KLASIFIKASI DAUN TEH SIAP PANEN MENGGUNAKAN CNN DENGAN ARSITEKTUR *MOBILENETV2*

SKRIPSI

FEBRIADY MARPAUNG

188160032

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MEDAN AREA
MEDAN
2023

KLASIFIKASI DAUN TEH SIAP PANEN MENGGUNAKAN CNN DENGAN ARSITEKTUR *MOBILENETV2*

SKRIPSI

Diajukan sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana di Fakultas Teknik
Universitas Medan Area

Oleh:

FEBRIADY MARPAUNG
188160032

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MEDAN AREA
MEDAN
2023

HALAMAN PENGESAHAN

Judul Skripsi	: Klasifikasi Daun Teh Siap Panen Menggunakan CNN Dengan Arsitektur *MobileNetV2*
Nama	: Febriady Marpaung
NPM	: 188160032
Fakultas	: Teknik
Prodi	: Teknik Informatika

Disetujui Oleh
Komisi Pembimbing

Nurul Khairina, S.Kom., M.Kom
Pembimbing I

Dr. Rahmad Syah, S.Kom., M.Kom
Dekan Fakultas Teknik

Rizki Muliono, S.Kom., M.Kom
Ketua Prodi

Tanggal Lulus : 04 Agustus 2023

HALAMAN PERNYATAAN

Saya menyatakan bahawa skripsi yang saya susun, sebagai syarat memperoleh gelar sarjana merupakan hasil karya tulis saya sendiri. Adapun bagian-bagian tertentu dalam penulisan skripsi ini yang saya kutip dari hasil karya orang lain telah dituliskan sumbernya secara jelas sesuai dengan norma, kaidah, dan etika penulisan ilmiah.

Saya bersedia menerima sanksi pencabutan gelar akademik yang saya peroleh dan sanksi-sanksi lainnya dengan peraturan yang berlaku, apabila di kemudian hari ditemukan adanya plagiat dalam skripsi ini.

Medan, 04 Agustus 2023

Febriady Marpaung
188160032

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR/SKRIPSI/TESIS UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Medan Area, saya yang bertanda tangan di bawah ini :

Nama	: Febriady Marpaung
NPM	: 188160032
Program Studi	: Teknik Informatika
Fakultas	: Teknik
Jenis Karya	: Skripsi

Demi penembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Medan Area **Hak Bebas Royaliti Noneksklusif (Non-exclucive Royality-Free Right)** atas karya ilmiah saying yang berjudul : **Klasifikasi Daun Teh Siap Panen Menggunakan CNN Dengan Arstiektur *MobileNetV2***

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royaliti Noneksklusif ini Universitas Medan Area berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan skripsi saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Medan

Pada tanggal : 04 Agustus 2023

Yang menyatakan

(Febriady MArpaung)

RIWAYAT HIDUP

Penulis dilahirkan di Pondok Atas, pada tanggal 7 Februari 2001 dari ayah Aser Marpaung dan ibu Nurmaya Siahaan. Penulis merupakan putra pertama dari 4 (empat) bersaudara. Penulis menyelesaikan pendidikan sekolah dasar di SD 095181 Mekar Sidamanik pada tahun 2012. Pada tahun yang sama penulis melanjutkan pendidikan sekolah menengah pertama di SMPN 1 Sidamanik selama 3 (tiga) tahun dan selesai pada tahun 2015. Penulis melanjutkan tingkat pendidikan selanjutnya di SMAN 1 Sidamanik dan lulus pada tahun 2018. Tahun 2018 penulis juga terdaftar sebagai mahasiswa Teknik Informatika di Universitas Medan Area.

Selama mengikuti perkuliahan, Penulis melaksanakan Kerja Praktek (KP) di UPT. PDOLPD Provinsi Sumatera Utara.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Kuasa atas segala karuniaNya sehingga skripsi ini berhasil diselesaikan. Tema yang dipilih dalam penelitian ini ialah Deep Learning dengan judul "Klasifikasi Daun Teh Siap Panen Menggunakan CNN Dengan Arsitektur *MobileNetV2*".

Skripsi ini merupakan salah satu syarat untuk menyelesaikan pendidikan untuk mencapai gelar sarjana di Program Studi Teknik Informatika Fakultas Teknik Universitas Medan Area. Pada kesempatan ini penulis mengucapkan terima kasih kepada :

1. Bapak Prof. Dr. Dadan Ramdan, M.Eng, M.Sc. selaku Rektor Universitas Medan Area.
2. Bapak Dr. Rahmad Syah, S.Kom., M. Kom selaku Dekan Fakultas Teknik Universitas Medan Area.
3. Bapak Rizki Muliono, S.Kom., M.Kom selaku Kepala Program Studi Teknik Informatika Universitas Medan Area.
4. Ibu Nurul Khairina S.Kom., M.Kom selaku Dosen pembimbing yang telah membantu penulis dari segi materi dan moril sehingga penulis dapat menyelesaikan skripsi ini.
5. Orang tua penulis yaitu Bapak Aser Marpaung dan Ibu Nurmaya Siahaan dan nenek penulis yaitu Sorta Siahaan yang telah mendoakan tiada henti dan memberikan semangat serta membantu penulis dalam segi materi dan moril sehingga penulis dapat menyelesaikan skripsi ini dengan sebaik baiknya.
6. Adik penulis yaitu Padekan Marpaung, Desi Artia Favor Marpaung dan Devi Artia Favor Marpaung yang telah memberikan semangat kepada penulis dalam menyelesaikan skripsi ini dengan sebaik baiknya.
7. Seluruh Dosen dan Staf Program Studi Teknik Informatika Universitas Medan Area.
8. Teman dekat penulis, Jesica Paulina Damanik yang selalu memberikan semangat selama penulisan proposal skripsi ini.

9. Seluruh teman-teman yang sudah memberikan dukungannya selama penulisan proposal skripsi ini, khususnya teman-teman Teknik Informatika angkatan 2018.
10. Seluruh pihak yang tidak dapat disebutkan satu persatu yang membantu dalam penyelesaikan skripsi ini

Penulis menyadari bahwa tugas akhir/skripsi/tesis ini masih memiliki kekurangan, oleh karena itu kritik dan saran yang bersifat membangun sangat penulis harapkan demi kesempurnaan tugas akhir/skripsi/tesis ini. Penulis berharap tugas akhir/skripsi/tesis ini dapat bermanfaat baik untuk kalangan pendidikan maupun masyarakat. Akhir kata penulis ucapkan terima kasih.

Medan, 04 Agustus 2023

Penulis,

(Febriady Marpaung)

ABSTRAK

Penentuan daun teh siap panen merupakan faktor penting dalam menentukan kualitas dan nilai jual dari produk teh. Klasifikasi daun teh siap panen dapat membantu para petani teh dan produsen teh dalam menentukan waktu yang tepat untuk memetik daun teh sehingga dapat menghasilkan teh dengan kualitas terbaik. Oleh karena itu diperlukan ada pendekatan digital agar dapat mengenali daun teh siap panen dengan cepat dan akurat. Penelitian ini menggunakan metode *Convolutional Neural Network* (CNN) dengan arsitektur MobileNetV2 dalam melakukan klasifikasi tingkat daun teh siap panen daun teh. Berdasarkan hasil *training* pada 6 skenario model yang diuji, diperoleh tingkat akurasi tertinggi pada pengujian skenario *model* 2 sebesar 100% menggunakan *hyperparameter epoch* 50, *input shape* : 224x224x3 (*RGB channel*), *batch size* : 32, dan *optimizer* : *Adam*. Hasil akurasi dari model setelah diuji menggunakan *data testing* (100 citra per *class*) didapatkan hasil akurasi sebesar 100% dengan nilai presisi (*precision*) sebesar 100%, *recall* 100%, dan *f1-score* 100%.

Kata Kunci : Klasifikasi, Daun teh siap panen, Daun Teh, Convolutional Neural Network, MobileNetV2

ABSTRACT

The determination of ready-to-harvest tea leaves is a crucial factor in determining the quality and market value of tea products. Classifying ready-to-harvest tea leaves can assist tea farmers and tea producers in identifying the right time to pluck tea leaves to produce the highest-quality tea. Therefore, a digital approach is needed to quickly and accurately recognize ready-to-harvest tea leaves. This research utilizes the Convolutional Neural Network (CNN) method with the MobileNetV2 architecture to classify the readiness level of tea leaves. Based on the training results of six tested model scenarios, the highest accuracy was obtained in scenario model 2 at 100% using hyperparameters: 50 epochs, input shape: 224x224x3 (RGB channel), batch size: 32, and optimizer: Adam. The accuracy of the model after testing with testing data (100 images per class) yielded an accuracy of 100%, with precision, recall, and f1-score all at 100%.

