakang, rumusan masalah, tujuan dan manfaat penelitian, batasan masalah, serta sistematika penulisan.

1.1 Latar belakang

Indonesia termasuk salah satu negara yang memiliki pertumbuhan ekonomi digital tertinggi di Asia Tenggara. Hal ini menjadi salah satu strategi utama Indonesia dalam mendorong tercapainya transformasi ekonomi sekaligus bertujuan untuk mendorong pemulihan ekonomi pasca pandemi Covid-19[1]. Pertumbuhan ekonomi yang pesat ini seharusnya dapat mengubah dan mengurangi jumlah penduduk yang mengalami masalah nutrisi. Angka kekurangan gizi global telah meningkat secara signifikan dari 8% pada tahun 2019, kemudian mencapai 9.3% pada tahun 2020 dan terus meningkat dengan laju yang lebih lambat menjadi 9.8% pada 2021, setelah pandemi melanda[2].

Menurut laporan dari *Food and Agriculture Organization of the United Nations* (FAO), *International Fund for Agricultural Development* (IFAD), *United Nations Children's Fund* (UNICEF), *World Food Programme* (WFP) dan *World Health Organization* (WHO), tahun 2022, Indonesia termasuk sebagai salah-satu negara dengan angka kekurangan gizi tertinggi di kawasan Asia Tenggara (6.5% dari total populasi nasional)[2]. Berdasarkan hasil survei oleh Kementerian Kesehatan RI, angka *underweight* di Indonesia mengalami peningkatan sebesar 0.1%, yaitu menjadi 17,1% pada tahun 2022. Angka *wasting* juga mengalami peningkatan sebesar 0.6%, sehingga pada tahun 2022 menjadi 7.7%[3]. Selain itu, angka *stunting* di Indonesia mencapai 21.6% pada 2022[3]. Tingginya angka *stunting* dan *wasting* berdampingan dengan meningkatnya angka kelebihan berat badan dan obesitas. Penyakit seperti obesitas dan penyakit jantung semakin meningkat karena tidak adanya

pemahaman yang baik mengenai kandungan kalori pada makanan yang dikonsumsi. Pada tahun 2022, di Indonesia angka penduduk yang mengalami obesitas mencapai 3.5%[3]. Hampir seperempat dari peningkatan jumlah orang yang mengalami obesitas berasal dari kelompok umur di atas 15 tahun[2].

Permasalahan kekurangan gizi ini tentu saja memiliki faktor penyebabnya. Pola makan yang buruk menjadi salah satu faktor penyebab masalah kesehatan ini, bahkan hingga penjuru dunia[4], [5]. Banyak orang yang hanya mengandalkan makanan yang mudah diakses di luar rumah untuk memenuhi kebutuhan hariannya, seperti makanan cepat saji, makanan kemasan, makanan olahan, dll. Namun, konsumsi makanan semacam itu sering kali tidak dapat memenuhi kebutuhan kalori harian yang dibutuhkan oleh tubuh, bahkan tidak baik untuk tubuh[6].

Selain itu, masyarakat juga tidak terlalu memperhatikan kalori yang dikonsumsi sudah memenuhi kebutuhan harian atau tidak. Minimnya literasi terhadap kandungan kalori ini tentu membuat orang menjadi semakin tidak peduli dengan kesehatan dan asupan tubuhnya masing-masing. Dalam upaya mengurangi permasalahan ini diperlukan upaya komprehensif, termasuk peningkatan terhadap peningkatan literasi gizi dan program yang mendorong pola makan seimbang dan gaya hidup sehat. Oleh sebab itu, inovasi yang dapat memberikan pemahaman mengenai literasi kandungan kalori makanan sangat diperlukan dalam upaya mengatasi permasalahan tersebut.

Sebagaimana permasalahan yang telah diuraikan pada paragraf sebelumnya, penelitian ini akan lebih terfokus pada pengembangan model *machine learning*. Dengan memanfaatkan pengenalan gambar (*image recognition*) yang menggunakan algoritma *Convolutional Neural Network (CNN)* sebagai fitur pendeteksi gambar makanan, memungkinkan pengguna untuk mengetahui nilai gizi atau kalori pada makanan yang dikonsumsi melalui fitur *input* gambar yang disediakan.

Melalui penggunaan sistem aplikasi ini, diharapkan masyarakat dapat lebih sadar akan pentingnya pola makan sehat dan memahami nilai gizi dari makanan yang mereka konsumsi, serta memastikan kebutuhan kalori harian mereka terpenuhi, sehingga dapat mengurangi risiko *stunting*, *wasting*, dan obesitas.

1.2 Rumusan Masalah

Berikut ini adalah daftar rumusan masalah dalam penelitian tugas akhir ini, yang didasarkan pada latar belakang yang telah disebutkan sebelumnya:

1. Bagaimana membangun model menggunakan algoritma *Convolutional Neural Network* (CNN) untuk klasifikasi gambar makanan?
2. Bagaimana performa atau akurasi model saat melakukan deteksi gambar makanan?

1.3 Tujuan dan Manfaat Penelitian

1.3.1 Tujuan Penelitian

Adapun tujuan dari dilakukannya penelitian ini, yaitu:

1. Mengetahui dan memahami hasil implementasi algoritma *Convolutional Neural Network (*CNN) dalam pengembangan model yang dapat melakukan klasifikasi gambar makanan.
2. Mengetahui hasil performa atau akurasi dalam melakukan deteksi dan klasifikasi gambar makanan.

1.3.2 Manfaat Penelitian

Masyarakat dapat memperoleh manfaat dari hasil penelitian ini salah satunya dalam memahami estimasi kalori dari makanan yang dikonsumsi serta memastikan kalori harian terpenuhi.

1.4 Batasan Masalah

Berikut ini merupakan batasan masalah dalam penelitian yang dilakukan:

1. Fokus penelitian ini terletak pada pengembangan model *machine learning* dengan penerapan algoritma *Convolutional Neural Network* (CNN) untuk melakukan klasifikasi gambar makanan.

2. Dalam menentukan estimasi kalori didasarkan pada kalori per 100 gram untuk masing-masing buah dan sayur yang diambil pada *website FatSecret* Indonesia.
3. *Dataset* yang digunakan, seperti *training set* dan *test set*, sebagian besar diambil melalui internet dengan situs utama sebagai sumber data yaitu *Kaggle*.
4. Data gambar yang digunakan pada penelitian ini dibatasi untuk gambar 20 jenis buah-buahan dan sayuran saja. Data tersebut terdiri dari anggur, apel, belimbing, buah naga, jagung, jeruk, kentang, kol, labu, mangga, nanas, pepaya, pir, pisang, selada, semangka, *strawberry*, timun, tomat, dan wortel.

1.5 Sistematika Penulisan

Secara keseluruhan, terdapat lima bab dalam penelitian ini, sebagaimana dijelaskan untuk masing-masing bab sebagai berikut:

1. **Bab I Pendahuluan,** menyajikan latar belakang, rumusan masalah, tujuan dan manfaat yang berkaitan dengan penelitian, batasan masalah, dan sistematika penulisan. Bab I akan menjadi dasar dari penelitian yang selanjutnya akan menghasilkan suatu kesimpulan.
2. **Bab II Kajian Literatur**, menjelaskan mengenai definisi-definisi, teori-teori dengan analisis penelitian, serta penelitian terkait dan menjadi pedoman dalam melakukan penelitian ini.
3. **Bab III Metodologi Penelitian**, pada bab ini akan terdapat penjelasan mengenai tahapan penelitian dan rancangan penelitian. Sub bab rancangan penelitian terdiri dari jenis penelitian, metode analisis data, metode pengumpulan data, metode pengujian, metode implementasi dan evaluasi, serta lingkungan pengembangan.

4. **Bab IV Implementasi dan Evaluasi**, berisi mengenai analisis dan proses pengembangan dari sistem yang akan dikembangkan, hasil akhir dari penelitian yang telah dilakukan, dan evaluasi dari hasil penelitian yang didapatkan.
5. **Bab V Kesimpulan dan Saran**, bab ini merupakan bab terakhir di mana akan diuraikan mengenai kesimpulan dari penelitian yang sudah dilakukan serta saran untuk penelitian selanjutnya.

BAB II
KAJIAN LITERATUR

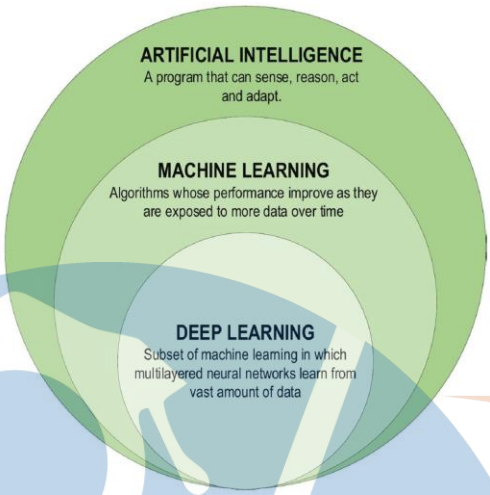
Pada Bab II ini, disajikan kajian literatur mengenai definisi-definisi, teori-teori dengan analisis penelitian, serta penelitian terkait yang relevan dengan topik penelitian ini. Untuk memahami dasar-dasar teori dan prinsip-prinsip yang mendasari penelitian ini, akan lebih mudah memahami dengan membaca bab ini.

2.1 Kalori

Kalori merupakan terminologi yang digunakan dalam perhitungan satuan energi yang terkandung dalam makanan[7]. Agar sistem tubuh mampu berfungsi secara efektif sebagai sumber energi untuk aktivitas sehari-hari, kalori dari makanan sangat dibutuhkan[7]. Makanan yang mengandung nutrisi penting seperti protein, lemak, dan karbohidrat akan menghasilkan kalori. Setiap individu membutuhkan jumlah kalori yang berbeda berdasarkan indeks berat badan, tinggi badan, usia, serta aktivitas dan gerakan yang dilakukan sepanjang hari. Terlalu banyak atau terlalu sedikit kalori dalam tubuh tidak baik untuk kesehatan. Oleh karena itu, penting untuk menghitung kebutuhan kalori tubuh per hari. Biasanya untuk mengontrol asupan kalori dilakukan dengan cara diet atau mengatur pola makan[7].

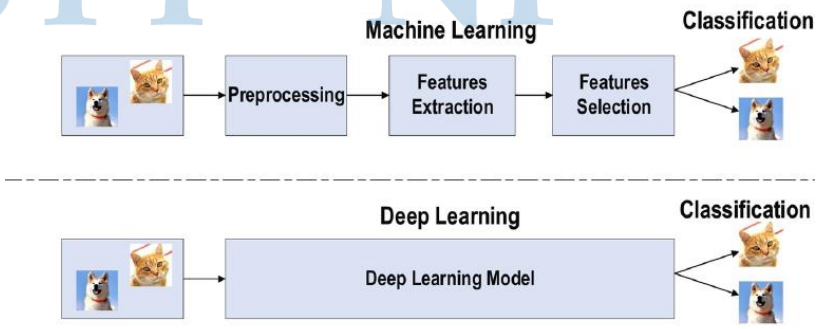
2.2 *Deep Learning*

Deep learning merupakan sebuah disiplin pembelajaran mesin (*machine learning*) (lihat Gambar 2. 1) yang melibatkan penerapan jaringan syaraf tiruan untuk mengekstrak dan memahami pola dari data[8]. Penerapan jaringan syaraf tiruan tersebut terinspirasi dari pola pemrosesan informasi yang terjadi dalam otak manusia. Untuk melakukan proses tersebut, *deep learning* menggunakan sejumlah data besar yang kemudian digunakan untuk mengelompokkan *input* ke dalam label yang lebih spesifik[9].



Gambar 2. 1 Hubungan artificial intelligence, machine learning, dan deep learning[9]

Sebelum menggunakan *deep learning,* proses klasifikasi gambar dilakukan dengan menggunakan *machine learning* konvensional yang memerlukan beberapa urutan tahapan. Urutan tersebut terdiri dari pra-pemrosesan (*pre-processing*), ekstraksi fitur (*feature extraction*), pemilihan fitur yang tepat (*feature selection*), pembelajaran (*learning*), dan klasifikasi (*classification*). Pada tahapan-tahapan tersebut, *feature selection* memiliki dampak yang besar terhadap performa dari teknik pembelajaran mesin konvensional tersebut. Oleh karena itu, bias yang terjadi pada tahap *feature selection* dapat menyebabkan terjadinya kekeliruan ataupun kesalahan pada hasil klasifikasi antar kelas[9]. Sebaliknya, *deep learning* memiliki kemampuan untuk melakukan otomasi pada pembelajaran fitur (*feature learning*) dalam melakukan beberapa tugas, tidak seperti metode *machine learning* konvensional[9], [10].



Gambar 2. 2 Perbedaan antara deep learning dan machine learning konvensional[9]

Dalam proses memahami pola dan mengolah data, *deep learning* berfokus pada penggunaan arsitektur jaringan syaraf tiruan (*deep neural networks*) untuk memahami dan memproses data yang kompleks. Arsitektur yang digunakan memiliki beberapa *layer* meliputi *input layer* dan *output layer*. Masing-masing *layer* memberikan interpretasi yang berbeda-beda terhadap data yang diberikan kepada *layer-layer* tersebut[11].

Metode *deep learning* dibagi menjadi tiga kategori: *unsupervised, semi-unsupervised,* dan *supervised*. Selain itu, *deep reinforcement learning* (DRL), atau biasa disebut *reinforcement learning* (RL) merupakan metode pembelajaran lainnya yang masuk ke dalam kategori antara *supervised* dan *unsupervised*[9]. *Reinforcement learning* bekerja dengan melakukan interaksi dengan *environment*, sedangkan *supervised* bekerja dengan mengandalkan label data sampel yang diberikan. Sebaliknya, metode *unsupervised* bekerja tanpa perlu mengandalkan label data.

Metode yang digunakan pada penelitian ini adalah *supervised learning. Supervised learning* merupakan metode yang bekerja dengan menggunakan data yang sudah memiliki label. Metode ini memiliki sekumpulan *input* dan resultan *output* $(x^t, y^t) \sim \rho$[9]. Misalnya, model akan memprediksi $y^t = f(x_t)$ jika *input* yang diberikan adalah x_t dan akan menghasilkan nilai *loss* yaitu $l(y^t, y_t)$[9]. Kemudian, parameter jaringannya akan terus melakukan *update* agar menghasilkan estimasi hasil yang lebih baik untuk *output* yang tepat dan sesuai.

Pada *deep supervised learning* ini, terdapat beberapa teknik, diantaranya *Recurrent neural networks* (RNN), *Convolutional neural networks* (CNN), dan *Deep Neural Networks* (DNN). Teknik DNN lebih bersifat umum sehingga dapat digunakan untuk bidang yang lebih luas, termasuk pengenalan gambar, pengenalan suara, maupun pemrosesan bahasa alami. Sedangkan teknik RNN umumnya sering digunakan dalam pengenalan suara dan bidang pemrosesan bahasa alami (*Natural Language Processing*). Sebaliknya, teknik CNN lebih dikhususkan untuk bidang pengenalan gambar.

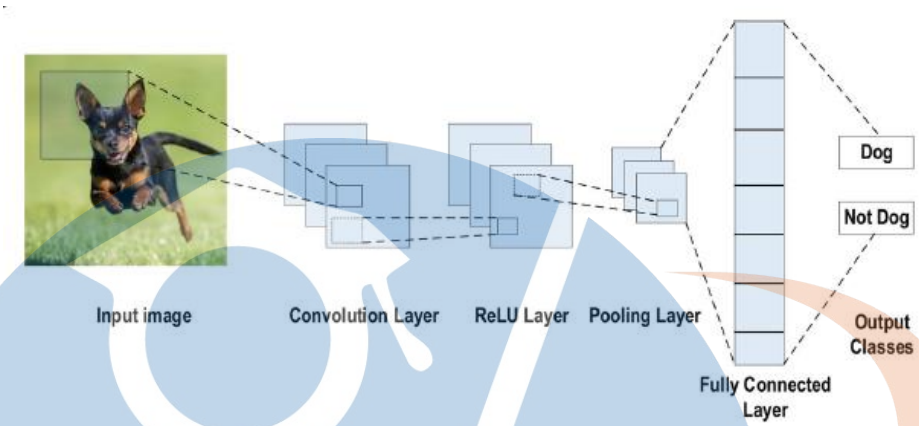
2.3 *Convolutional Neural Network (CNN)*

Jaringan saraf tiruan *multilayer* atau yang disebut *Convolutional Neural Network* (CNN) dirancang khususnya untuk mengekstrak informasi dari sebuah citra atau gambar[12]. CNN merupakan salah satu pendekatan *deep learning* yang dapat digunakan untuk skenario *computer vision*[13] seperti deteksi objek atau area pada gambar[14], serta klasifikasi gambar dan klasifikasi video[15]. Cara kerja CNN yaitu memanfaatkan proses konvolusi. Proses konvolusi merupakan sebuah proses pada CNN di mana terdapat sebuah *kernel* konvolusi (filter) berukuran tertentu yang bergerak pada sebuah gambar.

Mirip dengan jaringan syaraf konvensional, struktur CNN terinspirasi dari jaringan syaraf (neuron) pada otak manusia dan hewan. Lebih tepatnya, CNN menyimulasikan urutan sel kompleks yang membentuk korteks visual dalam otak kucing[16]. Pada sel korteks visual, hanya memproses sebagian kecil dari sebuah citra, bukan keseluruhan citra. Sel-sel tersebut secara spasial mengekstrak korelasi lokal yang terdapat di dalam *input*, seperti filter lokal pada *input*. Hal ini sama seperti pada CNN. *Weights* dan *local connection* pada CNN digunakan untuk mengoptimalkan semua struktur data *input* 2D seperti halnya sinyal citra. Operasi ini memanfaatkan sejumlah parameter yang sangat kecil, sehingga menyederhanakan proses *training* dan mempercepat jaringan[9].

Mirip dengan *multi-layer perceptron* (MLP), arsitektur CNN (Gambar 2. 3) terdiri dari banyak *layer* konvolusi (*convolution layer*) sebelum *layer* sub-sampling (*pooling layer*) dan diakhiri dengan *layer fully connected* (FC *layer*). Terdapat tiga jenis *layer* yang berbeda dalam CNN: *layer hidden*, *layer output*, dan *layer input*. Jumlah *hidden layer* yang digunakan bervariasi, tergantung pada arsitektur yang digunakan[17]. *Input* x dari masing-masing *layer* dibentuk oleh *array* tiga dimensi yang terdiri dari lebar (*width*), tinggi (*height*), dan kedalaman (*depth*) atau $m \times m \times r$, dimana lebar dan tinggi (m) memiliki ukuran yang sama dengan lebar. Kedalaman (r) menyatakan jumlah *channel* warna dalam sebuah gambar atau jumlah filter, misalnya pada sebuah gambar

RGB, maka r sama dengan tiga. Lalu lebar dan tinggi menyatakan ukuran matriks.



Gambar 2. 3 Contoh arsitektur CNN pada klasifikasi gambar[9]

Pada setiap *layer* konvolusi terdapat beberapa *kernel* (filter) yang dilambangkan dengan huruf k yang juga memiliki *array* tiga dimensi $(n \times n \times q)$, sama sepeti *input* gambar. Untuk nilai n harus lebih kecil dari m dan nilai q sama dengan atau kecil dari r. *Kernel-kernel* tersebut merupakan basis dari *local connection* yang saling berbagi parameter-parameter yang sama (bias b^k dan *weight* W^k) untuk menghasilkan k *feature maps* (h^k) dengan masing-masing memiliki ukuran $(m - n - 1)$ dan berkonvolusi dengan *input*. *Layer* konvolusi akan menghitung sebuah *dot product* dari *input* dan *weights* seperti pada persamaan (2. 1). Selanjutnya dengan menerapkan non-linearitas atau fungsi aktivasi pada *output layer* konvolusi, maka kita memperoleh:

$$h^k = f(W^k * x + b^k) \quad (2.\ 1)$$

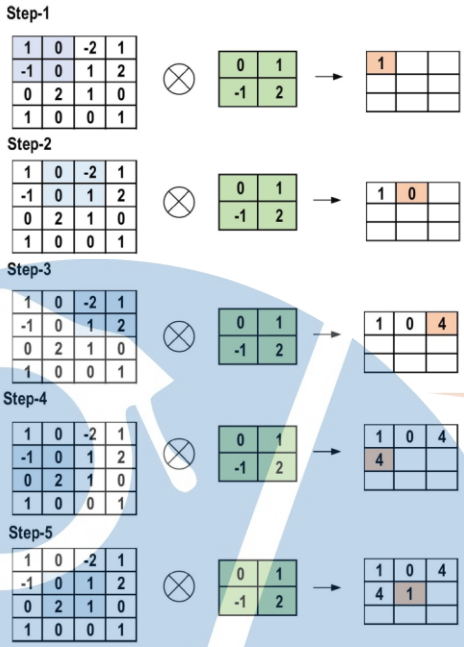
Kemudian, untuk setiap *feature maps* pada *pooling layer* akan dilakukan *down-sampling*. Oleh karena itu, parameter jaringan akan berkurang, sehingga akan mempercepat proses pelatihan dan memungkinkan untuk mengatasi masalah *overfitting*. Selanjutnya akan digunakan fungsi *pooling* untuk semua *feature map* pada area yang berdekatan dengan ukuran $p \times p$, dimana p merupakan ukuran *kernel*. Lalu, setiap *layer* FC akan menerima *feature* tingkat menengah

dan rendah, serta membuat abstraksi tingkat tinggi. Hal tersebut mewakili *layer* tahap terakhir seperti jaringan saraf pada umumnya. Dengan demikian, skor klasifikasi dihasilkan dengan menggunakan *layer* akhir (misalnya *Support Vector Machines* (SVM) atau *softmax*). Setiap skor merepresentasikan probabilitas kelas tertentu[9].

2.3.1 *Convolution Layer*

Convolution layer atau *layer* konvolusi merupakan *layer* yang paling utama dari CNN. Pada *layer* ini terdapat filter konvolusi yang tersusun dari matriks nilai-nilai bilangan diskrit (biasa disebut *kernel*). Pada tahap awal dari proses *training* pada CNN, akan diberikan kumpulan bilangan acak yang menjadi nilai dari *weights* pada *kernel*. Kemudian, pada setiap proses *training*, *weigths* tersebut akan terus diperbarui dan disesuaikan. Dengan demikian *kernel* akan memperoleh fitur-fitur yang penting dari gambar *input*.

Proses konvolusi dilakukan dengan cara *kernel* akan berkonvolusi dengan *input* gambar dengan cara bergerak pada seluruh bagian gambar secara horizontal dan vertikal. Pada setiap pergeseran akan dilakukan perhitungan *dot product* (Gambar 2. 4) antara *input* dan *kernel*, yang kemudian menghasilkan *output feature map*[9]



Gambar 2. 4 Perhitungan pada setiap proses dari layer konvolusi[9]

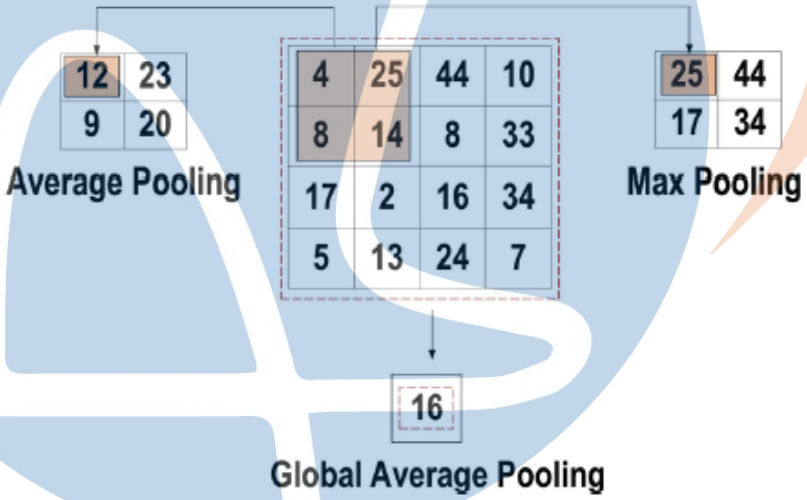
2.3.2 *Pooling Layer*

Pada *pooling layer* dilakukan proses *sub-sampling* pada *feature maps*. Proses ini dilakukan dengan tujuan untuk mereduksi dimensi dari *feature maps* yang diperoleh dari proses konvolusi sebelumnya. Manfaat dari *pooling layer* ini data yang digunakan menjadi mudah diolah dan mudah untuk mengontrol *overfitting*[18]. Hal tersebut dikarenakan proses *pooling* layer merepresentasikan data menjadi lebih kecil. Walaupun ukurannya diperkecil, informasi penting yang ada dalam *feature maps* tetap ada dalam setiap proses *pooling*[9]. Lebih tepatnya, *pooling layer* dapat dideskripsikan dengan dua parameter, yaitu luas spasial (*spatial extent*) dilambangkan dengan e dan langkah *(strides)* dilambangkan dengan s. Dimensi lebar (w_{out}) dan tinggi (h_{out}) yang dihasilkan untuk setiap *feature map* terlihat pada persamaan berikut[19].

$$w_{out} = \left[\frac{w_{in} - e}{s}\right] + 1 \quad (2.\ 2)$$

$$h_{out} = \left[\frac{h_{in} - e}{s}\right] + 1 \quad (2.\ 3)$$

Fungsi *pooling layer* juga untuk mempercepat komputasi tanpa harus menghilangkan informasi penting dari piksel gambar. Jenis *pooling layer* terdiri dari *tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling* (GAP), dan *global max pooling*. Metode *pooling* yang paling banyak digunakan yaitu *max pooling*[18].



Gambar 2. 5 Tiga jenis operasi pooling [9]

Metode *max pooling* dilakukan dengan memilih nilai maksimum yang ada pada suatu area tertentu (lihat Gambar 2. 5). Pada *max pooling*, dari empat nilai pada bagian kiri atas akan dihasilkan satu nilai maksimum, yaitu 25 sebagai hasil *pooling*. Cara yang sama juga dilakukan untuk empat nilai pada bagian kanan atas, kiri bawah, dan kanan bawah. Di mana akan menghasilkan data hasil *pooling* yaitu 25, 44, 17, dan 34. Dibandingkan dengan teknik *pooling* lainnya, teknik *max pooling* menghasilkan performa yang lebih baik[18].

2.3.3 *Activation Function*

Activation function (fungsi aktivasi) merupakan komponen penting dari algoritma CNN[20], [21], [22]. *Activation function* digunakan untuk melakukan aktivasi (mengaktifkan) fitur neuron dengan tujuan menyelesaikan masalah non-linear[21], [22]. Tanpa adanya *activation function*, *neural networks* hanya dapat mempelajari hubungan linear antara *input* dan *output* yang diinginkan[20]. Sifat non-linear dari *activation function* ini memastikan bahwa *mapping input* dan *output* bersifat non-linear, sehingga memungkinkan CNN untuk mempelajari pola yang lebih kompleks[9]. Berikut beberapa *activation function* yang umum digunakan[9], [20], [22].

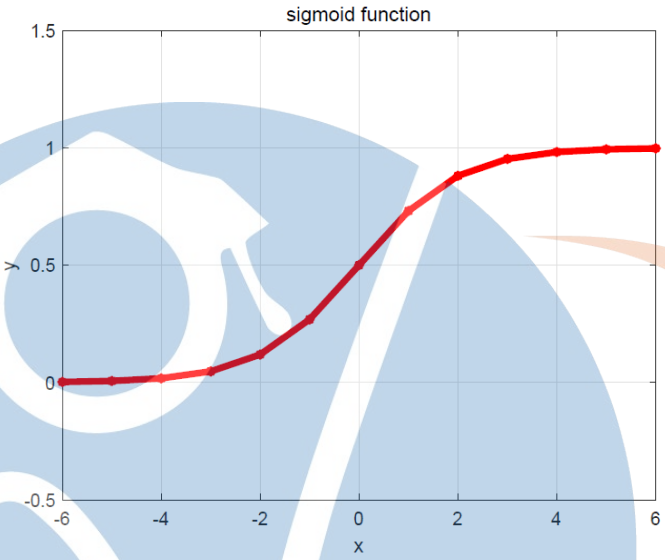
1. ***Sigmoid***

   Sigmoid pada dasarnya dirancang untuk *binary classification* akan tetapi saat ini telah memiliki penerapan yang luas dalam berbagai topik yang berkaitan dengan *attention model* dan juga *bounded output regression*[20]. *Input* dari *activation function* ini adalah berupa bilangan riil, sedangkan *output*-nya dibatasi antara 0 dan 1[9]. Secara matematis, *sigmoid* dapat direpresentasikan seperti berikut[9], [20].

$$sigmoid(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.\ 4)$$

   Pada persamaan (2. 4), parameter x merepresentasikan *input* data pada *sigmoid activation function*. Kurva *sigmoid* dapat dilihat pada Gambar 2. 6[22]. Dari kurva *sigmoid function* tersebut, terlihat bahwa *sigmoid* memiliki karakteristik *saturability* yang tidak terlalu tinggi[22]. Artinya, kemiringan grafik cenderung nol ketika *input*-nya sangat besar atau sangat kecil. Ketika kemiringan fungsi mendekati nol, gradien yang diteruskan ke jaringan yang mendasarinya menjadi sangat kecil[19], [22]. Hal ini akan membuat parameter jaringan sulit untuk dilatih secara efektif. Sementara itu, kecenderungan nilai bobot yang hanya berubah ke satu arah dikarenakan *output* dari fungsi ini selalu bernilai positif akan

mempengaruhi tingkat konvergensi[22]. Dikarenakan masalah tersebut, *sigmoid activation function* sudah tidak sepopuler seperti sebelumnya[23].



Gambar 2. 6 Kurva sigmoid activation function [22]

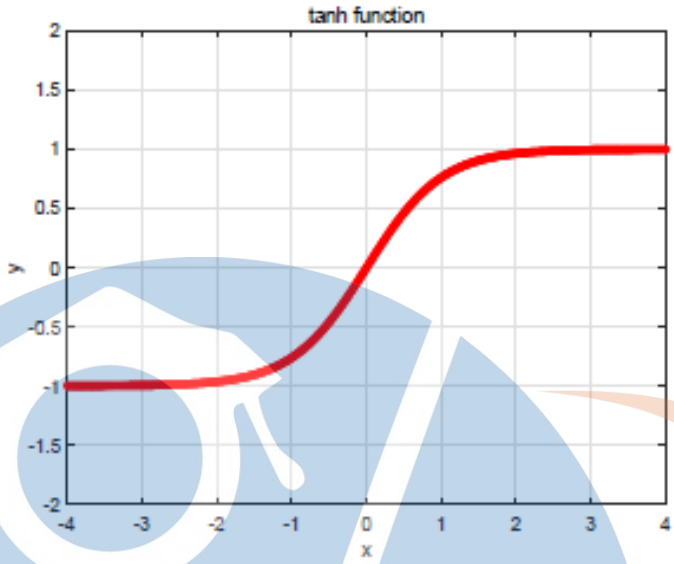
2. *Tanh*

Tanh activation function mirip dengan *sigmoid* dan merupakan versi pembaharuan dari *sigmoid*, sering juga disebut *symmetric sigmoid*[9], [20], [22], [23]. Seperti *sigmoid*, *input* dari *tanh activation function* adalah bilangan riil, di mana *output*-nya dibatasi antar -1 dan 1[9][19]. Secara matematis, *tanh function* dapat didefinisikan seperti persamaan (2. 5) berikut[9], [22], [23].

Terlihat pada Gambar 2. 7 bahwa tingkat konvergensi *tanh function* lebih tinggi dibandingkan dengan *sigmoid function*[22].

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2.\ 5)$$

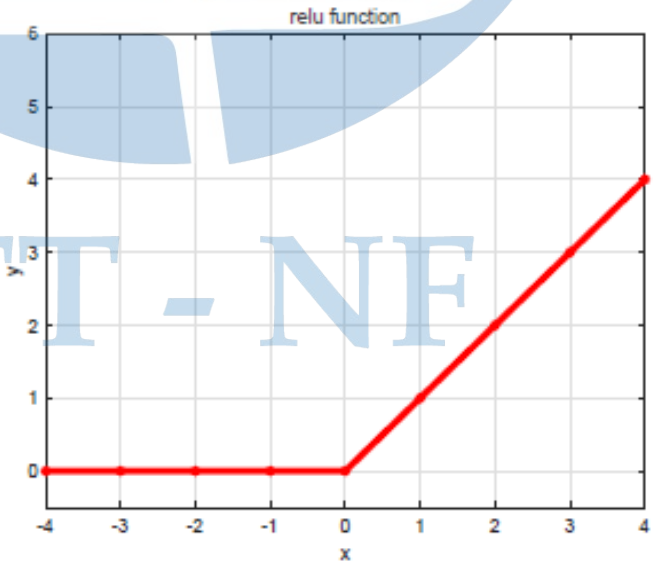
Hal tersebut berarti bahwa proses pembelajaran tetap berlangsung pada nilai negatif karena rentang nilai fungsi garis kurva hiperbolik[23]. Namun, permasalahan mengenai *gradient diffusion* tetap ditemukan[22]



Gambar 2. 7 Kurva tanh activation function[22]

3. *ReLU*

ReLU merupakan *activation function* yang paling umum digunakan dalam implementasi CNN[9], [22]. *ReLU* merupakan sebuah non linear *activation function* yang dapat melakukan operasi *derivative*. Pada kondisi tertentu, jaringan yang dilengkapi dengan *ReLU activation function* akan selalu konvergen menuju kondisi yang stabil[20].



Gambar 2. 8 Kurva ReLU activation function[22]

Pada kurva yang ditampilkan pada Gambar 2. 8, dapat dilihat bahwa *ReLU function* ini akan memaksa *output* menjadi nol jika nilai *input* kurang dari atau sama dengan nol. Jika tidak, maka nilai *output* akan sama dengan nilai *input*[22]. Metode ini dapat menciptakan karakteristik yang jarang sampai batas tertentu. Secara matematis, *ReLU* didefinisikan seperti persamaan (2. 6)[9], [20], [22], [23].

$$ReLU(X) = max(0, x) = \begin{cases} x & jika\ x \geq 0 \\ 0 & jika\ x < 0 \end{cases} \quad (2.\ 6)$$

Dibandingkan dengan dua fungsi sebelumnya, *ReLU function* memberikan tingkat komputasi yang jauh lebih cepat[9], [22]. Berikut beberapa alasan yang menjadi keuntungan dari *ReLU finction*[20].

1) *Gradient stability*

   Gradien dari *ReLU* (ketika *input*-nya positif) akan selalu satu, dengan demikian gradien tersebut tidak akan terhapus. Oleh karena itu, kasus di mana gradien *ReLU* mati akan terjadi ketika semua *output ReLU* bernilai nol pada *hidden layer*.

2) Secara Komputasi lebih murah

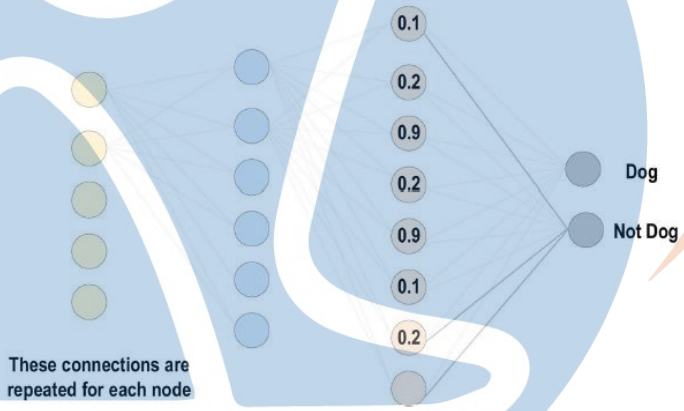
   Berbeda dengan *Activation function* lain yang membutuhkan evaluasi eksponen dan melakukan operasi pembagian, *ReLU* lebih sederhana. Hal tersebut dikarenakan *ReLU* hanya membutuhkan sebuah operasi maksimum untuk menghasilkan *output*. Hal yang sama juga berlaku dalam perhitungan gradien. Karena hal tersebutlah, *ReLU* lebih disukai daripada *activation function* kompleks lainnya.

3) *Sparsity*

   ReLU menghasilkan kerenggangan pada layer karena jika input menjadi nol, maka koneksi menjadi tidak relevan dengan model. Hal ini memungkinkan untuk menganalisis fitur atau variabel yang penting.

2.3.4 *Fully Connected Layer*

Umumnya, *fully connected layer* ini berada di akhir dari arsitektur CNN. Pada *layer* ini, setiap neuron terhubung dengan semua neuron yang ada pada *layer* sebelumnya. Oleh karena itu, *layer* ini disebut pendekatan *Fully Connected* (FC). *Input* dari FC *layer* ini didapatkan dari *pooling layer* maupun *convolution layer* yang terakhir. *Input* ini berbentuk sebuah vektor, yang mana berasal dari hasil *flattening* terhadap *feature maps* menjadi vektor satu dimensi. *Output* dari FC *layer* ini merepresentasikan *output* akhir dari CNN, seperti yang terlihat pada Gambar 2. 9 berikut [9]. *Layer fully connected* yang paling akhir memiliki jumlah unit yang sama dengan banyak kelas yang digunakan[24].



Gambar 2. 9 Fully connected layer[9]

2.3.5 *Loss Function*

Hasil akhir klasifikasi akan tercapai dari *output layer*, yang mana merupakan *layer* terakhir dari arsitektur CNN. *Loss function* digunakan pada *output layer* ini untuk menghitung prediksi *error* yang dibuat pada seluruh sampel *training* pada CNN model[9]. *Loss function* digunakan selama proses *training* untuk mengoptimalkan parameter model. Fungsi ini akan mengukur perbedaan antara hasil prediksi oleh model dengan hasil yang sebenarnya, dengan tujuan meminimalkan perbedaan hasil tersebut[25]. Memilih *loss function* yang tepat untuk model sangat penting karena setiap *loss function* akan memiliki hasil yang berbeda pada evaluasi yang sama[26]

Pada metode klasifikasi, terdapat beberapa jenis klasifikasi dengan *loss function* yang berbeda sesuai permasalahan dan algoritmanya[9], [25]. Berikut beberapa *loss function* yang umum digunakan[9], [25], [26].

1. *Binary Cross-Entropy Loss*

Binary Cross Entropy (BCE) atau biasa dikenal sebagai *log loss*, merupakan *loss function* yang biasa digunakan untuk permasalahan *binary classification*. Fungsi ini mengukur perbedaan antara probabilitas yang diprediksi dari sebuah kelas dan label kelas yang sebenarnya. *Cross-entropy* merupakan konsep yang sangat dikenal dalam teori informasi dan biasa digunakan untuk mengukur perbedaan antara dua distribusi probabilitas. Dalam *binary classification*, kelas sebenarnya biasanya diwakili oleh *one-hot encode* vektor, di mana kelas sebenarnya memiliki nilai 1, sedangkan kelas lainnya memiliki nilai 0. Probabilitas dari kelas yang sebenarnya dilambangkan dengan $p(y = 1|x)$ dan probabilitas prediksi dari kelas lainnya dilambangkan dengan $p(y = 0|x)$.

$$L(y,p) = -(y\ log(p) + (1-y)\, log(1-p)) \tag{2. 7}$$

Secara matematis *loss function* dapat didefinisikan pada persamaan (2. 7), yang mana dapat dipisahkan menjadi dua bagian seperti persamaan (2. 8) berikut.

$$\begin{cases} -log(p) & jika\ y = 1 \\ -log(1-p) & jika\ y = 0 \end{cases} \tag{2. 8}$$

Di mana y merupakan label kelas sebenarnya (0 atau 1) dan p merupakan probabilitas prediksi dari kelas positif. *Loss function* akan dikurangi ketika probabilitas prediksi p sama dengan label kelas sebenarnya y. *Loss* dihitung untuk setiap sampel dan dirata-ratakan pada setiap *dataset*.