Keywords: Classification, Ready-to-harvest tea leaves, Tea Leaves, Convolutional Neural Network, MobileNetV2

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia memiliki berbagai macam flora, salah satunya adalah tanaman teh (*Camellia Sinensis*), dimana termasuk kedalam salah satu tanaman perdu yang berwarna hijau (*Evergreen Shrub*) yang dapat tumbuh mencapai ketinggian 6 sampai 9 meter. Indonesia juga merupakan salah dari dari 7 produsen teh terbesar di dunia (Auliasari dkk., 2020). Sumatera utara merupakan salah satu daerah penghasil teh yang memiliki tiga perkebunan teh di Simalungun, yaitu Kebun Teh Bahbutong, Kebun Teh Sidamanik dan Kebun Teh Tobasari. Dimana ketiga perkebunan ini berada dalam jarak yang berdekatan, tetapi masyarakat lebih mengenal produk dari tiga perkebunan ini sebagai teh Sidamanik, karena dahulu ketiga perkebunan teh tersebut berada di Kecamatan Sidamanik (Damanik dkk., 2022). Teh Sidamanik dikenal sebagai salah satu teh berkualitas di Indonesia. Perkebunan teh Sidamanik merupakan produsen teh hitam terbesar kedua di Indonesia setelah Jawa Barat (Nadila dkk., 2022).

Pada perkebunan PTPN IV Sidamanik, pihak perusahaan pengelola masih menggunakan cara konvensional dalam menentukan daun teh siap panen berdasarkan aturan gilir petik. Aturan ini disesuaikan dengan laju pertumbuhan daun teh dan umur pemangkasan terakhir terhadap daun teh. Penerapan aturan gilir petik menyebabkan pihak perusahaan tidak dapat mengetahui seluruh daun teh siap panen daun teh karena luasnya areal kebun teh yang memiliki waktu panen berbeda dan tingkat daun teh siap panen yang berbeda pada setiap blok tamanan teh.

Teh adalah salah satu komoditas penting yang dikonsumsi oleh masyarakat di seluruh dunia, khususnya di Indonesia. Penentuan daun teh siap panen merupakan faktor penting dalam menentukan kualitas dan nilai jual dari produk teh. Klasifikasi daun teh siap panen dapat membantu para petani teh dan produsen teh dalam menentukan waktu yang tepat untuk memetik daun teh sehingga dapat menghasilkan teh dengan kualitas terbaik. Oleh karena itu diperlukan ada pendekatan digital agar dapat mengenali tingkat daun teh siap panen dengan cepat

dan akurat. Pada penelitian ini akan menggunakan metode *Convolutional Neural Network* (CNN) dengan arsitektur *MobileNetV2* dalam melakukan klasifikasi daun teh siap panen daun teh.

CNN merupakan salah satu algoritma *Deep Learning* yang dirancang untuk mengolah data dalam bentuk data dua dimensi, misalnya gambar atau suara (Akmal Hariz dkk., 2022). CNN memiliki kemampuan menerima inputan berupa citra untuk dipelajari dan bisa menentukan aspek atau objek apa saja yang ada di dalam sebuah citra dan dapat digunakan atau diolah oleh mesin untuk dapat membedakan suatu gambar dengan gambar lainnya (Mudzakir dan Arifin, 2022). CNN dapat melakukan ekstraksi fitur dari citra secara otomatis dan kemudian melakukan klasifikasi berdasarkan fitur-fitur tersebut. Salah satu arsitektur *model Deep Learning* menggunakan CNN yang populer adalah *MobileNet*.

MobileNetV2 adalah modifikasi pertama dari *MobileNet*. Perbedaan *MobileNetV1* dengan *MobileNetV2* adalah penggunaan *inverted residual blocks* dan *linear bottleneck*. Dalam *bottlenec*k terdapat input dan *output* antara *model* sementara lapisan dalam merangkum kemampuan *model* untuk mengubah *input* dari tingkat yang lebih rendah (yaitu piksel) ke deskriptor dengan tingkat yang lebih tinggi (Dwijayana dan Wibawa, 2022). Pemilihan metode CNN dengan arsitektur *MobileNetV2* dalam penelitian ini didasari pada efektifitas metode dalam pemanfaatan *resource* yang tidak terlalu berat dan bisa menghasilkan akurasi yang maksimal berdasarkan hasil dari penelitian-penelitian sebelumnya.

Dalam penelitian (Ramayanti dkk., 2022) tentang klasifikasi citra kupu-kupu dengan membandingkan *model* arsitektur *VGG16* dan *MobileNetV2*, diperoleh hasil bahwa arsitektur *MobileNetV2* lebih unggul dibandingkan arsitektur *VGG16*, dengan akurasi terbaik diperoleh *MobileNetV2* tanpa *fine-tuning* yaitu mencapai presentase 96%. Penelitian lain dilakukan oleh (Zaelani dan Miftahuddin, 2022) tentang identifikasi jenis buah-buahan berdasarkan fitur daun dengan membandingkan *model* arsitektur *Efficientnet-B3* dan *model* arsitektur *MobilenetV2*, diperoleh hasil bahwa *model* arsitektur *MobileNetV2* lebih unggul dibandingkan dengan model arsitektur *Efficientnet-B3*. Hasil pengujian model *MobileNetV2* dengan menggunakan *hyperparameter epoch* 20, *optimizer Adam*

menghasilkan akurasi 99%, sedangkan model *MobileNetV2* dengan *hyperparameter epoch* 50 *optimizer Adamax* menghasilkan akurasi sebesar 98%.

Dalam klasifikasi daun teh siap panen, arsitektur *MobileNetV2* dapat digunakan untuk memproses citra daun teh dan melakukan klasifikasi. Pertama-tama, *model MobileNetV2* dilatih menggunakan *dataset* citra daun teh yang telah diberi label daun teh siap panen. Kemudian, *model* tersebut dapat digunakan untuk memprediksi daun teh siap panen daun teh pada citra baru yang dimasukkan ke dalam *model*. Dengan menggunakan metode CNN dengan arsitektur *MobileNetV2*, klasifikasi daun teh siap panen daun teh dapat dilakukan secara otomatis dan akurat. Hal ini dapat membantu dalam meningkatkan kualitas dan nilai jual dari produk teh yang dihasilkan.

Penelitian yang berkaitan dengan klasifikasi daun teh serta identifikasi daun teh siap panen pada umumnya telah dilakukan oleh beberapa peneliti-peneliti sebelumnya. Dalam penelitian yang dilakukan oleh (Jaelani dkk., 2020) menggunakan metode CNN dengan arsitektur *Mobilenet* menghasilkan akurasi sebesar 60%. Dalam penelitian selanjutnya dilakukan oleh (Suherman dkk., 2021) menggunakan metode CNN dengan arsitektur *LeNet-5* menghasilkan akurasi sebesar 94.55%.

Dalam penelitian (Wicaksono, 2019) tentang identifikasi daun teh siap panen menggunakan *Centroid Clustering* berbasis ruang warna *YCbCr* menghasilkan nilai akurasi sebesar 80%. Pada penelitian (Auliasari dkk., 2020) menggunakan metode ektraksi ciri seperti HIS (*Hue Saturation Intensity*) dan HSV (*Hue Saturation Value*) menghasilkan akurasi 100% pada fitur warna HIS dan 83,33% pada fitur warna HSV. Penelitian lain juga dilakukan oleh (Ibrahim dkk., 2022) dengan membandingkan arsitektur *VGGNet19* dan arsitektur *ResNet50*. Hasil penelitian menunjukkan bahwa arsitektur *VGGNet19* memperoleh hasil yang lebih baik karena memiliki nilai akurasi sebesar 97.5%.

Berdasarkan latar belakang serta penelitian terkait yang telah dijelaskan sebelumnya, maka pembaruan pada penelitian ini yang membedakan dengan penelitian terdahulu, yaitu terletak pada penerapan metode CNN dengan arsitektur *MobileNetV2* yang digunakan dalam mengklasifikasi tingkat daun teh siap panen

pada daun teh. Penelitian ini diberi judul "Klasifikasi Daun Teh Siap Panen Daun Teh Menggunakan CNN Dengan Arsitektur *MobileNetV2*".

1.2 Perumusan Masalah

Merujuk pada latar belakang masalah yang telah diuraikan serta hasil dari penelitian sebelumnya, maka dapat dirumuskan pokok permasalahan yang akan dibahas dalam penelitian ini, yaitu sebagai berikut:

1. Bagaimana menerapkan metode *Convolutional Neural Network* dengan arsitektur *MobileNetV2* dalam mengklasifikasi daun teh siap panen daun teh.
2. Bagaimana tingkat akurasi yang dihasilkan metode *Convolutional Neural Network* dengan arsitektur *MobileNetV2* dalam penerapannya pada proses klasifikasi tingkat daun teh siap panen.

1.3 Batasan Masalah

Permasalahan yang dibahas pada penelitian ini memiliki ruang lingkup yang luas, sehingga diberikan batasan-batasan agar penelitian lebih terarah dan sesuai dengan tujuan yang ingin dicapai, yaitu sebagai berikut:

1. Objek penelitian adalah daun teh verietas *Assamica klon Gambung* (GMB) 7 pada blok 12B di PTPN IV Sidamanik.
2. Jumlah data eksperimen yang digunakan yaitu 2000 *dataset* gambar citra daun teh dengan format *.jpg/jpeg* yang diambil dengan menggunakan kamera *smartphone*.
3. *Dataset* terbagi menjadi 3 yaitu data *training* dengan persentase 70%, data *validation* 20%, dan data *testing* 10%.
4. Hasil klasifikasi daun teh siap panen terbagi menjadi dua kelas, yaitu (a) daun teh peko (b) daun teh kepel.
5. Penelitian ini tidak membahas tentang permasalahan dalam klasifikasi tingkat daun teh siap panen daun teh secara *real-time*.
6. Algoritma *Deep Learning* yang digunakan adalah *Convolutional Neural Network* dengan arsitektur *MobileNetV2*.

7. Bahasa pemrograman yang digunakan untuk melakukan *training model* adalah *Python 3.10.11* dengan menggunakan bantuan *Google Collaboratory*.
8. Pengujian *model* menggunakan 6 skenario, yaitu berdasarkan perbandingan jumlah *epoch* (20 dan 50) serta *learning rate* (0.0003 dan 0.0005). Sedangkan parameter dari *input shape* : 224x224x3 (*RGB channel*), *batch size* : 32, dan *optimizer* : *Adam* dijadikan statis pada setiap skenario model yang diuji.
9. Pengujian *model* dengan data baru (*data testing*) akan di *deploy* kedalam aplikasi berbasis *web* menggunakan bahasa pemrograman *Python* dengan *Flask* sebagai *library* utama.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah maka tujuan yang ingin dicapai dalam penelitian ini, yaitu sebagai berikut:

1. Membangun sebuah *model* yang dapat mengklasifikasi daun teh siap panen menggunakan metode *Convolutional Neural Network* dengan arsitektur *MobileNetV*2 sehingga dapat membantu pihak perkebunan teh dalam meningkatkan kualitas dan nilai jual dari produk teh yang dihasilkan.
2. Menganalisa performansi dari tingkat akurasi *model* dari metode *Convolutional Neural Network* dengan arsitektur *MobileNetV2* sehingga dapat menghasilkan klasifikasi daun teh siap panen secara akurat.

1.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini, baik secara teoritis maupun secara praktis antara lain yaitu sebagai berikut:

1. Secara Teoritis

   Memberikan pemahaman yang lebih luas tentang prosedur dalam proses kerja algoritma *Convolutional Neural Network* dengan

menggunakan arsitektur *MobileNetV2*, khususnya untuk klasifikasi daun teh siap panen.

2. Secara Praktis
   a. Bagi Peneliti

      Kegiatan penelitian ini dapat menambah pengetahuan, wawasan dan pengalaman serta sebagai sarana untuk menuntaskan jenjang pendidikan yang sedang dilaksanakan.

   b. Bagi Universitas

      Hasil penelitian ini dapat menambah bahan referensi untuk penelitian lain dibidang *Deep Learning* dengan algoritma *Convolutional Neural Network* menggunakan arsitektur *MobileNetV*2, khusunya pada Program Studi Teknik Informatika Fakultas Teknik Universitas Medan Area.

1.6 Sistematika Penulisan

Sistematika penulisan dalam penyusunan skripsi ini dikelompokkan menjadi lima bab, yaitu sebagai berikut:

BAB I : PENDAHULUAN

Pada bab ini dijelaskan terkait penyampaian masalah yang dikemas melalui latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian serta sistematika penulisan pada penelitian.

BAB II : TINJAUAN PUSTAKA

Pada bab ini menguraikan rangkuman teori yang dihimpun dari berbagai pustaka yang relevan dengan topik yang menjadi objek kajian, untuk memperluas basis informasi dalam melakukan kajian dan akan digunakan sebagai basis argumentasi di dalam mengemukakan pandangan. Dasar teori yang dipakai dalam penelitian ini yaitu mengenai *Deep Learning*, *Convolutional Neural Network*, Arsitektur *MobileNetV2* dan beberapa teori pendukung lainnya.

BAB III : METODOLOGI PENELITIAN

Pada bab ini menjelaskan tentang tahapan-tahapan yang akan dilakukan dalam penelitian ini. Setiap rencana dari tahapan-tahapan penelitian dideskripsikan secara rinci.

BAB IV : HASIL DAN PEMBAHASAN

Bab ini berisi hasil penelitian serta pembahasan dari penelitian yang telah dilaksanakan. Dalam bab ini disajikan gambar, tabel, serta grafik dari hasil penelitian yang telah dilakukan.

BAB V : KESIMPULAN DAN SARAN

Bab ini merupakan kesimpulan dari keseluruhan pembahasan terutama hasil penelitian yang telah dijelaskan pada bab sebelumnya. Bab ini juga berisi saran untuk penelitian selanjutnya yang akan mengangkat tema yang sama dengan penelitian ini.

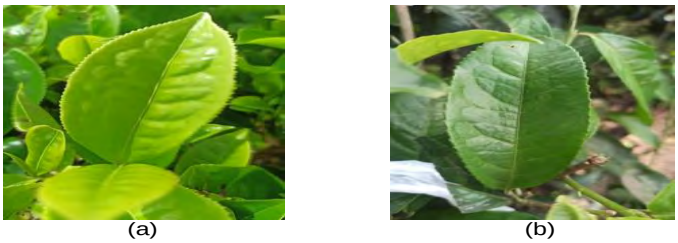
BAB II
TINJAUAN PUSTAKA

2.1 Tanaman Teh

Teh merupakan salah satu produk perkebunan yang memegang peranan penting dalam perekonomian Indonesia sebagai sumber pendapatan dan devisa negara serta menyediakan lapangan pekerjaan bagi masyarakat lokal dan untuk pembangunan daerah (Kementan RI, 2015). Tanaman teh (*Camellia sinensis (L). O. Kuntze)* termasuk kedalam falimi *Theaceae* dengan genus *Camellia*. Beberapa kultivar teh penting telah diketahui yaitu varietas China (*Camellia sinensis var. sinensis)*, varietas Assam (*Camellia sinensis ver. assamica)*, Kamboja, dan hibridanya berupa klon yang direkomendasikan. Tanaman teh yang ditanam di Indonesia 99% merupakan teh Assam karena jumlah produksi dan kualitas hasil teh yang tinggi. Tanaman dapat dipetik setelah mencapai usia tiga tahun. Daun yang dapat diambil meliputi pucuk atau tunas yang sedang tumbuh aktif yang disebut Peko, dan daun yang terletak di ketiak daun tempat ranting tumbuh yang disebut Daun kepel (Rahmat, 2020).

Penelitian ini bertujuan untuk mengklasifikasikan daun teh siap panen dengan menerapkan metode *Convolutional Neural Network* dengan arsitektur *MobileNetV2* yang merupakan salah satu metode *Machine Learning* yang memiliki kemampuan untuk mengolah informasi citra. Dengan menggunakan metode CNN dengan arsitektur *MobileNetV2*, klasifikasi daun teh siap panen dapat dilakukan secara otomatis dan akurat sehingga dapat membantu pihak perkebunan teh dalam meningkatkan kualitas dan nilai jual dari produk teh yang dihasilkan.

Hasil klasifikasi tingkat daun teh siap panen daun teh terbagi menjadi dua kelas, yaitu (a) daun teh peko dan (b) daun teh kepel. Gambar 2.1 merupakan contoh citra daun teh yang digunakan dalam penelitian ini.

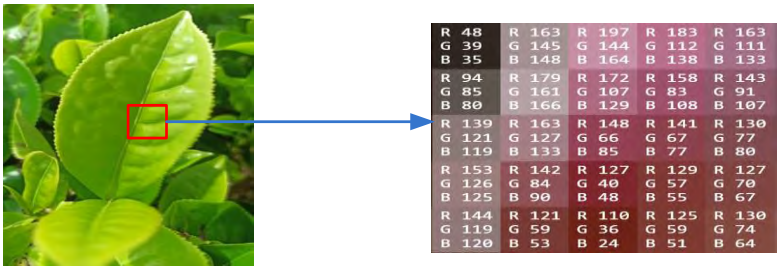


Gambar 2.1 Contoh Citra Daun Teh (a) Peko (b) Kepel

2.2 Citra Digital

Citra adalah suatu bentuk representasi atau gambaran dari suatu objek. Citra dapat berupa foto optik, sinyal video *analog* seperti gambar pada layar TV, atau format digital yang dapat disimpan di media penyimpanan sebagai *output* dari sistem perekaman data. Citra digital terdiri dari sebuah matriks 2 dimensi dengan M kolom dan N baris, yang disebut piksel. Piksel adalah elemen terkecil pada citra yang memiliki nilai untuk merepresentasikan warna pada layar monitor. Terdapat tiga jenis citra digital, yaitu citra *binary* (monokrom), citra skala keabuan (*grayscale*), dan citra warna (*true color*). Citra warna menggunakan tiga nilai piksel, yaitu merah, hijau, dan biru, untuk membentuk berbagai macam warna yang dapat ditampilkan pada layar monitor. (Allaam dan Wibowo, 2021).