2. *Hinge Loss*

Hinge *loss* umumnya digunakan pada permasalahan yang berhubungan dengan *binary classification*. Fungsi ini merupakan fungsi yang populer

digunakan pada klasifikasi *maximum-minimum*, digunakan untuk *support vector machines* (SVMs) misalnya pada klasifikasi *one-vs-all* di mana dilakukan klasifikasi terhadap sebuah sampel sebagai bagian dari salah satu dari banyak kategori dan keadaan di mana ditetapkan *margin of error*. Secara matematis dapat didefinisikan seperti berikut.

$$H(p, y) = \sum_{i=1}^{N} \max(0, m - (2y_i - 1)p_i) \quad (2.\ 9)$$

Margin m biasanya ditetapkan menjadi 1. Selain itu, prediksi *output* dilambangkan sebagai p_i , sedangkan *output* yang diinginkan dilambangkan sebagai y_i.

3. ***Categorical Cross-entropy Loss* (CCE)**

Categorical Cross Entropy (CCE) atau biasa dikenal dengan *negative log-likelihood loss* atau *multi-class log loss* atau *Sofmax Loss Function*, merupakan sebuah fungsi yang digunakan untuk tugas klasifikasi *multi-class*. Fungsi ini mengukur perbedaan antara probabilitas distribusi prediksi dan distribusi kelas sebenarnya. *Output* dari fungsi ini adalah probabilitas $p \in \{0,1\}$. Pada *output layer*, digunakan aktivasi softmax untuk menghasilkan *output* dalam distribusi probabilitas[9]. Secara matematis, probabilitas dari kelas *output* dapat didefinisikan sebagai berikut.

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \quad (2.\ 10)$$

Di mana, e^{a_i} merepresentasikan *output non-normalized* dari *layer* yang sebelumnya. Selain itu, N merepresentasikan jumlah dari neuron yang ada

pada *output layer*. Oleh karena itu, secara matematis *cross-entropy loss function* dapat didefinisikan seperti persamaan berikut.

$$H(p,y) = -\sum_{i} y_i \log(p_i) \qquad dimana \ \ i \in [1,N] \tag{2. 11}$$

Label yang sebenarnya merupakan *one-hot encoded* vektor pada *loss function categorical cross-entropy* konvensional. Artinya, elemen yang sesuai dengan kelas sebenarnya adalah 1 dan elemen lainnya adalah 0.

2.3.6 *Regularization*

Pada model CNN, *overfitting* merepresentasikan permasalahan penting yang berkaitan dengan upaya memperoleh generalisasi yang memiliki performa yang baik. Model dapat dikatakan *overfitting* apabila model bekerja dengan sangat baik pada data *training* dan tidak bekerja dengan baik pada data *testing* (data yang belum pernah diberikan kepada model). Model dikatakan "*just-fitted*" apabila model dapat berjalan dengan baik pada data *training* maupun data *testing*. Berbagai konsep pendekatan intuitif digunakan untuk membantu *regularization* dalam menghindari *overfitting*.

1. ***Batch Normalization Layer***

   Pada tahun 2015, peneliti dari *Google* merancang cara yang menarik untuk lebih mempercepat *training feed-forward* dan CNN dengan menggunakan teknik yang disebut *batch normalization*[19]. Pada *deep neural networks*, distribusi *input* dari *hidden layer* akan berubah secara konstan. Fenomena ini dianggap sebagai pergeseran kovarian internal. Umumnya, distribusi baru secara bertahap mendekati batas atas dan batas bawah dari interval fungsi aktivasi, yang dapat menyebabkan pengurangan dan hilangnya gradien dari *hidden layers* selama *backpropagation*. Itulah mengapa konvergensi dalam *deep neural networks* semakin lambat[27]. Mengatasi permasalahan tersebut akan memungkinkan proses *training* menjadi lebih cepat.

Batch normalization dapat mengubah distribusi *input* menjadi distribusi normal standar dengan *mean* 0 dan *varians* 1[27].

$$y = \frac{x - E[x]}{\sqrt{V\ ar[x] + \epsilon}} * \gamma + \beta \quad (2.\ 12)$$

Konversi ini membuat distribusi berada dalam interval sensitif fungsi aktivasi, yang berarti perubahan kecil pada *input* dapat menyebabkan perubahan besar pada *loss function*. Hal ini membuat gradien menjadi lebih besar, sehingga menghindari masalah dispersi gradien. Gradien yang lebih besar berarti konvergensi *deep neural networks* menjadi lebih cepat, yang mana dapat sangat mempercepat proses *training*[27]. Oleh karena itu, normalisasi terhadap *input* gambar membantu *training* dengan membuat model menjadi lebih peka terhadap variasi[19]. Manfaat dari penggunaan *batch normalization* adalah sebagai berikut[9].

- Mencegah timbulnya masalah gradien yang semakin menurun.
- Secara efektif dapat mengontrol inisialisasi *weight* yang buruk.
- Secara signifikan mengurangi waktu yang dibutuhkan untuk konvergensi jaringan.
- Membantu mengurangi ketergantungan *training* pada seluruh *hyperparameter.*
- Kemungkinan terjadinya *overfitting* akan berkurang.

2. *Drop Out*

Dropout merupakan jenis metode yang berbeda untuk mencegah *overfitting* dan telah menjadi salah satu metode yang paling umum digunakan dalam mencegah *overfitting* pada *deep neural networks*. Regularisasi dalam model pelatihan dapat diatasi dengan menggunakan *dropout layer* yang dapat mengurangi terjadinya terlalu banyak *error* saat *training*[28]. Saat proses *training*, *dropout* diimplementasikan dengan hanya membiarkan sebuah neuron tetap aktif dengan probabilitas p (*hyperparameter*), atau mengaturnya menjadi 0 jika tidak diaktifkan.

Secara intuitif, hal ini akan memaksa jaringan untuk tetap akurat meskipun tanpa adanya informasi tertentu[19]. Hal ini akan mencegah jaringan menjadi terlalu bergantung pada neuron mana pun atau kombinasi beberapa neuron[19], [28], [29].

Pada prinsipnya, setiap neuron memaksa neuron lain untuk terhubung setelah melewati aktivasi, tetapi hal tersebut menyebabkan *overfitting* dan *error* yang besar pada *training*. *Dropout* akan menonaktifkan jaringan secara acak yang terhubung ke neuron. Misalnya, jika $p = 0.5$, neuron harus membagi dua *output* pada saat *testing* untuk mendapatkan *output* yang diharapkan. Artinya, jika *output* dari neuron sebelum *dropout* adalah x, maka setelah *dropout*, *output* yang diharapkan adalah seperti persamaan berikut[19].

$$E[output] = px + (1 - p) \quad (2.13)$$

3. *Data Augmentation*

Data augmentation atau augmentasi data merupakan bagian dari berbagai teknik regularisasi yang bertujuan untuk meningkatkan kinerja model[30]. *Data augmentation* juga dideskripsikan sebagai strategi untuk mencegah *overfitting* melalui regularisasi. *Data augmentation* mencakup berbagai macam teknik yang digunakan untuk menghasilkan sampel *training* yang "baru" dari sampel *training* yang ada dengan menerapkan beberapa transformasi secara acak. Tujuan diterapkannya *data augmentation* ini adalah untuk meningkatkan generalisasi model (tetapi pada saat yang sama memastikan bahwa label kelas data tidak berubah)[31][9]. Berikut beberapa alternatif *data augmentation*.

- *Flipping*, gambar dapat dibalikkan (*flip*) secara horizontal maupun vertikal. Transformasi ini akan menghasilkan gambar yang diputar pada kelipatan 90 derajat. *Flipping* secara vertikal dapat dilakukan dengan memutar gambar pada 180 derajat[31]. Misalnya citra wajah

yang menghadap ke arah kanan dapat dibalik menjadi menghadap ke arah kiri.

- *Color space*, digunakan untuk mendapatkan variasi warna secara acak[18]. Melakukan augmentasi pada *channel* warna merupakan sebuah teknik alternatif yang sangat bisa untuk diimplementasikan[9]. Augmentasi warna yang sangat mudah dilakukan adalah memisahkan *channel* warna tertentu, misalnya merah, hijau, dan biru. Selain itu, meningkatkan dan mengurangi kecerahan gambar dapat diperoleh dengan menggunakan operasi metrik langsung untuk memanipulasi nilai RGB. Perubahan pencahayaan diperoleh dengan menyesuaikan nilai intensitas dalam histogram, serupa dengan yang digunakan dalam aplikasi penyuntingan foto.
- *Rotation*, ketika memutar gambar ke kiri maupun ke kanan dalam rentang 0 hingga 360 derajat di sekeliling sumbu, maka akan diperoleh augmentasi rotasi[9].
- *Translation*, untuk menghindari bias posisi dalam data gambar, transformasi yang dapat digunakan untuk masalah tersebut adalah menggeser gambar ke atas, ke bawah, ke kiri, atau ke kanan[9]. Untuk dapat mengidentifikasi objek di bagian mana pun dari gambar, maka konsep *translation* diterapkan pada gambar. Hal ini melibatkan pemindahan gambar di sepanjang arah X atau Y atau keduanya[31].

2.3.7 *Optimizer*

Optimizer digunakan untuk meminimalkan *error* antara data *training* dan data *validation*. Setelah *loss function* menghitung *error*, maka *optimizer* akan menghitung gradien, yang merupakan turunan dari *loss function*. Parameter jaringan harus selalu diperbarui melalui semua *training epochs*, sekaligus mencari jawaban yang paling optimal pada semua *training epochs* untuk meminimalkan *error*[9].

Sebagai ukuran langkah dalam melakukan *updating* parameter, digunakan *learning rate*. *Training epochs* merepresentasikan perulangan lengkap dari

proses pembaruan parameter yang melibatkan *dataset training* pada satu waktu. Pemilihan *learning rate* dapat mempengaruhi hasil akhir dari model[9].

Gradient Descent atau *Gradient-based learning algorithm* biasa digunakan untuk meminimalkan *error* pada *training*[9], [26], [32]. Gradien tersebut digunakan untuk mengatur *weights* dari model untuk memperoleh kinerja yang lebih baik[26]. Algoritma ini secara berulang meng-*update* paramete jaringan pada setiap *training epoch*. Secara spesifik, untuk melakukan *update* parameter dengan benar, perlu dilakukan perhitungan dari fungsi objektif gradien (kemiringan) dengan menerapkan *first-order derivative* sehubungan dengan parameter jaringan. Selanjutnya, parameter akan di-*update* dalam arah yang berlawanan dengan gradien untuk mengurangi *error*. Secara matematis, operasi ini dapat direpresentasikan sebagai berikut[9].

$$w_{ij^t} = w_{ij^{t-1}} - \Delta w_{ij^t}, \qquad \Delta w_{ij^t} = \eta * \frac{\partial E}{\partial w_{ij}} \tag{2. 14}$$

Di mana w_{ij^t} merupakan final *weight* pada *training epoch* saat ini, sementara *weight* pada *training epoch* sebelumnya $(t-1)$ dilambangkan dengan $w_{ij^{t-1}}$. *Learning rate* dilambangkan dengan η dan *error* dari prediksi dilambangkan dengan E. Pemilihan *optimizer* yang sesuai menjadi hal yang sangat penting dalam menentukan akurasi[26]. Berikut beberapa alternatif dari *gradient-based learning algorithm* yang sering digunakan.

1. ***Momentum***

   Pada *neural networks*, teknik ini menerapkan fungsi objektif. Teknik ini akan meningkatkan kecepatan *training* dan *accuracy* dengan menjumlahkan total gradien dari *step training* sebelumnya, yang mana diberi bobot melalui sebuah faktor λ. Namun, algoritma ini hanya berhenti pada lokal minimum, bukan pada global minimum. Hal ini merupakan kelemahan utama dari algoritma *gradient-based learning*[9]. Secara matematis *momentum* dapat didefinisikan sebagai berikut.

$$\Delta w_{ij^t} = \left(\eta * \frac{\partial E}{\partial w_{ij}}\right) + (\lambda * \Delta w_{ij^{t-1}}) \quad (2.\ 15)$$

Kenaikan *weight* pada *training epoch* ke t' dilambangkan dengan Δw_{ij^t}, sedangkan *learning rate* dilambangkan dengan η, dan kenaikan *weight* pada *training epoch* sebelumnya $(t-1)$ dilambangkan dengan $\Delta w_{ij^{t-1}}$. Nilai dari faktor *momentum* berada dalam rentang 0 sampai 1[9], [19]. Ketika nilai faktor momentum menjadi sangat rendah, model akan kehilangan kemampuannya untuk menghindari lokal minimum. Sebaliknya, ketika nilai faktor momentum menjadi tinggi, maka model akan mengembangkan kemampuan untuk konvergen dengan lebih cepat. Jika nilai faktor momentum yang tinggi digunakan bersama dengan *learning rate*, maka model akan melampaui batas global minimum dengan melewatinya[9].

2. ***RMSProp***

RMSProp (*Root Mean Square Propagation*) adalah *optimizer* yang dirancang untuk mengatasi permasalahan mengenai penurunan *learning rate* yang terlalu cepat[26], [32], [33]. *RMSProp* menormalkan gradien dengan rata-rata eksponensial dari besarnya gradien untuk setiap parameter[26]. Lebih spesifik, hasil dari *update* akumulasi vektor gradien, secara matematis dapat terlihat seperti persamaan berikut ini[19].

$$r_i = \rho r_{i-1} + (1+\rho) g_i \odot g_i \quad (2.\ 16)$$

Faktor penurunan ρ menentukan berapa lama gradien lama akan disimpan. Semakin kecil faktor ρ maka semakin singkat jangka waktu efektifnya[19].

3. ***Adam***

Adam merupakan kombinasi antara *Momentum* dan *RMSProp*[9], [19], [26], [32], [33]. Hal yang ditingkatkan pada *Adam* adalah mempertimbangkan kelancaran varian gradien dan menyediakan mekanisme koreksi bias. *Adam* mengurangi biaya komputasi,

membutuhkan lebih sedikit memori, dan tidak bergantung pada penyesuaian diagonal gradien[34]. Berbeda dengan *RMSProp* yang melakukan pengoptimalan berbasis gradien yang menggunakan adaptif *learning rate* dari waktu ke waktu, *Adam* memperlakukan *learning rate* sebagai *hypeparameter*[34].

Secara sederhana, kita ingin menyimpan *weighted moving average* secara eksponensial dari gradien (pada dasarnya konsep *velocity* dalam momentum klasik), yang mana dapat didefinisikan sebagai berikut[19].

$$\boldsymbol{m_i} = \beta_1 \boldsymbol{m}_{i-1} + (1 - \beta_1)\boldsymbol{g_i} \tag{2. 17}$$

Ini yang disebut dengan pendekatan yang disebut *first moment* dari gradien atau $\mathbb{E}[g_i]$. Begitu juga dengan *RMSProp*, kita bisa memantau *weighted moving average* secara eksponensial dari riwayat gradien[19]. Ini merupakan estimasi dari yang kita sebut *second moment* dari gradien atau $\mathbb{E}[g_i \odot g_i]$:

$$\boldsymbol{v_i} = \beta_2 \boldsymbol{v}_{i-1} + (1 - \beta_2)\, \boldsymbol{g_i} \odot \boldsymbol{g_i} \tag{2. 18}$$

Namun, hasil estimasi tersebut memiliki bias terhadap keadaan yang sebenarnya karena kita mulai dengan menginisialisasi kedua vektor tersebut menjadi vektor nol. Untuk memperbaiki bias ini, dapat menggunakan pembaruan dari *Adam* yang telah dikoreksi untuk memperbarui vektor parameter, seperti berikut[19].

$$\theta_i = \theta_{i-1} - \frac{\epsilon}{\delta \oplus \sqrt{\widetilde{v}_i}} \widetilde{m}_i \tag{2. 19}$$

Belakangan ini, Adam menjadi populer karena upaya perbaikan terhadap bias inisialisasi nol dan kemampuannya untuk menggabungkan konsep inti di balik *RMSProp* dan *Momentum* secara lebih efektif. Satu-satunya

pengecualian adalah bahwa *learning rate* mungkin perlu dimodifikasi dalam kasus-kasus tertentu dari nilai *default* 0.001[19].

2.3.8 *Metrik*

Pada bidang *machine learning,* umumnya berfokus pada prediksi hasil berdasarkan data yang ada. Jika *output* yang dihasilkan adalah merepresentasikan kelas yang berbeda, maka hal itu disebut "*classification problem*". Jika *output* yang dihasilkan adalah pengukuran numerik, maka disebut dengan "*regression problem*". Pada klasifikasi biasa yang sering digunakan adalah dua kelas saja, sedangkan jika memiliki lebih banyak kelas disebut "*multi-class classification*". Secara prinsip, tugas prediksi diselesaikan dengan menggunakan teknik matematika. Solusi yang digunakan bisanya menggunakan faktor yang sama, yaitu menggunakan data yang tersedia (variabel X) untuk memperoleh hasil prediksi dari variabel Y[35].

Model klasifikasi memberikan probabilitas untuk ditetapkan dalam kelas tertentu untuk setiap unit yang mungkin. Dalam kasus *multi-class classification* penentuan kelas yang akan diprediksi untuk setiap unit ada berbagai kemungkinan, di antaranya nilai probabilitas tertinggi dan *softmax* adalah teknik yang paling banyak digunakan[35].

Indikator performa sangat bermanfaat ketika tujuan yang ingin dituju adalah untuk melakukan evaluasi dan membandingkan model klasifikasi yang berbeda atau teknik dari *machine learning* yang digunakan. Banyak metrik yang digunakan biasanya didasarkan pada *Confusion Matrix*[35]. Berikut penjelasan lebih lanjut.

1. ***Confusion Matrix***

   Confusion matrix digunakan untuk menentukan kinerja algoritme klasifikasi[25]. *Confusion matrix* terdiri dari jumlah *true positive* (TP), *true negative* (TN), *false positive* (FP), dan *false Negative* (FN) yang dihasilkan dari algoritma[25]. *Confusion matrix* untuk permasalahan *binary classification* dapat dilihat pada tabel di bawah ini.

Tabel 2. 1 *Confusion Matrix*

	Predicted Positive	***Predicted Negative***
Actual Positive	*True Positive (TP)*	*False Negative (FN)*
Actual Negative	*False Positive (FP)*	*True Negative (TN)*

Di mana:

- TP: jumlah gambar yang diklasifikasikan dengan benar sesuai label yang sebenarnya.
- TN: jumlah gambar yang diklasifikasikan dengan benar sebagai label yang bukan sebenarnya.
- FP: jumlah gambar yang diklasifikasikan dengan salah terhadap label yang sebenarnya.
- FN: jumlah gambar yang diklasifikasikan dengan salah sebagai label yang bukan sebernarnya.

Menggunakan nilai dari *confusion matrix* dapat menghitung metrik performa seperti *accuracy*, *precision*, *recall*, dan *F1-score.*

2. *Accuracy*

Accuracy merupakan metrik yang paling banyak digunakan untuk klasifikasi objek. *Accuracy* menghitung rasio sampel yang diklasifikasikan dengan benar terhadap jumlah total sampel[25]. Secara matematis, *accuracy* dapat direpresentasikan sebagai berikut [25], [35].

$$Acc = \frac{jumlah\ sampel\ yang\ diklasifikan\ dengan\ benar}{Total\ jumlah\ sampel} \quad (2.\ 20)$$

Atau jika dinyatakan dalam bentuk nilai *confusion matrix*, dapat didefinisikan sebagai berikut.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.\ 21)$$

Sesuai dengan persamaan (2. 21) di atas, metrik *accuracy* mempertimbangkan jumlah elemen *True Positive* dan *True Negative* dan

jumlah semua elemen *confusion matrix*. *True Positive* dan *True Negative* adalah elemen yang diklasifikasikan dengan benar oleh model dan berada pada diagonal utama dari *confusion matrix*, sedangkan penyebut merupakan semua elemen di luar diagonal utama yang diklasifikasikan secara tidak benar oleh model. Jadi, secara sederhana, *accuracy* merupakan probabilitas bahwa prediksi model benar[35].