Citra *true color* memiliki piksel-piksel yang terdiri dari tiga komponen warna: merah (R), hijau (G), dan biru (B). Setiap komponen warna memiliki rentang nilai antara 0 hingga 255 (8 bit). Warna pada setiap piksel dihasilkan dari kombinasi nilai-nilai R, G, dan B yang membentuk total kemungkinan warna sebanyak 2553 = 16.581.375. Ukuran total *bit* untuk setiap piksel adalah 24 *bit* (8 *bit* untuk masing-masing R, G, dan B).. Contoh dari citra *true color* dapat dilihat pada Gambar 2.2.



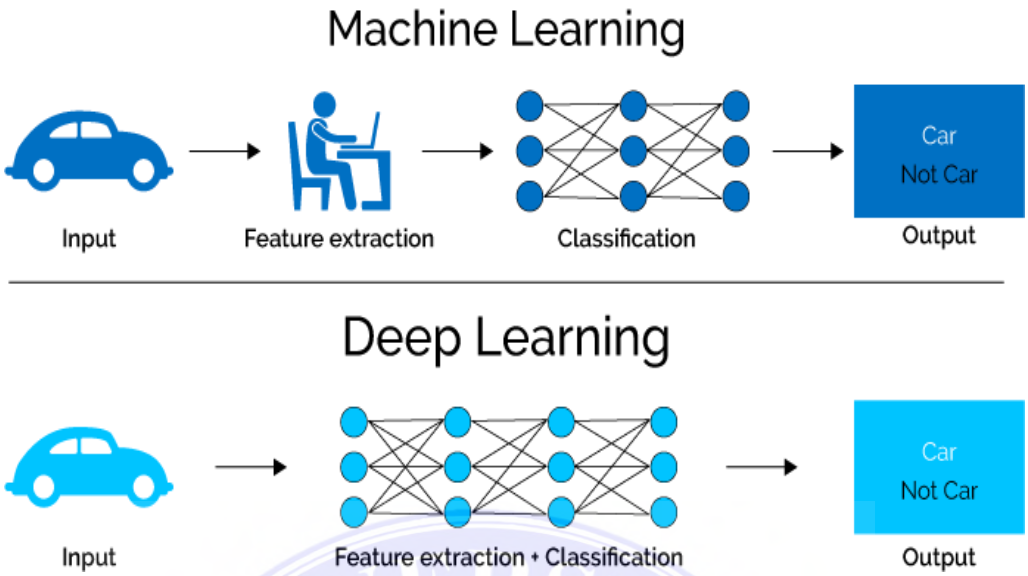
Gambar 2.2 Citra True Color

Dalam penelitian ini, jenis citra yang digunakan sebagai dataset dalam proses klasifikasi tingkat daun teh siap panen daun teh adalah dengan menggunakan citra warna RGB (*True Color*) dengan format *.jpg/jpeg*. Dalam aplikasi di bidang *data science*, khususnya *Machine Learning* dengan menggunakan *Convolutional Neural Network* (CNN), pengolahan citra seringkali digunakan untuk augmentasi citra dengan tujuan meningkatkan variasi citra yang digunakan dalam pelatihan model. Beberapa operasi augmentasi citra yang umum dilakukan meliputi:

1. *Cropping*, adalah proses pengolahan gambar yang melibatkan penghapusan sebagian dari gambar.
2. *Flipping*, adalah tindakan mengubah orientasi gambar, entah itu secara horizontal, vertikal, atau keduanya.
3. Rotasi, adalah langkah memutar gambar sejumlah derajat tertentu terhadap titik pusatnya, baik searah dengan arah jarum jam atau berlawanan arah jarum jam.
4. *Resizing*, adalah prosedur yang bertujuan untuk mengubah resolusi gambar agar sesuai dengan keinginan.

2.3 Deep Learning

Deep Learning (DL) atau pembelajaran kedalaman merupakan sub-bidang dari *Machine Learning* (ML) yang menggunakan jaringan saraf tiruan (*Artificial Neural Network*/ANN) dan biasanya memiliki jumlah *layer* yang sangat banyak. Proses belajar di dalamnya dilakukan dengan cara mengidentifikasi pola yang tersembunyi pada data dan mengklasifikasikannya untuk menghasilkan *output* yang diinginkan ketika mendapat input baru (Allaam dan Wibowo, 2021). Adapun gambaran perbedaan antara ML dengan DL dapat dijelaskan pada Gambar 2.3.



Gambar 2.3 Ilustrasi Perbedaan Machine Learning Dengan Machine Learning
Sumber : Allaam dan Wibowo, 2021

Perbedaan utama antara *Machine Learning* (ML) dan *Deep Learning* (DL) terletak pada penggunaan *feature extractor*. Dalam ML, *feature extractor* perlu dibuat secara manual yang memakan waktu dan tenaga yang cukup banyak. Sementara itu, DL secara otomatis dapat mengekstrak fitur-fitur untuk melakukan klasifikasi. Namun, DL memerlukan jumlah data yang besar untuk melatih algoritma. Beberapa aplikasi DL termasuk dalam menghasilkan suara ke dalam video dan klasifikasi citra (Allaam dan Wibowo, 2021).

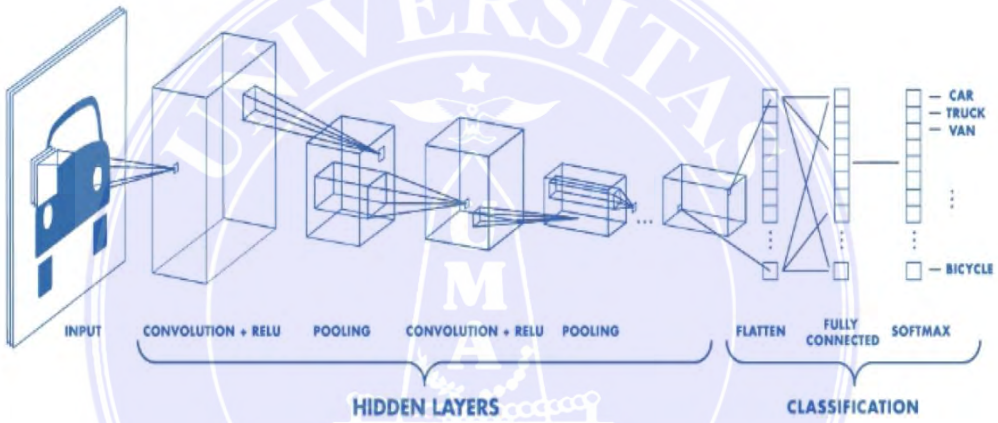
2.4 Convolutional Neural Network

Metode yang digunakan untuk mendeteksi dan mengenali objek pada sebuah gambar dapat dilakukan dengan berbagai cara, salah satunya adalah dengan menggunakan *Convolutional Neural Network* (CNN) yang sering digunakan khususnya pada data image atau citra. CNN merupakan salah satu algoritma DL (Allaam dan Wibowo, 2021) yang dirancang khusus untuk melakukan pengolahan data dua dimensi (Asrianda dkk., 2021), misalnya data gambar atau suara.

CNN digunakan sebagai metode *supervised learning* untuk mengklasifikasikan data berlabel, dengan tujuan mengelompokan data kedalam kategori yang sudah ada. Penggunaan CNN umumnya untuk mengenali benda

atau pemandangan, melakukan deteksi, segmentasi objek dan klasifikasi citra. (Allaam dan Wibowo, 2021). Dalam klasifikasi citra menggunakan CNN, terdapat dua tahap utama yaitu *feature extractor* dan *classification*/*fully connected* (ANN). Pada tahap *feature extractor*, citra diolah menjadi sekumpulan fitur numerik, yang nantinya akan digunakan sebagai *input* pada tahap *classification*/*fully connected* (ANN) untuk klasifikasi citra. Fitur yang dihasilkan masih dalam bentuk *array* multidimensi, sehingga perlu di-*flatten* menjadi *vector* satu dimensi sebelum digunakan pada tahap berikutnya. (Allaam dan Wibowo, 2021).

Adapun gambaran dalam tahapan proses klasifikasi citra dengan menggunakan metode CNN dapat dijelaskan pada Gambar 2.4.



Gambar 2.4 Tahapan Klasifikasi Citra di CNN (Arsitektur CNN)
Sumber : Allaam dan Wibowo, 2021

Pemanfaatan teknologi *Deep Learning* melalui metode CNN dapat memudahkan proses klasifikasi citra untuk menentukan tingkat daun teh siap panen daun teh. Metode CNN terbukti menjadi pilihan yang paling optimal untuk masalah klasifikasi citra karena kemampuan ekstraksi fitur citra yang dilakukannya secara otomatis. Hal ini dapat membantu menghemat waktu dan tenaga dalam proses klasifikasi citra.

Cara kerja arsitektur CNN berdasarkan Gambar 2.11 terdiri dari beberapa lapis (*layers*), dimulai dari *Input Layer*, *Convolutional Layer*, Fungsi Aktivasi *ReLU*, *Pooling Layer*, *Flatten Layer*, *Fully Connected Layer* (ANN), Fungsi Aktivasi *Softmax* dan *Dropout*. Masing-masing dari *layer* tersebut memiliki

perannya tersendiri dalam klasifikasi citra. Adapun penjelasan lebih rinci mengenai cara kerja CNN (Allaam dan Wibowo, 2021) dapat diuraikan yaitu sebagai berikut:

1. *Input Layer*

   Input layer pada CNN merepresentasikan gambar yang dimasukkan ke dalam model. Jika gambar yang dimasukkan memiliki ukuran 224x224 piksel dan menggunakan format RGB (*Red*, *Green*, *Blue*), maka gambar *input* tersebut akan berupa array multidimensi dengan ukuran 224x224x3 (dengan 3 merepresentasikan jumlah channel warna RGB).

2. *Convolutional Layer*

   Convolutional Layer adalah komponen utama dalam arsitektur CNN (Asrianda et al., 2021) yang menerima *input* gambar secara langsung. Operasi pada *layer* ini dilakukan melalui konvolusi, yaitu kombinasi *linear* dari *filter* yang diterapkan pada area lokal dari gambar masukan. *Convolutional Layer* merupakan *basis* atau dasar dari CNN, dimana terdiri dari kernel atau *filter* yang pada awalnya memiliki bobot acak. *Filter* ini akan mengalami perubahan bobot saat dilakukan pelatihan model. *Filter* tersebut akan dioperasikan dengan melakukan operasi konvolusi pada seluruh bagian citra, dengan cara digeser dan dikalikan pada bagian citra dari kiri atas ke kanan bawah. Tujuan dari *convolutional layer* adalah untuk mengekstraksi fitur pada citra *input*, seperti tepi, sudut, dan tekstur. Hasil keluaran dari *convolutional layer* disebut sebagai *feature map*. (Allaam dan Wibowo, 2021). Secara umum operasi konvolusi dapat ditulis dengan menggunakan rumus pada persamaan (2.1).

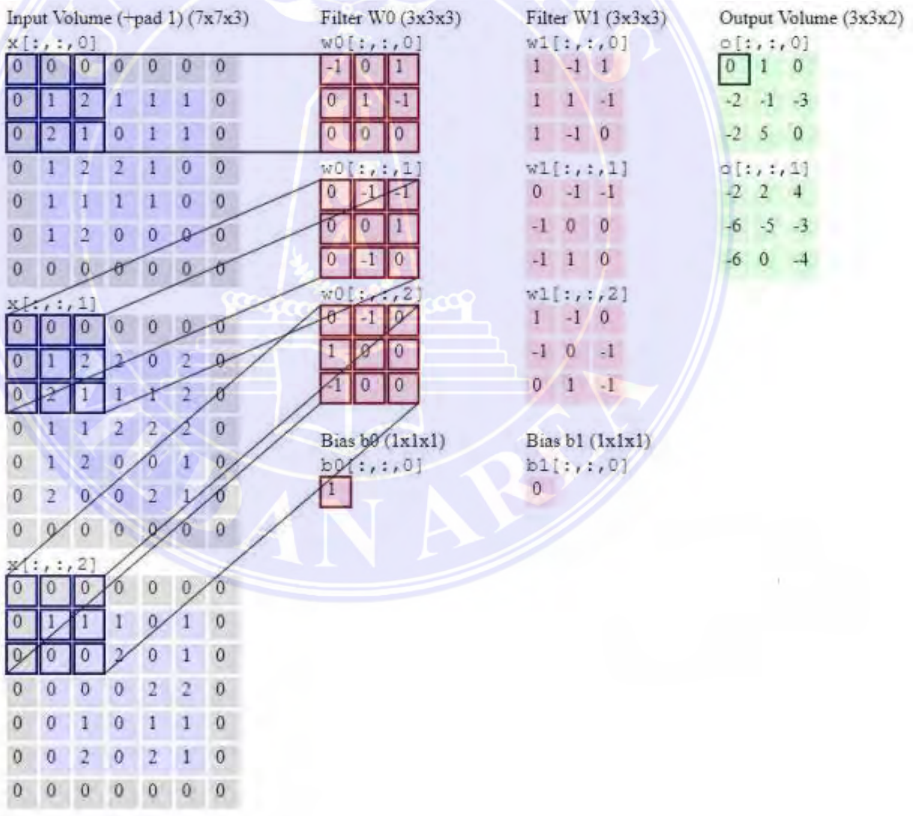
   $$s_t = (x * w)_t \quad \text{.....(2.1)}$$

   Fungsi s_t menghasilkan satu *output*, yaitu *feature map*, dengan dua argumen *input*. Argumen pertama adalah *input*, yang dalam hal ini merupakan citra, dan argumen kedua adalah kernel atau *filter* yang digunakan dalam operasi konvolusi. Dalam konteks citra dua dimensi, parameter t dapat dianggap sebagai piksel dan digantikan dengan i

dan j. Oleh karena itu, untuk melakukan operasi konvolusi pada citra dengan lebih dari satu dimensi, dapat digunakan rumus pada persamaan (2.2) berikut.

$$S_{(i,j)} = (K * I)_{(i,j)} = \sum_{i=1}^{m}\sum_{j=1}^{n}(I_{(i-m,j-n)} * K)_{(m,n)} \ldots\ldots\ldots\ldots\ldots (2.2)$$

Persamaan 2.2 merupakan rumus untuk menghitung operasi konvolusi dengan i dan j sebagai piksel dari citra. Rumus ini bersifat komutatif dan hanya terjadi saat kernel K dan *input* I dapat dibalik relatif terhadap satu sama lain. Konvolusi dapat dipandang sebagai perkalian matriks antara *input* citra dan *filter* dengan hasil *output* yang diperoleh melalui *dot product*. Adapun ilustrasi proses konvolusi dapat dijelaskan pada Gambar 2.5.



Gambar 2.5 Ilustrasi Convolutional Layer

Sumber : (Darwis, 2022)

Gambar 2.5 menunjukkan ilustrasi dari proses konvolusi pada citra yang direpresentasikan sebagari *array* dua dimensi. Pada gambar tersebut, citra berukuran 7x7 dikonvolusi dengan kernel berukuran

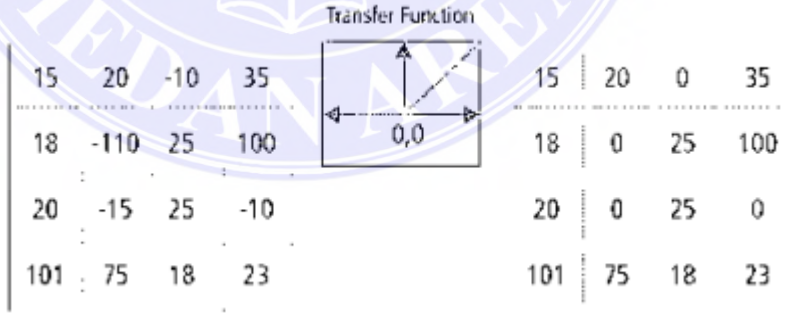
3x3. Hasil konvolusi dari citra tersebut adalah citra baru yang berukuran 3x3. Elemen pada citra hasil konvolusi dihitung dengan cara melakukan perkalian antara bobot kernel dengan nilai citra yang bersangkutan, dan jumlah dari perkalian tersebut dijadikan elemen pada citra hasil konvolusi. Untuk menjaga informasi tepi pada citra *input*, seringkali dilakukan penambahan *zero padding* pada sisi citra *input*.

3. Fungsi Aktivasi *ReLU*

Fungsi aktivasi *ReLU* (*Rectified Linear Unit*) digunakan pada tahap ekstraksi fitur setelah proses konvolusi dan sebelum proses *pooling*, serta pada setiap *neuron* di lapisan tersembunyi (*hidden layer*) pada tahap terhubung penuh (*fully connected*) (Allaam dan Wibowo, 2021). Fungsi *ReLU* adalah jenis fungsi aktivasi yang menghasilkan nilai *output neuron* yang sama dengan nilai *input*-nya jika *input* positif, dan 0 jika *input* negatif. Dengan kata lain, jika nilai *input* negatif, maka *output neuron* akan menjadi 0, sedangkan jika nilai *input* positif, maka *output neuron* akan sama dengan nilai *input*-nya (Kholik, 2021). *ReLU* merupakan lapisan aktivasi menggunakan persamaan (2.3).

$$f(x) = \max(0, x) \quad \ldots\ldots (2.3)$$

Adapun contoh operasi *ReLU* dapat dilihat pada Gambar 2.6.