2.4 *Image Classification*

Salah satu jenis implementasi *machine learning* adalah klasifikasi gambar (*image classification*), yang mengelompokkan gambar menjadi beberapa kelas sesuai dengan atribut yang ada di dalamnya. Gambar akan dikategorikan ke dalam kelas-kelas yang sudah ditentukan sebelumnya. Pada dasarnya, kelas adalah *tag* seperti "mobil", "hewan", "tanaman", atau item lainnya[36]. Umumnya, klasifikasi gambar/citra dilakukan dengan menggunakan dua metode (yang umum digunakan): klasifikasi *multi-class* dan klasifikasi *multi-label*.

Perbedaan utama antara klasifikasi *multi-class* dan klasifikasi *multi-label* adalah terletak pada bagaimana suatu citra dikelompokkan ke dalam suatu kelas. Setiap kelas dalam klasifikasi *multi-class* memiliki peran yang berbeda yang saling terpisah. Sebuah gambar hanya dapat menjadi bagian dari satu kelas, seperti tanaman atau hewan, bukan keduanya. Sebaliknya, setiap label dalam klasifikasi *multi-label* menunjukkan peran klasifikasi yang berbeda. Tetapi, karena peran ini saling berhubungan, maka keputusan yang sama akan diperoleh. Sebagai contoh, sebuah gambar dapat dikategorikan, misalnya, ke dalam kategori tanaman dan hewan. Metode klasifikasi *multi-class* akan membagi sampel ke suatu kelas yang telah ditentukan dan memprediksi kelas sampel berdasarkan karakteristik yang ada dalam sampel untuk memecahkan masalah. Namun, untuk menyelesaikan masalah klasifikasi *multi-label*, algoritma membagi sampel ke dalam beberapa kelas yang telah ditentukan

dengan memprediksi kelas sampel berdasarkan karakteristik yang ada dalam sampel[37].

2.5 *TensorFlow*

TensorFlow merupakan salah satu penyedia layanan *open source end-to-end,* digunakan pada *machine learning*, dan *deep neural network*. *TensorFlow* memiliki ekosistem *tools, library*, dan sumber daya komunitas yang luas dan fleksibel, sehingga memungkinkan para developer untuk membuat dan mengimplementasikan aplikasi berbasis *machine learning* dengan mudah dan menjaga para peneliti di bidang ini tetap kompetitif. *Tensorflow* sendiri dibangun dan dikembangkan oleh tim *Google Brain*, bersama dengan teknologi canggih yang berada di balik pengenalan gambar *Google Photos* dan pengenalan suara *Google Now*.

Kapasitas untuk mendefinisikan, menghitung, dan mengoptimalkan persamaan matematika yang melibatkan *array* multidimensi dengan efisien, penggunaan GPU yang transparan (manajemen komputasi), dan *scalability* komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar adalah beberapa dari sekian banyak manfaat *TensorFlow* untuk pembelajaran mesin dan pengembangan jaringan syaraf tiruan. Penggunaan GPU yang transparan (manajemen komputasi) dan *scalability* komputasi yang sangat baik di seluruh mesin dan kumpulan data yang besar adalah fitur-fitur menarik lainnya yang ditawarkan oleh *TensorFlow*[37].

2.6 Penelitian Terkait

Tabel 2. 2 Penelitian terkait

No	Nama dan Tahun	Judul	Topik	Subjek	Hasil
1	Randy Efan Jayaputra, 2020 [38]	Estimasi Kalori pada makanan melalui Citra Makanan menggunakan Model *SSD (Single Shot Detector)* pada *Mobile Device*	*Machine Learning*	Citra Makanan	Aplikasi *Mobile* untuk memperkir a-kan kalori makanan
2	I Putu Eka Dharma Udayana, dan Putu Gede Surya Cipta Nugraha. 2020 [39]	Prediksi Citra Makanan Menggunakan *Convolutional Neural Network* untuk Menentukan Besaran Kalori Makanan	*Machine Learning*	Citra Makanan	Sistem yang dapat mengenali gambar makanan
3	Wicakson, Septi Andryana, Benrahma, 2020 [40]	Aplikasi Pendeteksi Penyakit Pada Daun Tanaman Apel Dengan Metode *Convolutional Neural Network*	*Machine Learning*	Citra Daun Tanaman Apel	Aplikasi untuk mendeteksi penyakit apel dengan menggunak an citra daun apel.

1) Penelitian dilakukan oleh Randy Evan Jayaputra (2020) yang berjudul "Estimasi Kalori pada makanan melalui Citra Makanan menggunakan Model *SSD (Single Shot Detector)* pada *Mobile Device*". Penelitian ini berfokus pada pengembangan Aplikasi mobile untuk mengidentifikasi makanan menggunakan model *SSD (Single Shot Detector)* dan makanan yang berhasil diidentifikasi akan diestimasi kalorinya oleh aplikasi *mobile*. Informasi kalori per porsi yang disediakan oleh *FatSecret* Indonesia digunakan untuk memperkirakan kalori. Penelitian ini memiliki hasil akhir model yang bekerja dengan sangat baik dalam mendeteksi jenis makanan di aplikasi *mobile*, dengan akurasi rata-rata keseluruhan dan akurasi lebih dari 85% untuk sembilan jenis makanan (kategori). Aplikasi *mobile* dari hasil penelitian ini juga dapat menampilkan estimasi kalori makanan secara otomatis ketika jenis makanan terdeteksi[38].
2) Penelitian yang dilakukan oleh I Putu Eka Dharma Udayana, dan Putu Gede Surya Cipta Nugraha (2020) yang berjudul "Prediksi Citra Makanan Menggunakan *Convolutional Neural Network* untuk Menentukan Besaran Kalori Makanan". Dalam penelitian ini disebutkan bahwa metode klasifikasi menggunakan *Convolutional Neural Network (CNN)* sangat andal dalam menentukan keakuratan klasifikasi citra makanan. Hal ini dibuktikan dengan hasil akurasi sebesar 66% hingga 98%. Dengan data *training* yang optimal, hasil klasifikasi yang baik dapat dicapai dengan *subset* dari data *training*. Pada pengujian untuk menentukan kandungan kalori makanan, sistem mampu menampilkan kalori secara akurat berdasarkan data dari *database* kalori sistem. Bahkan, sistem mampu mengenali gambar makanan secara akurat, sehingga hanya perlu membandingkan nama makanan yang dikenali dengan *database* kalori sistem[39].

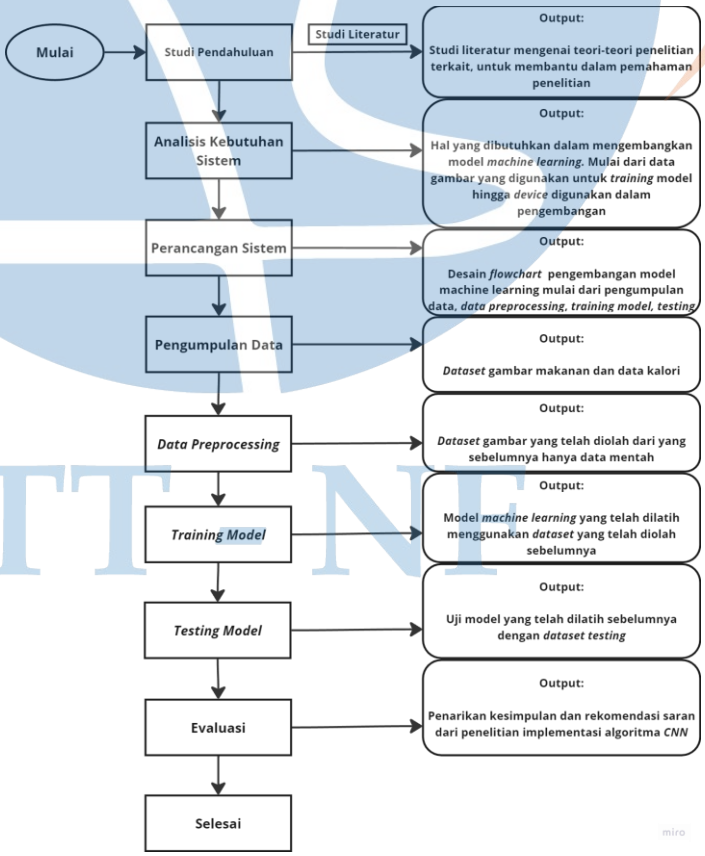
3) Penelitian yang dilakukan oleh Wicaksono, Septi Adryana, dan Benrahman (2020) yang berjudul "Aplikasi Pendeteksi Penyakit Pada Daun Tanaman Apel Dengan Metode *Convolutional Neural Network*", menggunakan metode CNN dengan arsitektur LeNet-5 yang dapat memproses 3151 data gambar dengan akurasi minimal 75%. Tahapan CNN meliputi *Convolutional Layers, Rectified Linear Units (ReLUs), Sub-Sampling, Flattening,* dan *Fully Connected Layers*. Hasil pengujian dievaluasi dengan menggunakan *image data testing*. Proses evaluasi dilakukan dengan menggunakan *confusion matrix*. Aplikasi yang dirancang teruji dengan akurasi model sebesar 99.4% dan akurasi validasi sebesar 97.8%, yang mengindikasikan bahwa aplikasi dapat digunakan untuk mendeteksi penyakit apel dengan menggunakan citra daun apel.

BAB III
METODOLOGI PENELITIAN

Pada Bab III, berisi pembahasan perihal tahapan penelitian mulai dari studi Pustaka sampai penarikan kesimpulan, serta rancangan penelitian yang mencakup penjelasan dan temuan dari kegiatan yang berkaitan dengan analisis lingkungan dan studi kasus, serta analisis kebutuhan, pemrosesan data, teknik eksperimen, teknik evaluasi, dan sebagainya.

3.1 Tahapan Penelitian

Pada bagian ini berisi pembahasan mengenai tahapan penelitian yang dilakukan, mulai dari studi literatur sampai tahapan penarikan kesimpulan. Tahapan yang secara umum akan dilakukan dalam penelitian ini adalah sebagai berikut.



Gambar 3. 1 Tahapan Penelitian

Sesuai yang telah digambarkan pada Gambar 3. 1, tahapan penelitian yang dilakukan dimulai dengan tahapan studi pendahuluan dan diakhiri dengan tahapan evaluasi. Tahapan-tahapan ini dibagi menjadi bagian persiapan, pengembangan, dan evaluasi. Bagian persiapan terdiri dari tahapan studi pendahuluan, analisis, dan perancangan sistem. Bagian pengembangan terdiri dari tahapan pengumpulan data, *data preprocessing*, *training model*, dan *testing model*. Terakhir yaitu bagian evaluasi terdiri dari tahapan evaluasi.

Penelitian dimulai dari bagian yang pertama, yaitu bagian persiapan. Seperti yang telah disebutkan sebelumnya, bagian persiapan terdiri dari tahapan studi pendahuluan, analisis, dan perancangan sistem. Pada tahapan studi pendahuluan, dilakukan studi literatur mengenai teori-teori penelitian terkait yang nantinya akan membantu dalam pemahaman mengenai penelitian ini. Tahapan selanjutnya yaitu analisis. Pada tahapan ini, dilakukan analisis hal-hal apa saja yang dibutuhkan dalam mengembangkan model *machine learning* yang dapat melakukan klasifikasi gambar makanan yang memiliki banyak kelas menggunakan algoritma CNN. Berikutnya, tahapan perancangan sistem. Tahapan ini merupakan tahapan di mana dilakukan perancangan alur pengembangan model *machine learning.* Pada tahapan ini juga nantinya akan menghasilkan *output* berupa *flowchart* alur pengembangan model.

Setelah menyelesaikan bagian persiapan, mulai dari tahapan studi literatur hingga perancangan sistem, maka tahapan penelitian selanjutnya adalah bagian pengembangan. Tahapan yang dilakukan pada bagian ini yaitu tahapan pengumpulan data, *data preprocessing*, *training model*, dan *testing model.* Pada tahapan pengumpulan data, dilakukan pengumpulan *dataset* gambar makanan yang nantinya akan digunakan dalam pengembangan model *machine learning* dan pengumpulan data kalori makanan sesuai dengan kelas data makanan. Tahapan selanjutnya, yaitu *data preprocessing*, dimana dilakukan pengolahan *dataset* gambar yang sebelumnya hanya data mentah. Pada tahap ini juga dilakukan *clean up* data dan *augmentasi* data. Setelah itu, dilakukan

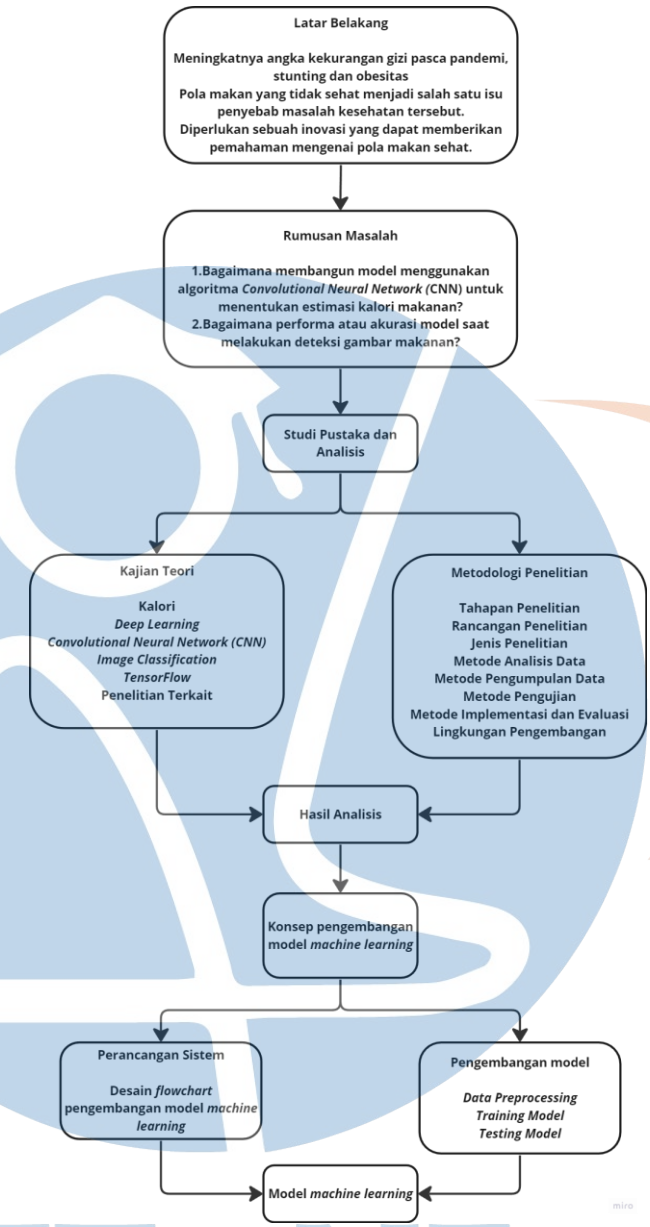
tahapan *training model*. Pada tahapan ini, data yang sebelumnya sudah diproses dan diolah pada tahapan *data preprocessing*, kemudian digunakan pada tahapan ini untuk melakukan *training* pada model, sehingga model dapat mengenali gambar-gambar yang ada pada *dataset* dan mengelompokkan setiap gambar ke dalam kelas-kelas yang sesuai. Terakhir, masuk pada tahapan *testing model*. Pada tahapan ini dilakukan pengujian pada model yang sudah melewati tahap *training* dengan menggunakan data yang baru dan tidak termasuk ke dalam data *training*.

Setelah menyelesaikan semua tahapan baik pada bagian persiapan maupun bagian pengembangan secara berurutan dari awal hingga akhir, langkah selanjutnya yaitu melakukan evaluasi. Tahapan evaluasi merupakan bagian terakhir pada penelitian ini. Pada tahapan ini, dilakukan penarikan kesimpulan dari penelitian yang telah dilakukan pada tahapan pengembangan. Kemudian, akan diberikan rekomendasi saran untuk penelitian selanjutnya sesuai dengan hasil yang sudah diperoleh dari penelitian ini.

3.2 Rancangan Penelitian

Pada bagian rancangan penelitian ini terdapat deskripsi dan hasil kegiatan analisis latar belakang penelitian, pendefinisian kebutuhan dan rumusan masalah, studi pustaka dan analisis, metodologi penelitian, hasil analisis, konsep pengembangan, perancangan sistem, pengembangan model, dan hasil model.

Rancangan penelitian yang telah dijelaskan sebelumnya ditampilkan dalam diagram alir. Diagram alir berikut ini merupakan garis besar dari rancangan penelitian.



Gambar 3. 2 Rancangan Penelitian

Selanjutnya rancangan penelitian dapat didetailkan kembali sesuai sub-sub bab di bawah ini:

1. Studi Pendahuluan

   Tahap ini merupakan tahap awal dalam penelitian ini. Pada tahap ini dilakukan studi pendahuluan melalui studi literatur untuk mengumpulkan informasi dan data awal dalam menentukan permasalahan yang akan dikembangkan pada penelitian ini. *Output* dari tahap ini merupakan studi

literatur mengenai teori-teori penelitian terkait untuk membantu pemahaman dalam penelitian.

2. Analisis

   Tahap selanjutnya yaitu melakukan analisis terhadap kebutuhan sistem. *Output* dari tahap ini yaitu analisis terhadap hal-hal apa saja yang dibutuhkan dalam mengembangkan sistem yang dapat melakukan klasifikasi gambar makanan dengan menerapkan algoritma *Convolutional Neural Network (CNN)*. Analisis dilakukan dengan cara melakukan observasi dan perbandingan terhadap aplikasi yang serupa dengan memperhatikan kelebihan dan kekurangan pada setiap aplikasi.

3. Perancangan Sistem

   Pada tahap ini, dilakukan perancangan proses pengembangan model *machine learning* untuk melakukan klasifikasi gambar berdasarkan kelas-kelas yang ada. *Output* dari tahap ini yaitu desain *flowchart* dari pengembangan model machine learning yang menerapkan algoritma *Convolutional Neural Network* (CNN).

4. Pengumpulan Data

   Pada tahapan pengumpulan data. Data yang dikumpulkan yaitu data gambar. Pada tahapan ini dilakukan pengumpulan data gambar makanan dan data kalori. *Output* dari tahapan ini yaitu data kalori dari masing-masing jenis makanan yang ada pada *dataset* gambar makanan. Selain itu, juga terdapat *output* berupa *dataset* gambar makanan yang nantinya akan digunakan saat melakukan *training* pada model *machine learning*.

5. *Data Preprocessing*

   Pada tahap ini dilakukan proses pengolahan dan persiapan data mentah gambar makanan sebelum digunakan untuk *training* model. *Output* dari tahapan ini adalah berupa data gambar yang telah diolah untuk memastikan bahwa data yang digunakan dalam model CNN memiliki format yang benar dan kualitas yang baik untuk meningkatkan akurasi model.

6. *Training Model*

   Tahap selanjutnya yaitu *training* model dengan menggunakan data yang telah diolah sebelumnya. *Output* dari tahap ini adalah model *machine learning* yang telah melewati tahap pelatihan dengan menggunakan *dataset* training yang telah diolah pada tahap data *preprocessing* sebelumnya.

7. *Testing Model*

   Tahap testing merupakan tahap di mana model hasil tahap *training model* sebelumnya melewati tahap testing. Di tahap ini akan diukur dan dilihat bagaimana akurasi dari model, apakah telah mencapai tingkat akurasi yang diinginkan atau belum. *Output* dari tahap ini adalah hasil uji model terhadap data testing berupa akurasi.

8. Evaluasi

   Tahap terakhir dari penelitian ini yaitu tahap evaluasi. *Output* dari tahap ini adalah penarikan kesimpulan dari hasil penelitian yang telah didapatkan dan menentukan saran untuk penelitian selanjutnya.