Gambar 2.6 Ilustrasi Proses ReLU

Sumber : Asrianda dkk., 2021

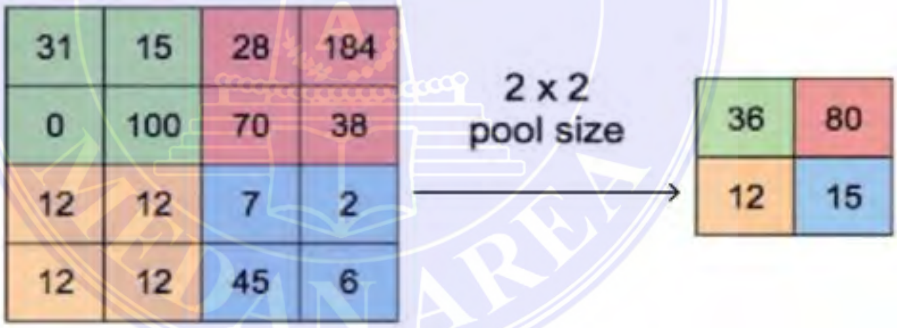
4. *Pooling Layer*

Pooling layer adalah *filter* dengan ukuran tertentu yang digerakkan pada seluruh area peta fitur. Jenis pooling yang sering digunakan adalah *Max Pooling* (mengambil fitur paling penting) dan *Average*

Pooling. Tujuan dari *layer pooling* adalah untuk mengurangi dimensi atau menurunkan resolusi peta fitur (*downsampling*), sehingga mengurangi jumlah bobot yang harus diperbarui pada saat pelatihan dan mempercepat komputasi (Allaam dan Wibowo, 2021). *Pooling* atau *subsampling* bertujuan untuk mengurangi ukuran matriks dengan membaginya menjadi beberapa *grid* kecil dan mengambil nilai maksimum dari setiap *grid.* Dengan cara ini, ukuran matriks dapat dikurangi dan informasi yang relevan tetap dapat dipertahankan. Persamaan (2.4) merupakan operasi *pooling*.

$$y = max_{i,j}^{h,w} X_{i,j} \quad \ldots\ldots (2.4)$$

Pooling layer beroperasi secara independen terhadap setiap kedalaman dari *input* dan melakukan proses *resize* secara spasial menggunakan operasi *max*. Bentuk umum dari *pooling layer* adalah *filter* berukuran $2x2$ yang diaplikasikan dengan *stride* berukuran 2 yang mengambil sampel pada setiap kedalaman *input*. Ilustrasi *pooling layer* dengan operasi *average pooling* dapat dilihat pada Gambar 2.7.



Gambar 2.7 Ilustrasi Proses Average Pooling

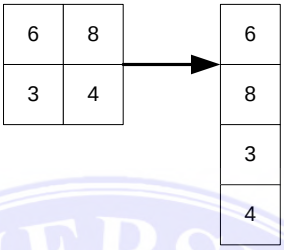
Sumber : (Darwis, 2022)

Pada Gambar 2.7, *input* berupa citra berukuran $4x4$, keluaran dari proses *pooling* adalah matriks $2x2$ dengan dimensi yang lebih kecil dari citra awal.

5. *Flatten Layer*

   Flatten Layer merupakan langkah yang dilakukan sebelum masuk ke *Fully Connected Layer*. *Feature map* dari tahap *feature extractor* masih berbentuk *array* multidimensi, sehingga perlu dilakukan proses

flatten untuk mengubah ulang fitur (*reshapre feature map*) menjadi vektor satu dimensi agar dapat digunakan sebagai *input* untuk tahap klasifikasi atau *fully connected*. Dengan cara ini, informasi pada *feature map* tetap terjaga, tetapi dalam format yang sesuai untuk diproses oleh lapisan terhubung penuh (*fully connected*). (Allaam dan Wibowo, 2021). Gambar 2.8 merupakan contoh dari *Flatten Layer*.



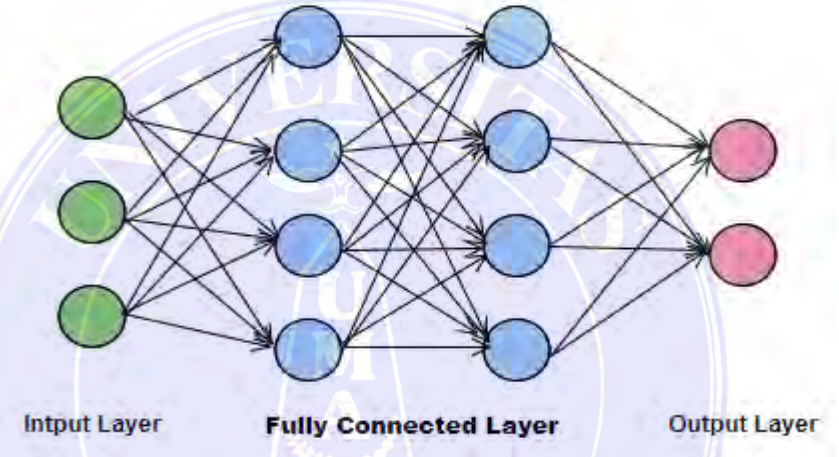
Gambar 2.8 Ilustrasi Proses Flatten Layer

Gambar 2.8 merupakan hasil akhir dari *pooling* akan dibuat menjadi 1 vektor saja, dimana hasil dari *flattening* selanjutnya akan digunakan sebagai *input* pada *Fully Connected Layer*. *Flatten* merupakan proses yang berguna untuk mengonversi *feature map* yang berbentuk matriks *multidimensional* menjadi vektor, agar dapat digunakan sebagai *input* pada *Fully Connected Layer* (Juliansyah dan Laksito, 2021).

6. *Fully Connected Layer*

   Fully Connected Layer terdiri dari *Input Layer*, *Hidden Layer*, dan *Output Layer*, dan strukturnya mirip dengan jaringan saraf tiruan (ANN). Oleh karena itu, *Fully Connected Layer* sering disebut juga sebagai ANN. Pada *Fully Connected Layer*, semua *node* dari lapisan sebelumnya terhubung dengan semua *node* di lapisan berikutnya, sehingga informasi dapat diproses secara menyeluruh (Allaam dan Wibowo, 2021). *Layer* ini menerima *input* dari *output* pada proses sebelumnya untuk mendapatkan fitur yang paling berkolerasi dengan *class* tertentu. Salah satu fungsi utama dari *layer* ini adalah menggabungkan semua *node* yang sebelumnya terorganisir dalam matriks dua dimensi menjadi vektor satu dimensi. Proses penggabungan ini juga dikenal sebagai *flatten* (Asrianda dkk., 2021).

Fungsi aktivasi *ReLU* umum digunakan disetiap *hidden layer*, termasuk *output layer*. Untuk kasus klasifikasi dengan lebih dari 2 label/kelas/kategori, *Softmax* merupakan fungsi aktivasi yang umum digunakan pada *output layer*. *Fully Connected Layer* bertujuan untuk memproses data sehingga dapat dilakukan klasifikasi, dan *output* yang dihasilkan dari *Fully Connected Layer* adalah probabilitas terhadap kategori, terutama jika *Softmax* digunakan (Allaam dan Wibowo, 2021). Gambar 2.9 merupakan tampilan *Fully Connected Layer* serta *output layer*.



Gambar 2.9 Ilustrasi Proses Fully Connected Layer

Fully Connected Layer pada Gambar 2.9 berperan sebagai *hidden layer* yang merupakan lapisan yang menentukan ke *output* manakah citra akan dikelompokan. Pada Gambar 2.9 menggunakan 2 kelas sebagai *output*, sehingga menggunakan fungsi aktifasi *ReLU*.

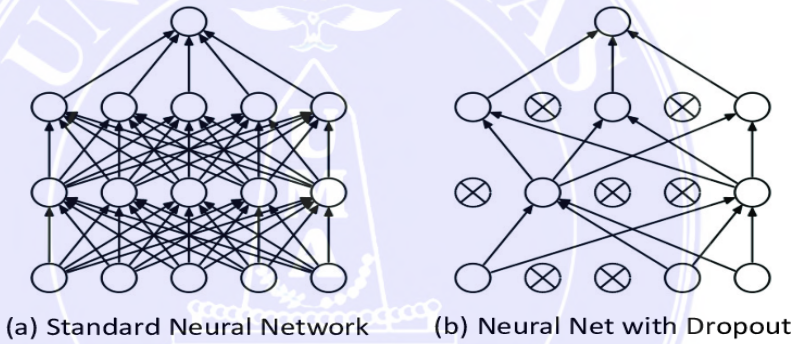
7. Fungsi Aktivasi *Softmax*

   Funsi aktivasi *softmax* digunakan pada *output layer* di tahap *fully connected* atau klasifikasi dengan lebih dari 2 kelas. Fungsi ini berguna untuk menghitung probabilitas setiap kelas target terhadap semua kelas target yang memungkinkan, sehingga dapat membantu menentukan kelas target yang tepat untuk *input* citra. Salah satu keuntungan menggunakan *Softmax* adalah *output* probabilitasnya

memiliki rentang nilai antara 0 hingga 1, dan jumlah seluruh probabilitas akan sama dengan satu.