3.2.1 Jenis Penelitian

Jenis penelitian yang digunakan dalam penelitian ini adalah penelitian pengembangan (R&D). Penelitian R&D merupakan metode dan langkah-langkah yang digunakan untuk menghasilkan dan menciptakan produk baru atau untuk mengembangkan dan menyempurnakan produk yang sudah ada dengan tujuan menguji efektivitas produk tersebut sehingga dapat dipertanggungjawabkan. Penelitian R&D didasarkan menjadi dua tujuan, yaitu pengembangan prototipe produk dan perumusan saran-saran metodologis untuk pendesainan dan evaluasi prototipe produk tersebut. Dalam penelitian R&D terdapat dua tipe[41]. Adapun dua tipe tersebut adalah sebagai berikut.

a. Tipe pertama difokuskan pada perancangan dan evaluasi produk atau program tertentu. Tujuan utama dari tipe ini adalah untuk memahami proses pengembangan dan mengidentifikasi kondisi yang mendukung implementasi program tersebut.

b. Tipe kedua berfokus dan menitikberatkan pada analisis program pengembangan yang telah dilakukan sebelumnya. Tujuan dari tipe ini adalah untuk mendapatkan gambaran mengenai prosedur perancangan dan evaluasi yang efektif.

Menurut Sugiyono [41], penelitian R&D memiliki empat tingkatan. Pada tingkatan 1, penelitian bertujuan untuk menghasilkan rancangan tanpa dilanjutkan dengan pembuatan maupun pengujian produk. Pada tingkatan 2, peneliti tidak melakukan penelitian tetapi langsung menguji produk yang sudah ada. Pada tingkatan 3, peneliti mengembangkan atau merevisi produk yang sudah ada, membuat produk revisi, dan menguji efektivitasnya. Terakhir, tingkatan 4, penelitian bertujuan untuk menciptakan produk baru dan menguji efektivitas produk tersebut.

Berdasarkan penjelasan sebelumnya, maka disimpulkan bahwa metode penelitian R&D merupakan penelitian yang digunakan untuk tujuan mengembangkan produk tertentu serta menguji efektivitasnya. Pengembangan produk dimulai dengan penelitian yang berfokus pada analisis kebutuhan. Untuk memastikan produk tersebut memiliki manfaat yang baik di masyarakat, dilakukan pengujian efektivitas. Hasil akhir dari penelitian ini adalah model *machine learning* yang mampu mengklasifikasikan gambar makanan ke dalam 20 kelas yang telah ditentukan sebelumnya. Model ini kemudian dapat diimplementasikan ke dalam aplikasi yang dapat memberikan estimasi kalori makanan.

Penelitian ini berfokus akan pada pengembangan model *machine learning* serta implementasi algoritma *Convolutional Neural Network (CNN)* pada model *machine learning* yang dapat melakukan klasifikasi gambar, sehingga dapat mengenali gambar yang di*input* sesuai dengan *dataset* yang ada.

3.2.2 Metode Analisis Data

Dalam penelitian ini, metode analisis data yang digunakan adalah metode kuantitatif. Metode analisis data kuantitatif merupakan metode analisis dengan mengumpulkan data numerik dan melakukan analisis terhadap data tersebut menggunakan metode statistik. Tujuan dari metode ini adalah untuk menghasilkan data objektif dan empiris yang dapat diukur serta diekspresikan dalam bentuk angka.

Metode tersebut digunakan dalam penelitian ini. Metode analisis data kuantitatif dalam penelitian klasifikasi gambar menggunakan CNN melibatkan pengolahan dan evaluasi data numerik yang dihasilkan dari gambar digital. Dalam konteks penelitian ini, data yang digunakan adalah gambar buah-buahan yang dikelompokkan ke dalam 20 kelas yang berbeda. Tujuan utamanya adalah untuk melakukan *training* dan *testing* model CNN agar dapat mengklasifikasikan gambar dengan akurasi yang sesuai.

Proses pengolahan citra pada CNN melibatkan proses konvolusi. Konvolusi pada dasarnya berupa *dot product* antara filter dengan sebuah bidang reseptif kecil pada *input* (citra) yang berukuran sama dengan filter. Ukuran citra *input-an* dapat dinyatakan sebagai berikut.

$$w_1 \times h_1 \times d_1 \quad (3.\ 1)$$

Pada persamaan (3. 1), w_1 dan h_1 adalah lebar dan tinggi citra sedangkan d_1 adalah jumlah kanal *Red Greeen Blue* (RGB) atau disebut *deepth* atau kedalaman citra *inputan*[18]. Dari *inputan* ini akan diperoleh *output* yang dapat dihitung seperti berikut.

$$w_2 \times h_2 \times d_2 \quad (3.\ 2)$$

Sesuai persamaan (3. 2), di mana nilai $w_2, h_2,$ dan d_2 didapatkan dari perhitungan persamaan (3. 3), (3. 4), dan (3. 5) berikut ini.

$$w_2 = \frac{(w_1 - F + 2P)}{S} + 1 \quad (3.\ 3)$$

$$h_2 = \frac{(h_1 - F + 2P)}{S} + 1 \quad (3.\ 4)$$

$$w_2 = K \quad (3.\ 5)$$

Di mana, *F* adalah ukuran bidang reseptif atau *spatial extent* (tingkat spasial), *S* adalah lebar langkah atau *stride*, *P* adalah *zero padding*, dan *K* adalah jumlah filter.

3.2.3 Metode Pengumpulan Data

Data yang digunakan dalam penelitian ini merupakan data sekunder, yaitu *dataset* gambar buah-buahan dan sayuran yang diperoleh dari beberapa *dataset* yang didapat melalui situs data *open source Kaggle*[42], [43], [44], [45], [46]. Pada *dataset* tersebut terdapat 90380 gambar, dari 131 kelas buah-buahan dan sayuran. Namun, pada penelitian ini hanya diambil 20 kelas saja. Kelas data yang diambil terdiri dari anggur, apel, belimbing, buah naga, jagung, jeruk, kentang, kol, labu, mangga, nanas, pepaya, pir, pisang, selada, semangka, *strawberry*, timun, tomat, dan wortel.

Selain itu juga dikumpulkan data kalori sesuai dengan 20 jenis buah-buah dan sayuran yang digunakan. Data kalori ini diambil situs *FatSecret* Indonesia[47]. Data kalori diasumsikan berdasarkan data yang ada pada *website FatSecret* Indonesia yair per 100 gram untuk masing-masing jenis.

3.2.4 Metode Pengujian

Dalam penelitian ini, metode pengujian yang digunakan adalah metode *statistical testing*. *Statistical testing* merupakan pendekatan terhadap penarikan kesimpulan berdasarkan perhitungan statistik yang menentukan apakah data cukup mendukung hipotesis tertentu. Pada penelitian ini, pengujian *statistical testing* digunakan dalam mengevaluasi kinerja model.

Metode *statistical testing* dilakukan untuk mengetahui berapa persentase keberhasilan model dalam melakukan klasifikasi. Persentase keberhasilan didapatkan dari hasil perhitungan metrik *accuracy*. Metrik a*ccuracy* atau tingkat pengenalan menyatakan persentase dari jumlah *tuple* dalam data uji yang diklasifikasikan dengan benar oleh model[18]. Persamaan (3. *6*) berikut merupakan formula untuk menghitung metrik *accuracy* [9].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.\ 6)$$

TP dan *TN* merupakan jumlah *tuple* positif dan negatif, masing-masing diklasifikasikan dengan benar oleh model. Kemudian, untuk *FP* dan *FN* merupakan jumlah *tuple* negatif yang salah diklasifikasikan oleh model.

Metode pengujian ini merupakan komponen penting dalam melakukan validasi dan evaluasi model *machine learning*. Dengan menggunakan teknik pengujian *statistical testing* ini, peneliti dapat memastikan bahwa model yang dikembangkan tidak hanya efisien tetapi juga andal dan dapat diimplementasikan dengan baik.

3.2.5 Metode Implementasi dan Evaluasi

Pada bagian ini dilakukan implementasi pengembangan model *machine learning* yang dapat melakukan klasifikasi gambar makan ke dalam 20 kelas yang sudah ditentukan sebelumnya. Data yang sudah dikumpulkan sebelumnya, kemudian dilakukan *splitting* data menjadi data *training, validation,* dan, *testing.* Tahapan dalam proses implementasi ini meliputi data *preprocessing*, *training*, dan *testing*.

Pada tahap evaluasi, model *machine learning* yang berhasil dikembangkan kemudian dilakukan pengujian dengan mengukur metrik *accuracy*. Jika *accuracy* yang dihasilkan dari pengujian tersebut masih belum sesuai, maka akan dilakukan optimasi pada model, yaitu dengan konfigurasi ulang *hyperparameter* yang digunakan.

3.2.6 Lingkungan Pengembangan

Dalam membangun dan mengembangkan penelitian ini, dibutuhkan perangkat pendukung untuk dalam proses pengembangan penelitian ini. Adapun perangkat yang digunakan terbagi menjadi dua jenis, yaitu:

- *Hardware*

  Perangkat *hardware* yang digunakan dalam membangun dan mengembangkan penelitian ini, yaitu laptop *device* dengan rincian sebagai berikut:

  1) Asus vivobook 14x m1403qa,
  2) *Operating system* Windows 11 Home 64-bit,
  3) *Processor* AMD Ryzen™ 5 5600H,
  4) *Graphics* AMD Radeon(TM) Graphics,
  5) RAM 8 GB.

- *Software*

  Perangkat *software* yang digunakan dalam membangun dan mengembangkan penelitian ini adalah sebagai berikut:

  1) Google Colab,
  2) Bahasa pemrograman Python,
  3) *Library* TensorFlow,
  4) *Framework Streamlit.*

BAB IV
IMPLEMENTASI DAN EVALUASI

Pada bab ini akan membahas mengenai implementasi dari sistem yang telah dirancang sebelumnya, hingga membahas mengenai pengujian sistem serta evaluasi sistem.

4.1 Analisis dan Perancangan

Pada bagian ini membahas mengenai analisis dan perancangan model *machine learning* yang mengimplementasikan algoritma CNN untuk klasifikasi gambar makanan. Dalam mengembangkan model *machine learning* ini, analisis kebutuhan sistem dan perancangan model *machine learning* harus dilakukan terlebih dahulu. Perancangan sebuah sistem membutuhkan analisis, hal ini dilakukan untuk mengidentifikasi data dan informasi yang dibutuhkan untuk penelitian ini. Perancangan dilakukan untuk menentukan gambaran umum sistem yang akan dikembangkan.

4.1.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan langkah untuk menganalisis hal-hal apa saja yang dibutuhkan dalam pengembangan model *machine learning* yang dapat melakukan klasifikasi data gambar menggunakan algoritma CNN. Analisis ini diperlukan untuk memastikan bahwa semua komponen yang diperlukan tersedia dan dioptimalkan untuk mencapai hasil yang sesuai. Analisis ini mencakup identifikasi kebutuhan *hardware, software,* dan data serta manajemen data.

1) Kebutuhan *Hardware*

   Dalam mengembangkan model *machine learning* dibutuhkan perangkat keras (*hardware*) yang memadai. *Hardware* ini akan sangat membantu proses pengembangan berjalan dengan efektif dan efisien. Adapun *hardware* yang digunakan selama proses pengembangan ini adalah laptop merek Asus Vivobook dengan spesifikasi yaitu meliputi CPU dengan *processor* AMD Ryzen 5, RAM 8 GB, serta penyimpanan SSD dengan

kapasitas 512 GB untuk kecepatan akses data yang lebih tinggi. Selain itu, juga dibutuhkan perangkat tambahan untuk mendukung kinerja *hardware*. Perangkat tambahan tersebut meliputi *cooling system* dan *power supply*. *Cooling system* berfungsi sebagai penjaga suhu selama proses pelatihan model yang intensif dan *power supply* berfungsi untuk mendukung semua komponen *hardware* agar dapat stabil.

2) Kebutuhan *Software*

Selama proses pengembangan model *machine learning* dibutuhkan *software* yang mendukung. *Software* yang dibutuhkan meliputi bahasa pemrograman dan *library*, serta *tools* yang digunakan. *Tools* yang digunakan untuk pengembangan dan eksperimen interaktif yang memungkinkan penulisan dan eksekusi kode yang dapat dijalankan secara individual digunakan Google Colab. Bahasa pemrograman yang digunakan yaitu Python versi 3.9 64 bit, digunakannya bahasa ini karena Python merupakan bahasa yang paling umum digunakan dalam pengembangan *macine learning*.

TensorFlow digunakan sebagai *framework* utama untuk mengimplementasikan CNN. TensorFlow dipilih karena menawarkan banyak alat dan dukungan komunitas yang luas. Numpy dan Pandas juga digunakan sebagai *library* untuk melakukan manipulasi data dan komputasi numerik agar menjadi lebih efisien. Kemudian, untuk melakukan visualisasi data dan memahami data dan hasil model, digunakan Matplotlib.

3) Kebutuhan Data

Data merupakan komponen penting dalam pengembangan model *machine learning*, khususnya CNN. Kualitas, kuantitas, dan manajemen data yang tepat akan menentukan seberapa baik model dapat belajar dan generalisasi pada data baru. Oleh karena itu, dibutuhkan *dataset* yang akan digunakan nantinya.

Dataset merupakan kumpulan data yang digunakan selama proses *training, testing,* dan *validation* model *machine learning*. *Dataset* gambar yang digunakan harus relevan dengan tujuan pengembangan. Dengan tujuan untuk mengembangkan model yang dapat melakukan klasifikasi gambar makanan, maka pada pengembangan model pada penelitian ini, *dataset* yang digunakan merupakan *dataset* gambar makanan yang dibatasi menjadi gambar buah-buahan dan sayuran. Setiap gambar yang ada di dalam *dataset* ini memiliki label masing-masing sesuai dengan kelasnya. Label ini digunakan oleh algoritma untuk belajar mengasosiasikan *input* dengan *output* yang benar. Untuk pengembangan model pada penelitian ini, *dataset* akan memiliki 20 kelas (label) dengan total jumlah data sebanyak 3.362 gambar dengan model warna RGB. Jumlah data ini membantu model dalam belajar fitur yang lebih umum dan mengurangi *overfitting*.

Data gambar yang digunakan pada penelitian ini adalah data sekunder yang diambil dari beberapa sumber data *open source* di internet yaitu *kaggle*[43], [44], [45], [46], *google images, unsplash, istock, pexels,* dan *freepik*. Data yang dikumpulkan dari situs-situs tersebut merupakan gambar buah-buahan dan sayuran yang nantinya akan digunakan untuk proses *training* model *machine learning*. Setiap data gambar buah-buahan dan sayuran ini diletakkan di dalam satu folder yang di dalam terdapat folder-folder sesuai nama buah-buahan dan sayuran yang akan digunakan dalam proses pengembangan model ini. Di dalam folder buah/sayuran tersebut berisi gambar-gambar buah/sayuran dengan format gambar *jpg, jpeg*, dan *png*. Tiga format tersebut digunakan karena *tensorflow* hanya dapat mengenali dan mengolah tiga format gambar tersebut.

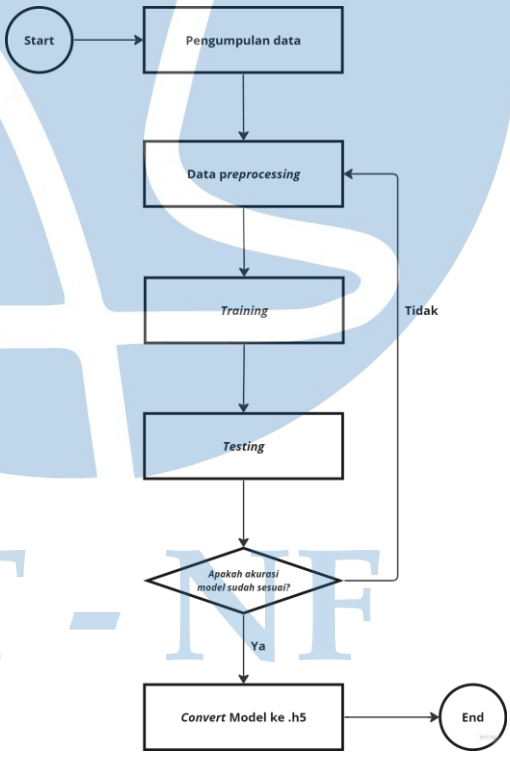
4) Penyimpanan Data dan Manajemen Data

Manajemen data yang efektif memastikan bahwa data yang digunakan untuk pelatihan model dapat diakses, diolah, dan disimpan dengan efisien. Oleh karena itu, sistem penyimpanan yang memadai sangat diperlukan

untuk menangani *dataset* yang besar. Pada penelitian ini, penyimpanan data yang digunakan berupa *cloud storage* yaitu Google Drive dengan ukuran penyimpanan sebesar 100 GB.

4.1.2 Perancangan Sistem

Secara garis besar, pada perancangan ini ada lima tahap utama dalam pengembangan sistem model *machine learning* yang dapat melakukan klasifikasi gambar makanan. Tahapan-tahapan tersebut yaitu terdiri dari tahapan pengumpulan data, *data preprocessing*, *training*, *testing*, dan *convert* model mejadi format "*.h5*" pada tahap akhir. Pada Gambar 4. 1 berikut merupakan *flowchart* perancangan sistem dalam pengembangan model *machine learning*.



Gambar 4. 1 *Flowchart* rancangan model

4.2 Implementasi dan Evaluasi Sistem

Pada bagian ini dijelaskan mengenai implementasi hingga evaluasi pada pengembangan model *machine learning* untuk kebutuhan sistem klasifikasi gambar makanan yang nantinya dapat diimplementasikan ke aplikasi yang dapat menentukan estimasi kalori makanan. Sesuai dengan yang telah dijelaskan pada 4.1.2, tahapan proses dalam penelitian ini meliputi: pengumpulan data, data *preprocessing*, *training*, *testing*, dan konversi model menjadi format ".h5", dan terakhir tahap evaluasi model yang dihasilkan.

4.2.1 Pengumpulan data

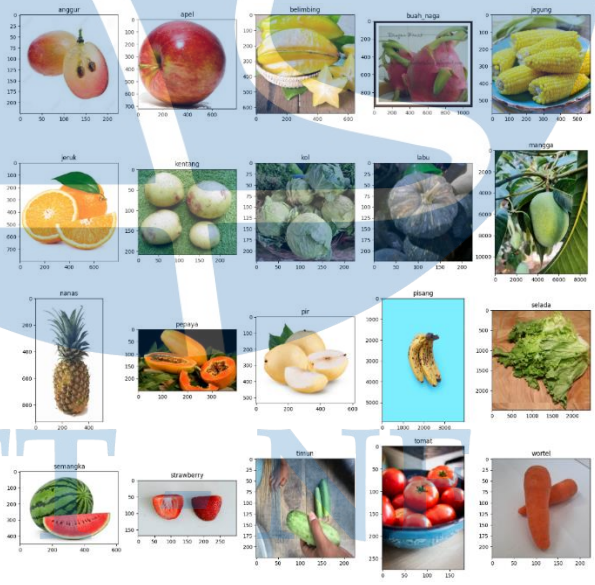
Pada tahap ini dilakukan proses pengumpulan *dataset* gambar buah-buahan dan sayuran. Data gambar ini diperoleh dari beberapa situs *open source* yaitu *kaggle*[43], [44], [45], [46], *google images*, *unsplash*, *istock*, *pexels*, dan *freepik*. Seperti yang sudah dijelaskan pada bagian 4.1.1, bahwa *dataset* yang digunakan terdiri dari gambar 20 jenis buah-buahan dan sayuran dengan total 3.362 gambar. Jenis-jenis tersebut yaitu terdiri dari anggur, apel, belimbing, buah naga, jagung, jeruk, kentang, kol, labu, mangga, nanas, pepaya, pir, pisang, selada, semangka, *strawberry*, timun, tomat, dan wortel. Jumlah data untuk setiap jenis ditunjukkan pada Tabel 4. 1.

Tabel 4. 1 Jumlah data per-kelas

Kelas	Jumlah
Anggur	179
Apel	173
Belimbing	159
Buah Naga	104
Jagung	181
Jeruk	169
Kentang	191
Kol	173
Labu	170
Mangga	169

Kelas	Jumlah
Nanas	178
Pepaya	199
Pir	104
Pisang	173
Selada	171
Semangka	160
Strawberry	151
Timun	180
Tomat	210
Wortel	168
Jumlah	**3.362**

Berikut sampel citra dari masing-masing gambar buah-buahan dan sayuran dari masing-masing kelas.



Gambar 4. 2 Sampel citra buah-buahan dan sayuran

Selain itu, juga dikumpulkan data kalori sesuai dengan jenis data makanan yang digunakan. Data kalori ini diambil dari *website FatSecret* Indonesia yang dapat diakses pada halaman https://www.fatsecret.co.id/kalori-gizi/. Berikut tampilan informasi kalori yang ditampilkan pada *website FatSecret* Indonesia.

Gambar 4. 3 Tampilan halaman informasi kalori *FatSecret* Indonesia

Dari pengumpulan data yang dilakukan, maka dihasilkan data kalori berdasarkan 20 jenis data buah dan sayuran seperti yang ditampilkan pada tabel berikut.

Tabel 4. 2 Data Kalori per jenis buah dan sayur

Nama buah	Porsi (Berat)	Estimasi Kalori
Anggur	100 gram	69 kkal
Apel	100 gram	52 kkal
Belimbing	100 gram	31 kkal
Buah Naga	100 gram	51 kkal
Jagung	100 gram	86 kkal
Jeruk	100 gram	62 kkal
Kentang	100 gram	77 kkal
Kol	100 gram	25 kkal
Labu	100 gram	26 kkal
Mangga	100 gram	65 kkal
Nanas	100 gram	48 kkal
Pepaya	100 gram	39 kkal
Pir	100 gram	58 kkal

Nama buah	Porsi (Berat)	Estimasi Kalori
Pisang	100 gram	89 kkal
Selada	100 gram	14 kkal
Semangka	100 gram	30 kkal
Strawberry	100 gram	32 kkal
Timun	100 gram	15 kkal
Tomat	100 gram	18 kkal
Wortel	100 gram	41 kkal

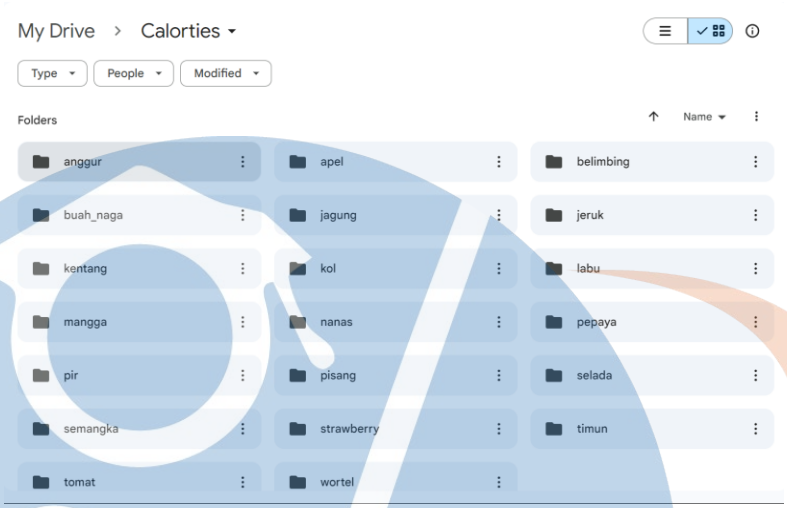
4.2.2 Data *Preprocessing*

Data *preprocessing* (pra-pemrosesan data) merupakan serangkaian tahapan untuk mengolah, mengubah, dan mempersiapkan data mentah sebelum digunakan dalam proses analisis[48]. Tujuannya adalah untuk mempersiapkan data mentah yang akan digunakan sebagai *input* untuk model CNN. Hal ini juga berfungsi untuk meningkatkan efisiensi dalam proses *training* model. Tahapan data *preprocessing* ini penting karena tidak semua metode berbasis *neural network* dapat secara langsung memproses data mentah untuk klasifikasi hasil akhir dan prediksi *error*. Oleh karena itu, teknik *preprocessing* data memainkan peran penting dalam diagnosis kesalahan berbasis CNN yang efisien.

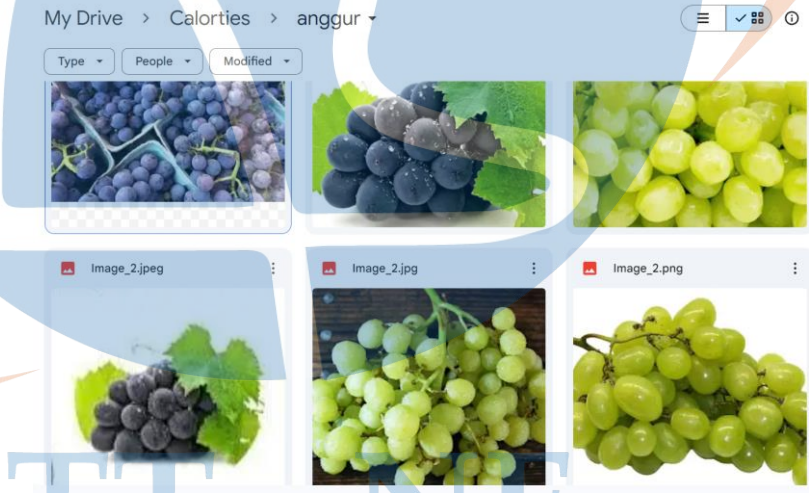
1. **Data *Splitting***

   Sebelum data dapat digunakan pada proses *training*, pada *dataset* akan dilakukan proses *splitting* (pembagian) data menjadi beberapa bagian. Proses *splitting* ini penting dilakukan untuk memastikan model yang dikembangkan dapat melakukan generalisasi yang baik pada data yang belum pernah diberikan sebelumnya. Data yang sebelumnya telah dikumpulkan untuk proses pengembangan disimpan di dalam *google drive*. Data ini masih diletakkan dalam 1 folder, yaitu folder yang diberi nama *Calorties*. Di dalam folder *Calorties* ini terdapat 20 folder untuk masing-masing jenis buah-buahan dan sayuran, di mana setiap folder buah/sayuran

tersebut berisi data gambar dalam format *jpg*, *jpeg*, dan *png*. Seperti yang terlihat pada Gambar 4. 4 dan Gambar 4. 5 di bawah ini.



Gambar 4. 4 Folder calorties



Gambar 4. 5 Folder buah anggur

Selanjutnya, dari data tersebut dilakukan *splitting* data dengan membagi *dataset* menjadi tiga bagian. Data dibagi menjadi data *training*, data *validation*, dan data *testing*. Proses *splitting* dilakukan dengan menggunakan *library sklearn*. *Sklearn* merupakan *library* yang merujuk pada *library scikit-learn* dan merupakan *library* yang populer dalam *machine learning*. *Library* ini menyediakan berbagai modul dan fungsi

yang berguna untuk pemodelan data, termasuk klasifikasi, *clustering*, regresi, dan pengolahan data. Pada proses *splitting* ini digunakan fungsi *train_test_split* yang diimpor dari modul *model_selection*. Fungsi ini digunakan untuk melakukan pembagian *dataset* menjadi *subset training*, *testing*, dan *validation*.

```
from sklearn.model_selection import train_test_split


X= df['path']

y= df['tag']


X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.20, random_state=300)


X_test, X_val, y_test, y_val = train_test_split(

    X_test, y_test, test_size=0.5, random_state=100)
```

Sesuai yang terlihat pada kode di atas, setelah melakukan *import* fungsi *train_test_split*, maka selanjutnya alamat lokasi setiap gambar disimpan di dalam variabel x dan label/kelas gambar akan disimpan ke variabel y. Kemudian, data dibagi menjadi *subset* data *train* dan data *test* terlebih dahulu. Terlihat pada kode bahwa *test_size=20*, parameter ini menentukan proporsi dari *dataset* yang digunakan sebagai data *testing*. Hal tersebut berarti 20% dari data yang ada di variabel x dan y akan digunakan sebagai data *testing*, dan 80% sisanya digunakan sebagai data *training*. Ukuran *training* 80% ini berfungsi untuk mendapatkan hasil pelatihan yang lebih baik, karena menggunakan data yang lebih banyak.

Kemudian, *subset* data *testing* yang diperoleh dari hasil pembagian data sebelumnya dilakukan *splitting* lagi untuk mendapatkan *subset* data *validation*. Sama seperti *splitting* sebelumnya yang menggunakan

parameter untuk membagi *dataset*, pada proses ini parameter yang digunakan adalah *test_size=0.5*. Artinya, 50% dari data *testing* sebelumnya digunakan sebagai data *testing* yang baru dan 50% lagi sebagai data *validation*. Hasil pembagian *dataset* dapat dilihat pada tabel Tabel 4. *3*.

Tabel 4. 3 Hasil pembagian dataset

Subset	**Ukuran (%)**
training	80%
validation	10%
testing	10%

```
df_tr = pd.DataFrame({'path':X_train
,'tag':y_train
,'set':'train'})

df_te = pd.DataFrame({'path':X_test
,'tag':y_test
,'set':'test'})

df_val = pd.DataFrame({'path':X_val
,'tag':y_val
,'set':'validation'})

print('train size', len(df_tr))
print('val size', len(df_te))
print('test size', len(df_val))
```

```
train size 2689
val size 336
test size 337
```

Gambar 4. 6 Hasil proses *splitting*

Sesuai dengan yang ditampilkan pada Gambar 4. 6, setelah masing-masing data subset disatukan di dalam *dataframe* masing-masing, dari jumlah total data 3.362 gambar, dihasilkan data *training* sebanyak 2.689 gambar, data *validation* sebanyak 336 gambar, dan data *testing* sebanyak 337 data.

Jumlah data untuk masing-masing kelas buah/sayuran dapat dilihat pada Gambar 4. 7. Hasil dari *splitting* ini kemudian disimpan ke dalam *google drive* agar dapat diakses dengan mudah saat melakukan proses *training* melalui *google colab*.

set	tag		train			validation		
test	anggur	19		anggur	135		anggur	25
	apel	8		apel	142		apel	23
	belimbing	16		belimbing	124		belimbing	19
	buah_naga	8		buah_naga	87		buah_naga	9
	jagung	18		jagung	150		jagung	13
	jeruk	20		jeruk	128		jeruk	21
	kentang	17		kentang	156		kentang	18
	kol	18		kol	137		kol	18
	labu	20		labu	130		labu	20
	mangga	15		mangga	144		mangga	10
	nanas	23		nanas	140		nanas	15
	pepaya	22		pepaya	154		pepaya	23
	pir	12		pir	82		pir	10
	pisang	15		pisang	142		pisang	16
	selada	16		selada	135		selada	20
	semangka	17		semangka	128		semangka	15
	strawberry	14		strawberry	123		strawberry	14
	timun	22		timun	141		timun	17
	tomat	18		tomat	172		tomat	20
	wortel	18		wortel	139		wortel	11

Gambar 4. 7 Hasil pembagian dataset pada masing-masing jenis buah/sayuran

2. Augmentasi Data

Teknik augmentasi data digunakan untuk meningkatkan *dataset training* dengan tujuan menaikkan performa generalisasi dan akurasi klasifikasi CNN[48]. Pada penelitian ini, sebelum dilakukan proses *training* model, hal yang dilakukan terlebih dahulu adalah proses augmentasi data. Augmentasi data dilakukan pada *dataset* dengan tujuan meningkatkan variasi *dataset*. Dengan memperbesar dan memperkaya *dataset* melalui teknik augmentasi, model akan dapat belajar lebih baik dengan berbagai variasi data. Hal ini akan meningkatkan performa dan keandalan model dalam mengenal data baru.

Teknik augmentasi pada penelitian ini menggunakan *class ImageDataGenerator* yang di-*import* dari *library Keras* yang disediakan oleh *framework TensorFlow*. Seperti yang terlihat pada kode yang dilampirkan di bawah ini, pada *subset training* dilakukan augmentasi dengan melakukan beberapa transformasi terhadap *dataset*. Pada *subset training* tersebut dilakukan normalisasi gambar dengan mengatur parameter *rescale=1./255*. Tujuan dari *rescaling* ini adalah melakukan

normalisasi terhadap gambar dengan membagi setiap piksel gambar dengan 255. Sehingga nilai-nilai piksel gambar yang sebelumnya memiliki nilai rentang 0 hingga 255 (gambar RGB), menjadi rentang 0 hingga 1. Hal yang sama juga dilakukan pada *subset validation* dan *subset testing*.

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.3,
                                   zoom_range=0.3,
                                   horizontal_flip=True,
                                   vertical_flip=True,
                                  height_shift_range=0.2,
                                   width_shift_range=0.2,
                                   rotation_range=40,
                                   brightness_range=[0.2,
                                   1.0],
                                channel_shift_range=150.0)

validation_datagen = ImageDataGenerator(rescale=1./255)

test_datagen = ImageDataGenerator(rescale=1./255)
```

Transformasi gambar pada *subset training* yang dilakukan juga meliputi *shear range, zoom range,* horizontal dan vertikal *flip, height shift range, width shift range, rotation range, brightness range,* dan *channel shift range.* Lihat parameter yang ada pada *train_datagen* yang ditampilkan pada kode di atas, berikut penjelasan untuk setiap parameter tersebut.

- *Shear_range=0.3,* artinya diterapkan proses penggeseran sebanyak 30% dari dimensi gambar, sementara bagian dimensi lainnya tetap atau tidak dilakukan pergeseran.
- *Zoom_range=0.3,* artinya diterapkan pembesaran atau pengecilan gambar secara acak hingga 30%. Hal tersebut berarti bahwa *zoom* akan berkisar antara 1-0.3 (70%) dan 1+0.3 (130%) dari ukuran asli. Berarti gambar bisa saja diperkecil menjadi 70% (paling kecil) dari gambar asli

atau di atasnya dan gambar juga bisa diperbesar menjadi 130% (paling besar) dari gambar asli atau lebih di bawahnya.

- *Horizontal_flip=True*, artinya gambar dibalik secara horizontal 180 derajat di sekitar sumbu vertikal.
- *Vertical_flip=True*, artinya gambar dibalik secara vertikal 180 derajat di sekitar sumbu horizontal.
- *Height_shift_range=0.2*, artinya gambar digeser secara vertikal hingga 20% dari total tinggi gambar secara acak. Misalnya, jika gambar memiliki tinggi 100 piksel, maka gambar bisa digeser hingga 20 piksel ke atas atau ke bawah. Jika gambar asli memiliki objek di tengah, setelah dilakukan *height shift*, objek bisa saja menjadi lebih dekat ke tepi atas gambar atau bawah gambar tergantung pada arah dan besaran pergeseran yang dipilih secara acak.
- *Vertical_shift_range=0.2*, sama seperti yang dijelaskan pada *height shift*, bedanya pada *vertical shift* gambar akan digeser hingga 20% secara horizontal dari total lebar gambar secara acak.
- *Rotation_range=40*, artinya gambar diputar dalam rentang 40 derajat, baik ke kanan maupun ke kiri (-40 hingga +40).
- *Brightness_range=[0.2, 1.0]*, artinya kecerahan gambar diubah secara acak dalam rentang 0.2 hingga 1.0 kali kecerahan asli. Nilai 0.2 berarti gambar bisa menjadi sangat gelap (20% dari kecerahan asli) sedangkan nilai 1.0 berarti gambar akan memiliki kecerahan yang sama dengan aslinya (100% dari kecerahan asli).
- *Channel_shift_range=150.0*, artinya nilai nilai piksel di setiap *channel* warna (RGB) dari gambar akan digeser secara acak dengan nilai antara -150 hingga +150. Pergeseran ini diterapkan secara terpisah pada masing-masing kanal warna, sehingga bisa menyebabkan perubahan warna keseluruhan gambar. Misal, gambar yang telah dilakukan *channel shift* bisa tampak memiliki dominasi warna yang berbeda dari gambar asli. Bisa saja gambar menjadi kemerahan, kehijauan, atau kebiruan tergantung pada pergeseran yang diterapkan.

Dari proses yang sudah dijelaskan sebelumnya, maka data *subset training* akan menjadi sangat bervariasi dari data aslinya. Seperti yang sudah dijelaskan pada paragraf dua, proses augmentasi dilakukan dengan menggunakan *ImageDataGenerator*. Menggunakan *ImageDataGenerator* berarti data augmentasi dilakukan dengan teknik yang disebut *in-place data augmentation* atau *on-the-fly data augmentation.* Proses dari *on-the-fly data augmentation* ini dilakukan seperti berikut.

1) Sekumpulan gambar *input* (sebuah *batch*) dimasukkan ke *ImageDataGenerator*.
2) *ImageDataGenerator* akan melakukan augmentasi dengan transformasi-transformasi yang sudah dijelaskan sebelumnya kepada setiap gambar yang ada dalam *batch* tersebut.
3) Hasil dari augmentasi yang dilakukan pada *batch* tersebut kemudian akan diberikan dan diproses oleh model pada tahap *training.*

   Jadi, jika ditentukan *batch_size=32*, setiap *train_datagen* dipanggil, maka *ImageDataGenerator* akan menghasilkan 32 gambar yang telah dilakukan augmentasi dan model akan menerima dan menggunakan 32 gambar pada *batch* tersebut pada proses satu iterasi *training*.

4.2.3 *Training* Model

Pada tahap ini dilakukan proses *training* model dengan menggunakan data yang telah diproses sebelumnya. Dalam penelitian itu, untuk membangun arsitektur CNN digunakan API *Sequential* dari *library Keras* yang disediakan oleh *framework TensorFlow*. Model dirancang untuk bisa melakukan klasifikasi gambar ke dalam 20 kelas yang sudah dijelaskan sebelumnya.

1. Membangun Arsitektur Model

```
def create_model():
model = tf.keras.models.Sequential([
Conv2D(32, (3, 3), activation='relu',
input_shape=(img_width, img_height, 3)),
BatchNormalization(),
```

```
MaxPooling2D(pool_size=(2, 2)),
Dropout(0.25),

Conv2D(64, (3, 3), activation='relu'),
BatchNormalization(),
MaxPooling2D(pool_size=(2, 2)),
Dropout(0.25),

Conv2D(128, (3, 3), activation='relu'),
BatchNormalization(),0p=
MaxPooling2D(pool_size=(2, 2)),
Dropout(0.25),

Conv2D(256, (3, 3), activation='relu'),
BatchNormalization(),
MaxPooling2D(pool_size=(2, 2)),
Dropout(0.25),