8. *Dropout Regularization*

   Dropout adalah teknik regularisasi sebuah teknik yang digunakan untuk mencegah terjadinya *overfitting neural network* dimana beberapa *neuron* akan dipilih secara acak dan tidak dipakai selama *training* (dengan kata lain membuang fitur atau mengurangi variansi). *Neuron-neuron* ini dapat dibilang dibuang secara acak. Dengan cara ini, kontribusi dari neuron yang tidak digunakan tidak akan dihitung pada saat *forward pass* dan bobot baru juga tidak diterapkan pada saat melakukan *backpropagation*. Selain mengurangi *overfitting*, *dropout* juga dapat meningkatkan kecepatan dalam pelatihan (*training)* model.



Gambar 2.10 Ilustrasi Penggunaan Dropout

Sumber : Allaam dan Wibowo, 2021

2.5 Overfitting dan Hyperparameter

Underfitting dan *overfitting model* adalah hal yang bisa terjadi ketika membuat *model* CNN. Model yang *underfitting* atau *overfitting* tidak akan melakukan prediksi dengan benar. Berikut penjelasan mengenai *underfitting* dan *overfitting* (Allaam dan Wibowo, 2021), yaitu sebagai berikut:

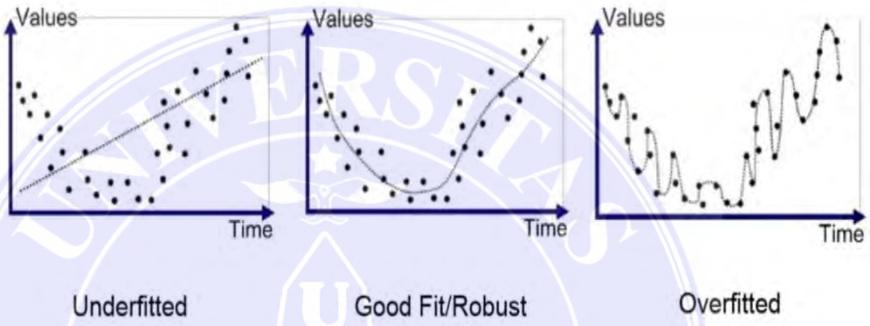
1. *Underfitting*

   Model akan mengalami *underfitting* apabila tidak mampu memahami pola data yang dihadapi, sehingga tidak dapat melakukan prediksi yang akurat baik pada dataset *training* maupun *testing*. Hal ini

tercermin dalam nilai *loss* yang tinggi dan akurasi yang rendah pada model yang *underfitting*.

2. *Overfitting*

   Overfitting terjadi ketika model terlalu difokuskan pada *dataset* training tertentu sehingga tidak dapat melakukan prediksi yang akurat pada dataset testing. *Overfitting* cenderung menangkap *noise* data yang tidak relevan, sehingga mengabaikan informasi yang seharusnya lebih penting. Model yang mengalami *overfitting* biasanya memiliki nilai *loss* yang rendah tetapi akurasinya juga rendah



Gambar 2.11 Underfitting, Good Fitting dan Overfitting

Sumber : Allaam dan Wibowo, 2021

Model yang baik adalah model yang mampu menjelaskan data secara akurat tanpa terpengaruh oleh data *noise*. Model yang baik akan memiliki *loss* rendah dan akurasi tinggi, dan mampu mengidentifikasi *trend* atau kelompok data dengan baik. Namun, dalam penggunaannya di CNN, seringkali terjadi *overfitting* sehingga diperlukan *fine-tuning* atau penyesuaian model. Salah satu cara penyesuaian yang dapat dilakukan adalah dengan menyesuaikan *hyperparameter,* yaitu parameter yang digunakan untuk mengontrol proses pelatihan dan perlu ditentukan oleh perancang model CNN. Dengan menyesuaikan *hyperparameter,* hasil akurasi prediksi dapat ditingkatkan.

Berikut beberapa *hypermarameter* yang bisa disesuaikan oleh perancang *model* CNN, yaitu sebagai berikut:

1. *Input Shape*

   Ukuran citra yang terlalu kecil dapat menyebabkan model CNN tidak dapat mempelajari fitur-fitur di dalam citra dengan baik. Oleh karena

itu, ukuran citra yang digunakan sebagai input perlu diperhatikan. Beberapa arsitektur CNN yang terkenal menyarankan penggunaan ukuran citra (*shape*) sebesar 224x224x3 piksel (3 melambangkan *channel* warna RGB).

2. *Batch Size*

   Pelatihan (*training*) data keseluruhan pada dataset yang besar, terutama pada data citra, akan menjadi sangat berat secara komputasi. Oleh karena itu, penggunaan *batch size* dapat membantu memecah *dataset* menjadi bagian-bagian kecil untuk mempercepat proses pelatihan. Sebagai contoh, jika jumlah *dataset* adalah 3.200 dan menggunakan *batch size* 64, maka akan ada 50 iterasi yang dilakukan dalam satu putaran pelatihan (50x64=3.200). Terdapat beberapa nilai yang sering digunakan sebagai *batch size*, seperti 16, 32, 64, dan 128.

3. *Epoch*

   Satu *epoch* merupakan satu putaran penuh pelatihan (*training*) dimana seluruh *dataset* telah digunakan. Tidak ada jawaban pasti terkait jumlah *epoch* yang optimal karena bergantung pada *dataset* yang digunakan. Namun, sebagai panduan, bisa menghentikan pelatihan saat nilai *loss* sudah *converage* (tidak turun lagi) atau ketika akurasi sudah tidak meningkat signifikan lagi.

4. *Optimizer*

   Optimizer merupakan algoritma atau metode yang berfungsi untuk melakukan *update* pada bobot dengan tujuan menurunkan nilai *loss* dan meningkatkan *score* akurasi. Salah satu *optimizer* yang paling sering digunakan dan dianggap efektif oleh para peneliti adalah *Adam Optimizer*, karena dianggap cepat dalam mencapai nilai *loss minimum (converge).*

5. *Learning Rate*

   Parameter yang mengatur kecepatan *model* mempelajari masalah pada saat pelatihan (*training*). Jika *learning rate* terlalu rendah, *model* memerlukan waktu yang lama untuk mencapai titik optimal. Namun, jika terlalu tinggi, *model* mungkin melewatkan titik optimal tersebut.

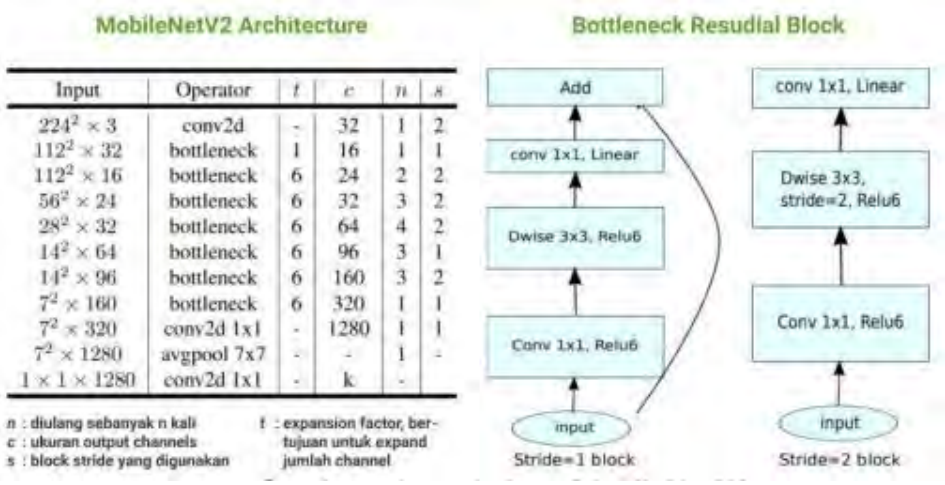
Terdapat beberapa nilai *learning rate* umum yang digunakan, seperti 0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1, dan 0.3.

2.6 Arsitektur MobileNetV2

MobileNet dikembangkan oleh *Google*, menggunakan layer yang disebut *depthwise separable convolution*. Layer ini terdiri dari *depthwise convolution* dan *pointwise convolution*, dan dirancang untuk mengurangi komputasi agar *model* yang dihasilkan lebih ringkas dengan jumlah parameter yang lebih sedikit. (Allaam dan Wibowo, 2021).

Dalam *normal convolution*, *filter* melakukan konvolusi dengan *input image* dan menghasilkan *feature map* sebagai *output image*. Sementara itu, *depthwise convolution* menggunakan jumlah *filter* yang sama dengan jumlah *channe*l pada *input image*. Setiap *filter* akan melakukan konvolusi dengan masing-masing channel pada *input image*, dan menghasilkan *feature map* dengan jumlah *channel* yang sama dengan jumlah *filter*. Agar *output image* dari *depthwise convolution* sama dengan *output image* dari *normal convolution,* maka dilakukan *pointwise convolution*. Pada tahap ini, *output* dari *depthwise convolution* dikonvolusi dengan *filter* 1x1 yang memiliki kedalaman sama dengan jumlah channel pada *output image* sebelumnya. Tujuannya adalah menghasilkan *output image* dengan kedalaman yang sama seperti *output* dari *normal convolution.*

MobileNetV2 merupakan pengembangan dari *MobileNet* yang memperkenalkan fitur baru yaitu *bottleneck*. *Bottleneck* merupakan sebuah blok dalam arsitektur yang menggunakan *depthwise separable convolution*.Adapun arsitektur dari *MobileNetV2* dapat dilihat pada Gambar 2.12.