Flatten(),
Dense(512, activation='relu'),
BatchNormalization(),
Dropout(0.5),
Dense(num_classes, activation='softmax')
])
model.compile(
loss = tf.keras.losses.CategoricalCrossentropy(),
optimizer = tf.keras.optimizers.Adam(),
metrics = ["accuracy"])

return model
```

Pada kode di atas merupakan kode yang digunakan untuk membangun arsitektur model. Penjelasan mengenai arsitektur yang dibangun pada kode tersebut adalah sebagai berikut.

1) Lapisan pertama yaitu *input layer* dengan *layer convolution* yang memiliki 32 filter dengan ukuran *kernel* 3×3 dan menggunakan *ReLU activation function*. Pada *layer* ini juga ditentukan ukuran *input* gambar, yaitu 150, 150, 3. Artinya gambar memiliki lebar 150, tinggi 150, dan angka 3 berarti *input* gambar memiliki *channel* warna RGB.
2) Lapisan pertama diikuti oleh *layer BatchNormalization*. *Layer* ini diterapkan setelah setiap *layer convolution* untuk menormalkan *output* dan menstabilkan pelatihan serta mempercepat konvergensi.

3) Lapisan selanjutnya yaitu *pooling layer* yang menggunakan *MaxPooling* dengan nilai *pool size*-nya adalah 2×2 dan juga diterapkan setelah setiap *layer BatchNormalization*. Lapisan ini berfungsi untuk mengurangi dimensi spasial gambar.
4) Kemudian, diikuti lapisan *dropout* dengan tingkat *droput* yaitu 25%. Hal ini berarti 25% dari neuron akan dinonaktifkan secara acak selama pelatihan.
5) Selanjutnya, diterapkan *layer convolution* kedua yang memiliki 64 filter. Sama seperti *layer convolution* yang pertama, *layer convolution* ini memiliki ukuran *kernel* 3×3 dan menggunakan *ReLU activation function*. Kemudian, *layer* ini diikuti *layer normalization, pooling,* dan *dropout* seperti yang sudah dijelaskan pada nomor 2), 3), 4) sebelumnya. Struktur ini diulangi dengan jumlah filter yang meningkat menjadi 128 dan 256 untuk *layer convolution* berikutnya. Jadi, total *layer convolution* dari yang diterapkan adalah empat.
6) Setelah diterapkan empat *layer convolution*, diterapkan lapisan *flatten* yang berfungsi untuk mengubah *output* dari *layer convolution* terakhir menjadi vektor satu dimensi untuk dimasukkan ke dalam *fully connected layer (Dense)*. Hal ini penting dilakukan karena lapisan *dense* hanya menerima *input* dalam bentuk vektor satu dimensi.
7) Selanjutnya, diterapkan *dense layer* yang memiliki 512 unit dengan *ReLU activation function.* Kemudian, diikuti lapisan *normalization* dan *dropout* 50%.
8) Terakhir, *output layer* dengan *dense layer* yang memiliki 20 unit. 20 unit ini merupakan banyaknya kelas yang ada pada *dataset*. *Layer* ini menggunakan *sofmax activation function* untuk menyelesaikan klasifikasi multi-kelas dan menghasilkan distribusi probabilitas untuk setiap kelas.

Kemudian, arsitektur yang sudah dibangun tersebut di-*compile* dengan menerapkan *Categorical Crossentropy* sebagai *loss function*. *Loss function* tersebut dipilih karena model akan menyelesaikan permasalahan klasifikasi

multi-kelas di mana terdapat 20 kelas yang sudah dijelaskan sebelumnya. Selain itu, juga diterapkan *Adam optimizer* dan metrik *accuracy*. Metrik *accuracy* digunakan untuk mengukur persentase prediksi yang benar jika dibandingkan dengan label yang sebenarnya. Hasil *summary* arsitektur model yang telah dibuat dapat dilihat pada Gambar 4. 8 berikut.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 batch_normalization (Batch  (None, 148, 148, 32)      128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0
 D)

 dropout (Dropout)           (None, 74, 74, 32)        0

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 batch_normalization_1 (Bat  (None, 72, 72, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 36, 36, 64)        0
 g2D)

 dropout_1 (Dropout)         (None, 36, 36, 64)        0

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 batch_normalization_2 (Bat  (None, 34, 34, 128)       512
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 17, 17, 128)       0
 g2D)

 dropout_2 (Dropout)         (None, 17, 17, 128)       0

 conv2d_3 (Conv2D)           (None, 15, 15, 256)       295168

 batch_normalization_3 (Bat  (None, 15, 15, 256)       1024
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 7, 7, 256)         0
 g2D)

 dropout_3 (Dropout)         (None, 7, 7, 256)         0

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 512)               6423040

 batch_normalization_4 (Bat  (None, 512)               2048
 chNormalization)

 dropout_4 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 20)                10260

=================================================================
Total params: 6825684 (26.04 MB)
Trainable params: 6823700 (26.03 MB)
Non-trainable params: 1984 (7.75 KB)
_________________________________________________________________
```

Gambar 4. 8 Summary model

Dari *summary* yang ditampilkan di atas, setiap layer memiliki *output shape* dan parameter. Terlihat pada lapisan pertama yaitu *conv2d*, memiliki *output shape* yaitu (None, 148, 148, 32) dengan jumlah parameter 896. *Output shape* ini dihasilkan dari proses konvolusi yang dilakukan dengan menggunakan *kernel* 3×3 dan *padding default* (karena tidak dilakukan pengaturan *padding*). Sehingga, ukuran *output* berkurang karena konvolusi tidak dilakukan pada bagian tepi gambar. Jadi, setiap pengaplikasian *kernel* akan mengurangi dimensi sebesar 1 pada setip sisi (satu kolom dan satu baris). Oleh karena *input* yang diberikan pada lapisan *conv2d* adalah (150, 150, 3) dengan 32 filter, maka *output* yang dihasilkan adalah (148, 148, 32). *Output* ini akan menjadi *input* dari lapisan selanjutnya. Hal yang sama juga dilakukan pada 3 *convolution layer* lainnya, yaitu *conv2d_1* dengan *output* (None, 72, 72, 64), *conv2d_2* dengan *output* (34, 34, 128), dan *conv2d_3* dengan *output* (15, 15, 256). Setelah *convolution layer*, diterapkan *normalization layer* dengan *output* yang dihasilkan masih sama dengan *input* yang diberikan dari *layer* sebelumnya. Hal ini dikarenakan *layer normalization* hanya melakukan normalisasi pada fitur tanpa mengubah ukuran.

Pooling layer, yaitu *max_pooling2d* memiliki *output* (74, 74, 32). *Output* tersebut diperoleh dari proses *pooling input* yang diberikan dari hasil *output layer normalization* yaitu (148, 148, 32). Dilakukan *pooling* dengan ukuran *pool*-nya yaitu 2×2 sehingga proses ini mengurangi ukuran *feature* menjadi setengahnya. Oleh karena itu, dari *input* (148, 148, 32) dihasilkan *output* (74, 74, 32). Setelah *pooling layer*, diterapkan *dropout layer*. Sama seperti *normalization layer*, *dropout layer* juga tidak mengubah ukuran karena fungsi dari *dropout* adalah mengabaikan atau menonaktifkan beberapa neuron saja. Hal yang sama akan terus diulangi setiap setelah *layer convolution*, seperti yang telah dijelaskan sebelumnya.

Setelah semua rangkaian *layer convolution layer convolution* yang diikuti *layer normalization*, *pooling*, dan *dropout* telah selesai dengan hasil *output*

dari *layer* terakhir adalah (7, 7, 256). Maka, *output* ini akan menjadi *input* dari *layer* selanjutnya, yaitu *flatten layer*. Seperti yang telah dijelaskan sebelumnya, *flatten* mengubah vektor tiga dimensi menjadi vektor satu dimensi. Sehingga dari *input* (7, 7, 256) dilakukan proses *flatten* yang menghasilkan *output* menjadi (12544). Kemudian, *output* ini menjadi *input* untuk *layer dense* yang memiliki 512 unit, yang berarti menghasilkan *output* (512). Kemudian, dilakukan *normalization* dan *dropout* dengan proses yang sama seperti yang telah dijelaskan pada *convolution layer*. *Output* dari proses *dropout* menjadi *input* untuk *layer* terakhir, yaitu *dense_1*. Pada *layer* ini *output* yang dihasilkan adalah 20 sesuai dengan jumlah kelas dari *dataset* yang digunakan.

Jumlah parameter pada setiap *layer* dihitung berdasarkan ukuran *kernel*, filter, dan bias. Terlihat pada *summary* yang ditampilkan, pada *conv2d* dengan *output* (148, 148, 32) memiliki jumlah parameter yaitu 896 parameter. Artinya, parameter ini didapatkan dari hasil perhitungan *weights* ditambah bias menghasilkan 896 parameter. Perhitungan yang sama juga dilakukan terhadap *layer convolution*, *normalization*, dan *dense* lainnya.

2. *Training*

Tahap setelah membangun arsitektur model yaitu melakukan *training* atau pelatihan terhadap model agar dapat melakukan tugas klasifikasi gambar dengan baik. Seperti yang telah dijelaskan sebelumnya, penelitian ini menggunakan *library Keras*. Dalam *Keras*, metode yang umum digunakan adalah *model.fit* seperti yang ditampilkan pada kode berikut.

```
history = model.fit(train_generator,
epochs= 100,
verbose=1,
validation_data=validation_generator,
steps_per_epoch=train_generator.samples // batch_size,
validation_steps=validation_generator.samples // batch_size,
callbacks=[early_stopping, reduce_lr])
```

Terlihat pada kode tersebut bahwa *training* dilakukan dengan 100 *epochs* dan menggunakan *callback*. Terdapat dua *callback* yang digunakan selama *training*, yaitu *early stopping* dan *reduce learning rate*. *Callback early stopping* akan menghentikan proses *training* jika metrik *accuracy* tidak membaik selama 10 *epochs* secara berturut-turut. Sedangkan *callback reduce learning rate* akan mengurangi *learning rate* dengan faktor akar kuadrat dari 0.01 atau faktor 0.1, jika metrik akurasi tidak membaik selama 5 *epochs* secara berturut-turut.

```
early_stopping = EarlyStopping(monitor='accuracy',
patience=10, restore_best_weights=True)

reduce_lr = ReduceLROnPlateau(monitor='accuracy',
factor=np.sqrt(0.01), patience=5)
```

Metode *model.fit* merupakan metode yang umum digunakan pada berbagai *library machine learning*, termasuk *Tensorflow/Keras.* Pada penelitian ini, model akan melakukan *training* dengan menggunakan data *training* melalui *train_generator*. *Train_generator* ini akan menghasilkan *batch* data secara bertahap. Generator ini digunakan untuk membaca data dari penyimpanan dan memberikan data *augmentasi* secara *real-time*.

```
train_generator = train_datagen.flow_from_directory(
    train_set,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True,
    color_mode='rgb'
)
```

Data gambar akan dimuat langsung dari direktori penyimpanan (dalam penelitian ini alamat *google drive* data disimpan pada variabel *train_set*).

Kemudian, *train_generator* akan menghasilkan *batch* data dan label yang sesuai untuk proses *training* model. Data gambar yang telah dimuat, kemudian dilakukan proses augmentasi seperti yang sudah dijelaskan pada bagian 2 (Augmentasi Data). Ukuran gambar yang dihasilkan oleh generator kemudian diubah menjadi (150, 150) untuk memastikan bahwa semua gambar memiliki dimensi yang konsisten ketika dimasukkan ke dalam model. Banyak sampel yang akan diproses oleh model dalam satu iterasi *training* ditetapkan pada bagian *batch_size*=32. Sehingga, generator akan menghasilkan 32 gambar beserta labelnya dalam satu *batch*. Digunakan *shuffle=True* dengan tujuan agar model tidak belajar urutan dari gambar tertentu terdapat dalam *dataset*.

Data gambar yang dikembalikan oleh *train_generator* ini akan diberikan kepada model selama proses *training*. Sesuai dengan kode yang ditampilkan di atas, proses *training* data akan dibagi ke dalam beberapa *batch* yang akan diproses dalam satu *epoch*. *Train_generator* akan dipanggil sebanyak 84 untuk memproses seluruh *dataset* dalam satu *epoch*. Angka 84 diperoleh dari hasil bagi antara jumlah data *training* dengan jumlah *batch size* (lihat perhitungan (4. 1)).

$$steps\ per\ epoch = \frac{2688}{32} = 84 \qquad (4.\ 1)$$

Artinya, model akan memproses 32 gambar pada setiap *step* (*batch*) dan membutuhkan 84 *step* untuk memproses semua (2688) data gambar *training* dalam satu *epoch*. Tujuan dari penggunaan *batch* ini adalah untuk menghemat memori, model dapat memperbarui *weight* lebih sering (mempercepat konvergensi), dan memastikan model hanya memperoleh seluruh data gambar sekali dalam setiap *epoch*.

Pada *epoch* pertama dihasilkan *loss* 1.46 (146%) dan *accuracy* 0.54 (54%) dengan *validation loss* 1.53 (153%) dan *validation accuracy* 0.56 (56%). *Training* terus berjalan hingga berhenti pada *epoch* ke-70 dari penentuan jumlah *epoch* awal yaitu 100. *Training* berhenti karena *callback* memaksa proses *training* untuk berhenti karena tidak lagi sesuai dengan kriteria yang

ditentukan. Dalam hal ini, *training* berhenti setelah *accuracy* tidak lagi membaik dari *epoch* ke-60, berarti setelah 10 *epoch* nilai *accuracy* tidak lagi membaik, maka *training* dihentikan dengan mengembalikan *weights* yang terbaik dari *epoch*.

Loss yang dihasilkan dari *epoch* terakhir (*epoch* ke-70) adalah 0.78 (78%) dan *accuracy* 0.75 (75%) dengan *validation loss* 0.98 (98%) dan *validation accuracy* 0.73 (73%).

3. ***Convert* model ke format *.h5***

Setelah model selesai melakukan tahap *training*, model *machine learning* disimpan dalam format dile HDF5 (dengan ekstensi "*.h5*"). HDF5 (*Hierarchical Data Format version 5*) merupakan format *file* yang sangat fleksibel dan dapat digunakan untuk menyimpan data dalam jumlah besar dan kompleks. Selain itu, HDF5 didukung oleh banyak bahasa pemrograman seperti *Python, C, C++*, dan *Fortan*, serta alat analisis data lainnya. Dalam *machine learning*, format ini sering digunakan dengan *library* seperti *TrnsorFlow* dan *Keras.* Model yang disimpan dalam format HDF5 dapat dengan mudah diakses dan diperbarui. Hal tersebut berguna selama proses *training* di mana model dapat disimpan dan dimuat kembali. Dengan format HDF5, seluruh model, termasuk arsitektur dan bobot, disimpan dalam satu *file*, sehingga membuat model menjadi sangat efisien dan praktis.

Berikut kode untuk melakukan konversi model ke dalam format HDF5.

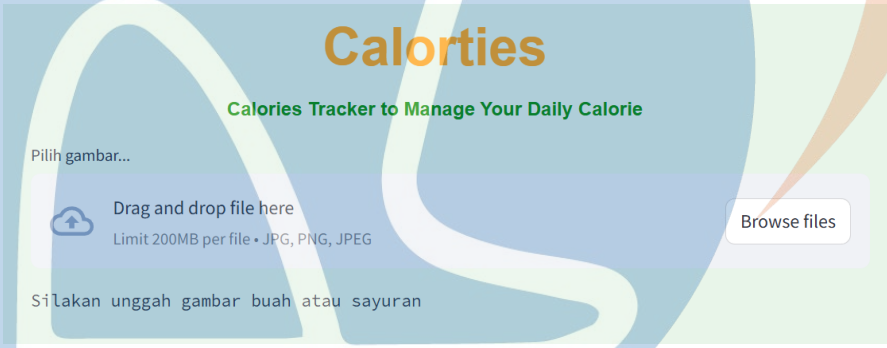
```
saved_model_path = "./calorties_model.h5"
model.save(saved_model_path)
```

4.2.4 *Testing*

Pada tahap *testing* model diimplementasikan pada aplikasi *web* menggunakan *framework Streamlit*. Streamlit merupakan *framework open-source* berbasis *Python* yang dirancang untuk memudahkan *developer* dalam mengembangkan aplikasi *web* interaktif, terutama yang berhubungan dengan *data science* dan

machine learning. *Streamlit* dipilih sebagai *framework* untuk *testing web* pada penelitian ini karena *Streamlit* sepenuhnya berbasis *Python*, sehingga mudah untuk melakukan integrasi dengan *library Python* lainnya, seperti *NumPy*, *TensorFlow*, dan sebagainya.

Aplikasi *web* untuk *testing* model dibangun dengan sederhana dengan tujuan utama untuk melihat apakah model dapat bekerja dengan baik setelah diimplementasikan ke aplikasi. Selain itu, pada aplikasi *web* ini nantinya akan menampilkan kandungan kalori dari gambar makanan yang di*input*. Di mana kalori yang ditampilkan nantinya akan didasarkan pada jumlah kalori per 100 gram makanan tersebut. Tampilan aplikasi dapat dilihat pada Gambar 4. 9 berikut.



Gambar 4. 9 Tampilan web testing model menggunakan streamlit

Pada *website* tersebut terdapat fitur *upload* gambar untuk mengunggah gambar buah-buahan atau sayuran yang akan dikonsumsi oleh pengguna. Pada penelitian ini, testing akan dilakukan dengan menggunakan 2 gambar buah dan sayur. Gambar yang digunakan yaitu 1 gambar kentang yang diperoleh dari internet dan 1 gambar kentang yang diambil menggunakan fitur kamera *handphone*.

Gambar 4. 10 Hasil testing gambar kentang dari internet

Testing pertama dilakukan dengan meng-*input* gambar kentang yang diambil dari internet. Kemudian, gambar diproses oleh model yang sudah dilatih sebelumnya. Model melakukan prediksi terhadap gambar kentang yang diunggah dan menghasilkan prediksi "kentang". Hal ini menunjukkan bahwa model berhasil mengenali gambar tersebut sebagai kentang dengan benar. Selain itu, model memiliki tingkat kepercayaan terhadap prediksi sebesar 90.67%. Tingkat kepercayaan yang dihasilkan cukup baik, artinya model sudah 90% yakin dengan hasil prediksi yang diberikan. Dari klasifikasi yang dilakukan maka akan ditampilkan total kalori dari makanan yang telah diprediksi oleh model. Terlihat pada gambar, bahwa kentang yang diprediksi memiliki total kalori sebesar 77 *kkal* per 100 gram.

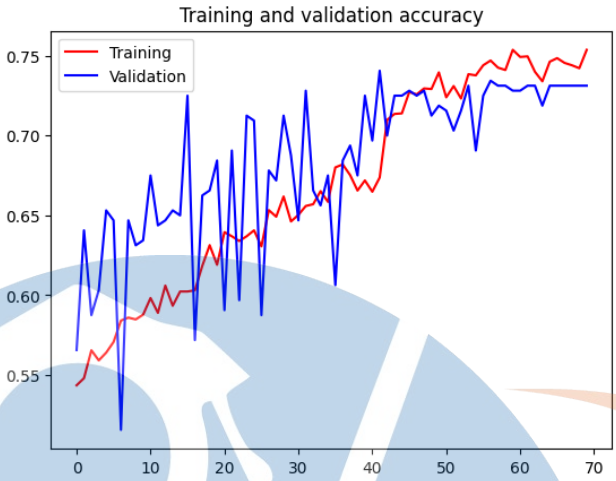
Gambar 4. 11 Hasil testing gambar kentang dari kamera handphone

Testing kedua dilakukan dengan mengunggah gambar kentang yang diambil dengan menggunakan kamera *handphone*. Pada Gambar 4. 11 terlihat bahwa model berhasil mengenali gambar sebagai kentang, sesuai dengan kelas asli dari gambar tersebut. Tingkat kepercayaan yang dihasilkan yaitu 99.75%, lebih tinggi dibandingkan gambar yang diambil dari internet. Di sini juga ditampilkan total kalori untuk gambar yang diprediksi sebagai kentang yaitu sebesar 77 *kkal* per 100 gram.

Dilihat dari hasil *testing*, secara keseluruhan, meskipun model berhasil mengklasifikasikan gambar sebagai kentang, model masih perlu ditingkatkan lagi untuk menghasilkan tingkat kepercayaan yang lebih tinggi.

4.2.5 Evaluasi

Dari hasil *training*, model menghasilkan *accuracy* sebesar 75% pada *epoch* terakhir dengan grafik perkembangan performa model dapat dilihat pada grafik yang ditampilkan pada Gambar 4. 12 dan Gambar 4. 13 berikut.

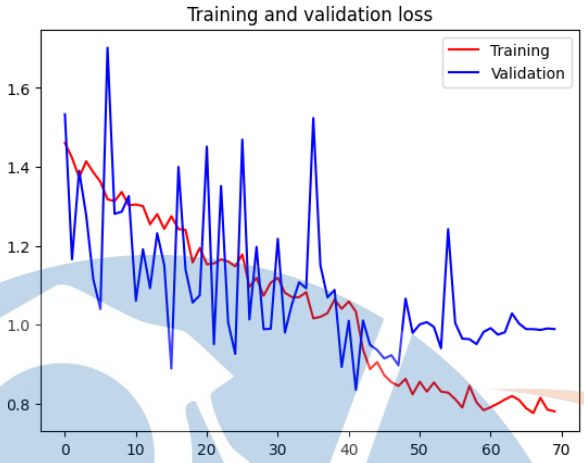


Gambar 4. 12 Grafik Accuracy Training dan Validation

Pada grafik yang ditampilkan pada Gambar 4. 12, terlihat bagaimana kenaikan *accuracy* selama proses *training* untuk setiap *epoch*. Sumbu X menunjukkan jumlah *epoch*, di mana setiap titik pada sumbu tersebut mewakili satu *epoch*. Sumbu Y menunjukkan nilai loss.

Berdasarkan grafik tersebut *accuracy training* meningkat secara konsisten dan stabil dengan beberapa fluktuasi kecil. Kenaikan konsisten ini menunjukkan bahwa model terus belajar dari data *training*. Sedangkan pada *accuracy validation* cendering berfluktuasi lebih banyak dibandingkan *accuracy training*. Ini dapat menunjukkan bahwa model mengalami beberapa *overfitting* pada data *training* di beberapa titik.

Dalam grafik tersebut, perbedaan antara grafik *accuracy training* dan grafik *accuracy validation* tidak terlalu besar, yang menunjukkan bahwa *overfitting* tidak terlalu parah atau model berhasil generalisasi dengan baik. Meskipun ada beberapa fluktuasi, *accuracy validation* juga meningkat seiring waktu, yang menunjukkan model cukup baik dalam generalisasi ke data baru.



Gambar 4. 13 Grafik loss Training dan Validation

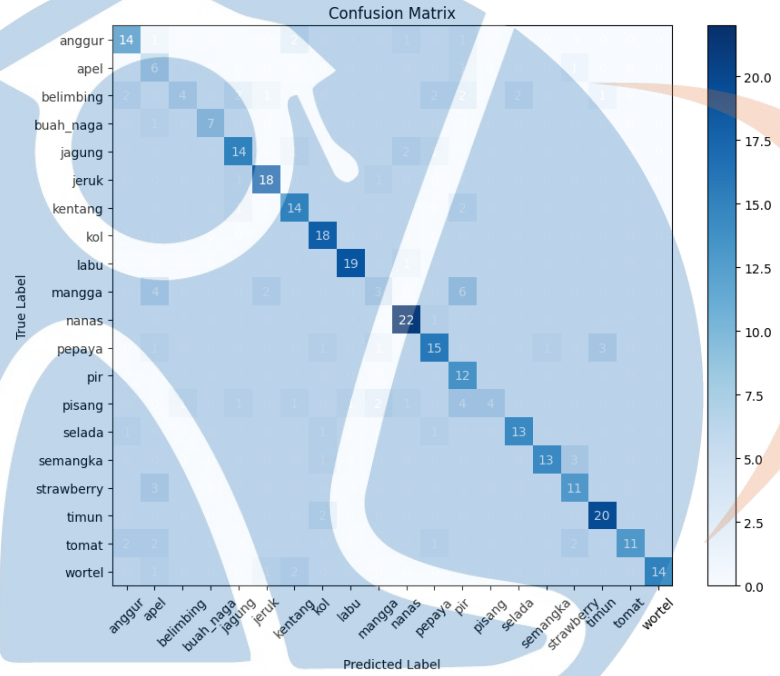
Sedangkan untuk *loss* pada *training* dan *validation* dapat dilihat pada Gambar 4. 13 di atas. Sama seperti pada grafik *accuracy*, sumbu X menunjukkan *epoch* dan sumbu Y menunjukkan nilai *loss*. Grafik garis merah mewakili *loss* pada data *training* selama *epoch*. Penurunan garis merah menunjukkan bahwa model sedang belajar dari data *training* dan mengurangi kesalahan prediksi. garis tersebut menunjukkan penurunan *loss* secara bertahap, yang menandakan bahwa model sedang belajar dan meningkatkan kemampuannya dalam melakukan prediksi data *training*.

Garis biru menunjukkan bagaimana *loss* model pada data *validation* selama *training* juga menurun seiring waktu. Namun, garis biru lebih berfluktuasi dibandingkan garis merah, terutama di awal *training*. Hal ini mengindikasikan bahwa model mengalami kesulitan untuk generalisasi pada awal *trining*. Namun, seiring bertambahnya *epoch*, fluktuasi ini cenderung berkurang, dan *loss validation* secara umum menunjukkan penurunan. Jika dilihat, baik *loss training* maupun *loss validation* terlihat menurun, yang menunjukkan bahwa *overfitting* tidak terlalu parah. Kedua *loss* (baik *training* maupun *validation*) menunjukkan penurunan yang stabil seiring waktu. Hal ini menandakan bahwa model konvergen dan belajar dengan baik.

Untuk memahami seberapa baik model yang telah dilatih dalam melakukan klasifikasi gambar yang diberikan, digunakan *confusion matrix*. Setiap baris

dalam matriks mewakili kelas yang sebenarnya, sedangkan setiap kolom mewakili kelas yang diprediksi.

Evaluasi menggunakan *confusion matrix* ini menggunakan *dataset* dari *testing*, artinya data yang dipersiapkan untuk *testing* dan belum pernah diproses dan dilihat oleh model. Berikut hasil *heatmap* dari *confusion matrix* pada model.



Gambar 4. 14 Confusion Matrix

Diagonal utama menunjukkan prediksi benar untuk setiap kelas yang diprediksi (*True Positive* dan *True Negative*). Di luar diagonal utama menunjukkan jumlah prediksi yang salah. *False Positive* ditunjukkan pada kolom kelas selain nilai pada diagonal utama, sedangkan *False Negative* berada pada baris kelas kecuali nilai pada diagonal utama.

Dari *confusion matrix* tersebut, kelas seperti "nanas" memiliki 22 jumlah prediksi benar, "labu" memiliki jumlah prediksi benar 19, dan "jeruk" memiliki jumlah prediksi benar yaitu 18. Artinya tiga kelas tersebut memiliki jumlah prediksi benar yang tinggi dibandingkan kelas lain. Hal tersebut menunjukkan bahwa model mampu mengenai gambar-gambar dari kelas ini dengan baik.

Namun, untuk kelas “belimbing” yang memiliki jumlah prediksi benar 4, “pisang” dengan prediksi benar 4, dan “mangga” dengan prediksi benar 3, artinya kelas-kelas ini memiliki jumlah prediksi benar yang lebih rendah. Hal tersebut, menunjukkan bahwa model kesulitan dalam mengenali gambar-gambar dari kelas ini.

Maka, dari nilai *confusion matrix* tersebut, dapat dihitung nilai *accuracy* dari model seperti berikut.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Accuracy = \frac{252}{336}$$

$$Accuracy = \frac{252}{336}$$

$$Accuracy = 0.75 \approx 75\%$$

Jadi, berdasarkan perhitungan tersebut, didapatkan akurasi model sebesar 75%. Namun, masih ada ruang untuk melakukan perbaikan agar model memiliki performa yang lebih baik lagi. Hal ini bisa dilihat dari *confusion matrix*, di mana terdapat beberapa kelas yang sering diprediksi sebagai kelas lain, misalnya kelas “belimbing” dan “pisang”. Artinya, model dapat diperbaiki lagi untuk meningkatkan kinerja secara keseluruhan, seperti menggunakan lebih banyak data *training* untuk kelas-kelas yang masih memiliki banyak kesalahan.

Maka, sesuai dengan hasil evaluasi yang telah dijelaskan di atas, untuk meningkatkan kinerja model perlu penambahan jumlah data pada proses *training* model. Terutama penambahan jumlah data pada beberapa kelas yang masih sering salah prediksi menjadi kelas lain oleh model, seperti kelas “mangga”, “belimbing”, dan “pisang”. Selain itu, penting juga untuk melakukan penambahan data pada kelas-kelas dengan distribusi *instance* yang belum seimbang, misal kelas “buah_naga” yang memiliki data lebih sedikit dibanding dengan kelas lainnya. Penambahan data ini bertujuan agar gambar yang akan diberikan pada model lebih bervariasi lagi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan dan Saran

5.1.1 Kesimpulan

Dari penelitian yang telah dilakukan, maka dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Model dikembangkan dengan menggunakan algoritma *Convolutional Neural Network* (CNN) dengan menggunakan data gambar buah-buah dan sayuran dengan total data sebanyak 3.363 gambar yang dikelompokkan ke dalam 20 kelas. Data kemudian dibagi menjadi 3 *subset*, yaitu *training* 80%, *validation* 10%, dan *testing* 10% dan masing-masing dilakukan *augmentasi* terhadap data. Data diberikan kepada model saat melakukan *training* hingga selesai pada *epoch* ke-70. Hasil dari *training* model kemudian dikonversi menjadi format HDF5 agar lebih fleksibel untuk diproses kembali.
2. Model memperoleh akurasi sebesar 75%. Namun, nilai akurasi ini tidak menjamin bahwa model selalu benar dalam melakukan klasifikasi data karena terdapat beberapa kelas yang masih sering salah prediksi oleh model.

5.1.2 Saran

Berdasarkan hasil yang diperoleh dari penelitian ini, maka penulis memberikan beberapa saran dalam upaya mengembangkan penelitian ini lebih lanjut dan memperoleh hasil yang lebih optimal. Adapun saran-saran tersebut adalah sebagai berikut:

1. Pada penelitian ini, beberapa kelas masih sering terjadi salah prediksi oleh model dan memiliki jumlah prediksi benar yang masih rendah. Terutama pada kelas “belimbing” dan “pisang” yang sering salah prediksi menjadi kelas lain. Oleh karena itu, dalam upaya meningkatkan akurasi model,

sangat disarankan digunakan data yang lebih banyak terutama untuk kelas-kelas yang masih sering salah prediksi dan memiliki jumlah prediksi benar yang rendah.

2. Akurasi model yang didapat dalam penelitian adalah 75% dan masih memerlukan peningkatan terhadap performa model. Oleh karena itu, disarankan untuk mempertimbangkan menggunakan metode lain seperti *transfer learning* atau metode *augmentasi* data dan ekstraksi fitur yang lebih baik untuk memperoleh hasil model yang lebih baik lagi.
3. Data yang digunakan pada penelitian ini terbatas hanya pada 20 jenis buah-buahan dan sayuran saja, disarankan memperluas variasi buah dan sayur yang digunakan. Lebih baik lagi menambahkan data makanan-makanan lainnya seperti nasi, lauk, dan makanan lainnya yang sering dikonsumsi di Indonesia. Hal ini bertujuan agar sistem dapat lebih mengenali berbagai macam jenis makanan di Indonesia dan tidak terbatas pada buah dan sayur saja.

DAFTAR PUSTAKA

[1] L. Haryo, "Siaran Pers HM.4.6/193/SET.M.EKON.3/4/2022 Menko Airlangga: Ekonomi Digital di Indonesia Tertinggi di Asia Tenggara."

[2] FAO, IFAD, UNICEF, WFP, and WHO, *The State of Food Security and Nutrition in the World 2022*. FAO, 2022. doi: 10.4060/cc0639en.

[3] S. Liza Munira, "Hasil Survei Status Gizi Indonesia (SSGI) 2022," Jakarta, Feb. 2023. Accessed: Sep. 06, 2023. [Online]. Available: https://ayosehat.kemkes.go.id/materi-hasil-survei-status-gizi-indonesia-ssgi-2022

[4] C. Lowe *et al.*, "The double burden of malnutrition and dietary patterns in rural Central Java, Indonesia," *Lancet Reg Health West Pac*, vol. 14, Sep. 2021, doi: 10.1016/j.lanwpc.2021.100205.

[5] I. K. Murni, D. C. Sulistyoningrum, R. Susilowati, M. Julia, and K. M. Dickinson, "The association between dietary intake and cardiometabolic risk factors among obese adolescents in Indonesia," *BMC Pediatr*, vol. 22, no. 1, Dec. 2022, doi: 10.1186/s12887-022-03341-y.

[6] T. Sudargo, H. Freitag, N. A. Kumayanti, and F. Rosiyani, *Pola Makan dan Obesitas*. Yogyakarta: UGM Press, 2018.

[7] L. D. Asih and M. Widyastiti, "MEMINIMUMKAN JUMLAH KALORI DI DALAM TUBUH DENGAN MEMPERHITUNGKAN ASUPAN MAKANAN DAN AKTIVITAS MENGGUNAKAN LINEAR PROGRAMMING," *Ekologia*, vol. 16, no. 1, pp. 38–44, 2016.

[8] Y. Bengio, I. Goodfellow, and A. Courville, "Deep Learning," 2015.

[9] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[10] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," May 27, 2015, *Nature Publishing Group*. doi: 10.1038/nature14539.

[11] Z. Zhang, P. Cui, and W. Zhu, "Deep Learning on Graphs: A Survey," Dec. 2018, [Online]. Available: http://arxiv.org/abs/1812.04202

[12] Y. Lecun, P. Haffner, L. Bottou, and Y. Bengio, “Object Recognition with Gradient-Based Learning.” [Online]. Available: http://www.research.att.com/~yann

[13] W. Fang, P. E. D. Love, H. Luo, and L. Ding, “Computer vision for behaviour-based safety in construction: A review and future directions,” Jan. 01, 2020, *Elsevier Ltd*. doi: 10.1016/j.aei.2019.100980.

[14] R. Chauhan, K. K. Ghanshala, and R. C. Joshi, *Convolutional Neural Network (CNN) for Image Detection and Recognition*. 2018.

[15] J. Moolayil, *Learn Keras for Deep Neural Networks: A Fast-Track Approach to Modern Deep Learning with Python*. Apress Media LLC, 2018. doi: 10.1007/978-1-4842-4240-7.

[16] D. H. Hubel and A. T. N. Wiesel, “54 With 2 plate and 20 text-ftgutre8 Printed in Gret Britain RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT’S VISUAL CORTEX,” 1962.

[17] M. E. ABDULFATTAH, L. NOVAMIZANTI, and S. RIZAL, “Super Resolution pada Citra Udara menggunakan Convolutional Neural Network,” *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 9, no. 1, p. 71, Jan. 2021, doi: 10.26760/elkomika.v9i1.71.

[18] S. T. , M. Sc. Prof. Dr. Suyanto, *Machine Learning Tingkat Dasar dan Lanjut*, 2nd ed. Bandung: Informatika Bandung, 2022.

[19] N. Buduma, N. Buduma, and J. Papa, *Fundamentals of Deep Learning*, 2nd ed. Sebastopol, CA 95472: O’Reily Media, Inc., 2022.

[20] W. Pedrycz and S.-M. Chen, “Studies in Computational Intelligence 865 Deep Learning: Algorithms and Applications,” 2020. [Online]. Available: http://www.springer.com/series/7092

[21] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, “A survey on modern trainable activation functions,” Jun. 01, 2021, *Elsevier Ltd*. doi: 10.1016/j.neunet.2021.01.026.

[22] Y. Wang, Y. Li, Y. Song, and X. Rong, “The influence of the activation function in a convolution neural network model of facial expression

recognition," *Applied Sciences (Switzerland)*, vol. 10, no. 5, Mar. 2020, doi: 10.3390/app10051897.

[23] S. KILIÇARSLAN, K. ADEM, and M. ÇELİK, "An overview of the activation functions used in deep learning algorithms," *Journal of New Results in Science*, vol. 10, no. 3, pp. 75–88, Dec. 2021, doi: 10.54187/jnrs.1011739.

[24] A. Ram and C. C. Reyes-Aldasoro, "The relationship between Fully Connected Layers and number of classes for the analysis of retinal images," Apr. 2020, [Online]. Available: http://arxiv.org/abs/2004.03624

[25] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, and E. A. Chavez-Urbiola, "Loss Functions and Metrics in Deep Learning," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.02694

[26] Y. Heo, "Loss Function Optimization for CNN Based Fingerprint Anti Spoofing," 2021. [Online]. Available: https://www.researchgate.net/publication/353325165

[27] S. H. Wang, K. Muhammad, J. Hong, A. K. Sangaiah, and Y. D. Zhang, "Alcoholism identification via convolutional neural network based on parametric ReLU, dropout, and batch normalization," Feb. 01, 2020, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s00521-018-3924-0.

[28] Norhikmah, A. Lutfhi, and Rumini, "The Effect of Layer Batch Normalization and Dropout of CNN model Performance on Facial Expression Classification," Yogyakarta, Aug. 2022. [Online]. Available: www.joiv.org/index.php/joiv

[29] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimed Tools Appl*, vol. 79, no. 19–20, pp. 12777–12815, May 2020, doi: 10.1007/s11042-019-08453-9.

[30] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," Dec. 01, 2022, *Elsevier B.V.* doi: 10.1016/j.array.2022.100258.

[31] Amity University, Institute of Electrical and Electronics Engineers, and Institute of Electrical and Electronics Engineers. United Kingdom and Republic

of Ireland Section, *Enhancing Performance of Deep Learning Models with different Data Augmentation Techniques: A Survey*. 2020.

[32] S. Bera and V. K. Shrivastava, "Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification," *Int J Remote Sens*, vol. 41, no. 7, pp. 2664–2683, Apr. 2020, doi: 10.1080/01431161.2019.1694725.

[33] J. Zhang, "Gradient Descent based Optimization Algorithms for Deep Learning Models Training," Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.03614

[34] S. Hikmat Haji and A. Mohsin Abdulazeez, "COMPARISON OF OPTIMIZATION TECHNIQUES BASED ON GRADIENT DESCENT ALGORITHM: A REVIEW," Guhok, 2021.

[35] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview," vol. 1, Aug. 2020, [Online]. Available: http://arxiv.org/abs/2008.05756

[36] A. ANHAR and R. A. PUTRA, "Perancangan dan Implementasi Self-Checkout System pada Toko Ritel menggunakan Convolutional Neural Network (CNN)," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 11, no. 2, p. 466, Apr. 2023, doi: 10.26760/elkomika.v11i2.466.

[37] I. Made Riartha Prawira and M. Syahrul Mubarok, "Klasifikasi Multi-Label Pada Topik Berita Berbahasa Indonesia Menggunakan Multinomial Naïve Bayes," 2018.

[38] R. E. Jayaputra, "Estimasi Kalori pada makanan melalui Citra Makanan menggunakan Model SSD (Single Shot Detector) pada Mobile Device," Yogyakarta, Oct. 2020.

[39] I. P. A. E. D. Udayana and P. G. S. C. Nugraha, "PREDIKSI CITRA MAKANAN MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK UNTUK MENENTUKAN BESARAN KALORI MAKANAN," Jan. 2020. [Online]. Available: http://www.depkes.go.id

[40] G. Wicaksono, S. Andryana, and Benrahman, "Aplikasi Pendeteksi Penyakit Pada Daun Tanaman Apel Dengan Metode Convolutional Neural Network," 2020. doi: https://doi.org/10.31328/jointecs.v5i1.1221.

[41] Okpatrioka, "Research And Development (R&D) Penelitian Yang Inovatif Dalam Pendidikan," *DHARMA ACARIYA NUSANTARA : Jurnal Pendidikan, Bahasa dan Budaya*, vol. 1, pp. 86–100, 2023.

[42] K. Seth, "Fruits and Vegetables Image Recognition Dataset," Kaggle.com. Accessed: Nov. 15, 2023. [Online]. Available: https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition

[43] M. I. Ahmed and S. M. Mamun, "Vegetable Image Dataset," Kaggle. Accessed: Mar. 24, 2024. [Online]. Available: https://www.kaggle.com/dsv/2965251

[44] S. Maher, "Fruits Dataset (Images)," Kaggle. Accessed: Mar. 24, 2024. [Online]. Available: https://www.kaggle.com/datasets/shreyapmaher/fruits-dataset-images

[45] K. Seth, "Fruits and Vegetables Image Recognition Dataset," Kaggle. Accessed: Mar. 24, 2023. [Online]. Available: https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition

[46] M.-D. Minut *et al.*, "Fruits-262 dataset: A dataset containing a vast majority of the popular and known fruits," Kaggle. Accessed: Oct. 05, 2023. [Online]. Available: https://www.kaggle.com/datasets/aelchimminut/fruits262

[47] FatSecret, "FatSecret Indonesia." Accessed: Jul. 06, 2024. [Online]. Available: https://www.fatsecret.co.id/

[48] S. Tang, S. Yuan, and Y. Zhu, "Data Preprocessing Techniques in Convolutional Neural Network Based on Fault Diagnosis towards Rotating Machinery," *IEEE Access*, vol. 8, pp. 149487–149496, 2020, doi: 10.1109/ACCESS.2020.3012182.