Gambar 2.12 Arsitektur MobileNetV2

Sumber : Allaam dan Wibowo, 2021

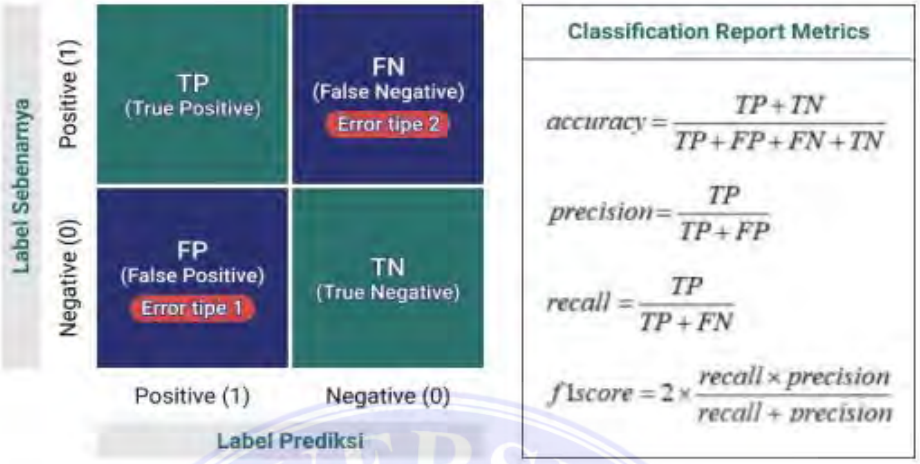
Arsitektur *MobileNetV2* terdiri dari dua jenis blok, yaitu blok *residual* dengan *stride* 1 dan dengan *stride* 2, seperti yang ditunjukkan pada Gambar 2.12. Kedua jenis blok ini digabungkan untuk membentuk arsitektur *MobileNetV2*. Menurut *Keras library*, arsitektur *MobileNetV2* terdiri dari 155 *layer* yang dibagi ke dalam 16 blok.

Dalam penelitian ini, arsitektur CNN *MobileNetV2* digunakan untuk mengklasifikasi tingkat daun teh siap panen daun teh. Keuntungan dari menggunakan arsitektur *MobileNetV2* adalah akurasi yang tinggi dan jumlah parameter pelatihan yang lebih sedikit dibandingkan dengan arsitektur CNN lainnya, sehingga komputasi yang dibutuhkan lebih ringan. Selain itu, *size model MobileNetV2* juga kecil namun tetap memberikan performa yang baik. Oleh karena itu, jika model tersebut akan di *deploy* dalam sebuah aplikasi seperti *website*, ukuran model yang ringan dan performa yang baik akan sangat bermanfaat.

2.7 Confusion Matrix dan Classification Report

Dalam *Machine Learning* dan *Deep Learning*, klasifikasi merupakan salah satu metode *Supervised Learning*. Evaluasi performa merupakan tahap penting dalam *life cycle model Machine Learning* dan *Deep Learning*. Dalam evaluasi

performa model klasifikasi, terdapat dua teknik yang umum digunakan yaitu *Confusion Matrix* dan *Classification Report.* (Allaam dan Wibowo, 2021).



Gambar 2.13 Confusion Matrix dan Rumus Metrics di Classification Report
Sumber : Allaam dan Wibowo, 2021

Berikut penjelasan lebih detail antara *Confusion Matrix* dan *Classification Report* pada Gambar 2.13, yaitu sebagai berikut:

1. *Confusion Matrix*

   Confusion matrix adalah tabel berukuran NxN (dengan N adalah jumlah kelas/label/kategori) yang memuat informasi tentang jumlah prediksi yang tepat atau salah dari suatu *model* klasifikasi, yang berguna untuk membandingkan nilai aktual dengan nilai prediksi. Setiap baris dalam matriks mewakili kelas yang sebenarnya, sedangkan setiap kolom mewakili kelas yang diprediksi. Confusion matrix menghasilkan empat jenis nilai yang terdiri dari

   a. *True Positive* (TP): Prediksi positif & nilai sebenarnya positif.
   b. *True Negative* (TN): Prediksi negatif & nilai sebenarnya negatif.
   c. *False Positive* (FP): Prediksi positif & nilai sebenarnya negatif.
   d. *False Negative* (FN): Prediksi negatif & nilai sebenarnya positif.

2. *Classification Report*

   Meskipun *confusion matrix* memberikan informasi yang detail tentang kinerja suatu model dalam melakukan klasifikasi, tetapi masih sulit bagi pengguna untuk memahami seberapa baik kinerja tersebut. Oleh

karena itu, *confusion matrix* dapat digunakan untuk menghitung *metrics* yang dapat mengukur kinerja model, yang kemudian disebut sebagai c*lassification report* menggunakan beberapa *metrics*, yaitu:

a. *Accuracy m*enggambarkan seberapa akurat *model* dalam mengklasifikasikan dengan benar.
b. *Precision m*enggambarkan akurasi antara data yang diminta dengan hasil prediksi *model*.
c. *Recall m*enggambarkan keberhasilan *model* dalam menemukan kembali sebuah informasi.
d. *F-1 score* menggambarkan perbandingan rata-rata precision dan recall yang dibobotkan.

2.8 Tools Pendukung

Dalam implementasi proyek menggunakan metode CNN, diperlukan bahasa pemrograman yang stabil, fleksibel, dan dilengkapi dengan sekumpulan *tools* seperti pustaka (*library*). Hal ini akan memudahkan dan mempercepat proses implementas. Saat ini, *Python* adalah bahasa pemrograman yang memenuhi kriteria tersebut. Untuk mengimplementasikan metode CNN, terdapat beberapa pustaka *Python* yang sering digunakan, seperti *Tensorflow, Keras, Sklearn, Numpy, Pandas, dan Matplotlib.*

2.8.1 Python

Python adalah bahasa pemrograman yang populer dan sering digunakan oleh berbagai kalangan, seperti akademisi dan masyarakat umum, untuk berbagai tujuan, seperti membuat aplikasi *desktop, website, game*, melakukan analisis, serta membangun proyek kecerdasan buatan, termasuk metode CNN. Ada banyak *library* dan. *library-library* ini memiliki fitur yang memadai untuk mempermudah proses implementasi dan pengembangan proyek CNN (Allaam dan Wibowo, 2021), yaitu sebagai berikut:

1. *Tensorflow*

   Tensorflow adalah *platform open source end-to-end* yang dibuat dan dikembangkan oleh *Google Brain* untuk mendukung *Machine*

Learning. *Tensorflow* memiliki alat, pustaka, dan sumber daya yang komprehensif dan fleksibel yang memungkinkan penelitian menggunakan *Machine Learning* yang mutakhir dan *develope*r dengan mudah membangun dan menerapkan aplikasi yang didukung *Machine Learning*. *Tensorflow* dapat digunakan dengan bahasa pemrograman *Python* yang stabil. *Tensorflow* menawarkan pemrograman yang melakukan berbagai macam tugas ML/DL (regresi dan klasifikasi).

2. *Keras*

   Keras adalah sebuah API (*Application Programming Interface*) tingkat tinggi dari *Tensorflow* yang dibuat oleh François Chollet, seorang *engineer* di *Google*. *Keras* dirancang agar mudah dipelajari dan digunakan, terutama bagi para peneliti, sehingga mereka dapat fokus pada eksperimen dan bukan pada penggunaan *library. Keras* dapat digunakan untuk membangun CNN, seperti melakukan *one-hot-encoding* pada label citra, membuat arsitektur CNN, menggunakan arsitektur CNN populer yang tersedia, melakukan *plotting model* CNN, memuat *dataset* citra ke dalam arsitektur CNN, serta melakukan *training* dan *testing* model CNN untuk menghasilkan evaluasi seperti akurasi dan *loss*. Selain itu, Keras juga dapat digunakan untuk menyimpan dan mengakses model CNN dalam format *h5*.

3. *Sklearn*

   Sebuah *tools* untuk *Machine Learning* yang digunakan untuk berbagai model statistik seperti *classification, regression, clustering*, dan *dimensionality reduction*. Dalam implementasi CNN, *Sklearn* dapat digunakan sebagai alat untuk menerapkan teknik *K-Fold Cross Validation* pada *dataset*, membuat *Confusion Matrix*, dan *Classification Report.*

4. *Numpy*

   Sebuah *library* yang berfungsi untuk memanipulasi dan mengolah *array* serta matriks. Salah satu contoh penggunaannya dalam CNN adalah dengan melakukan *reshape* pada dimensi citra, menghitung

rata-rata dan standar deviasi akurasi model CNN, dan melakukan pengolahan data.

5. *Pandas*

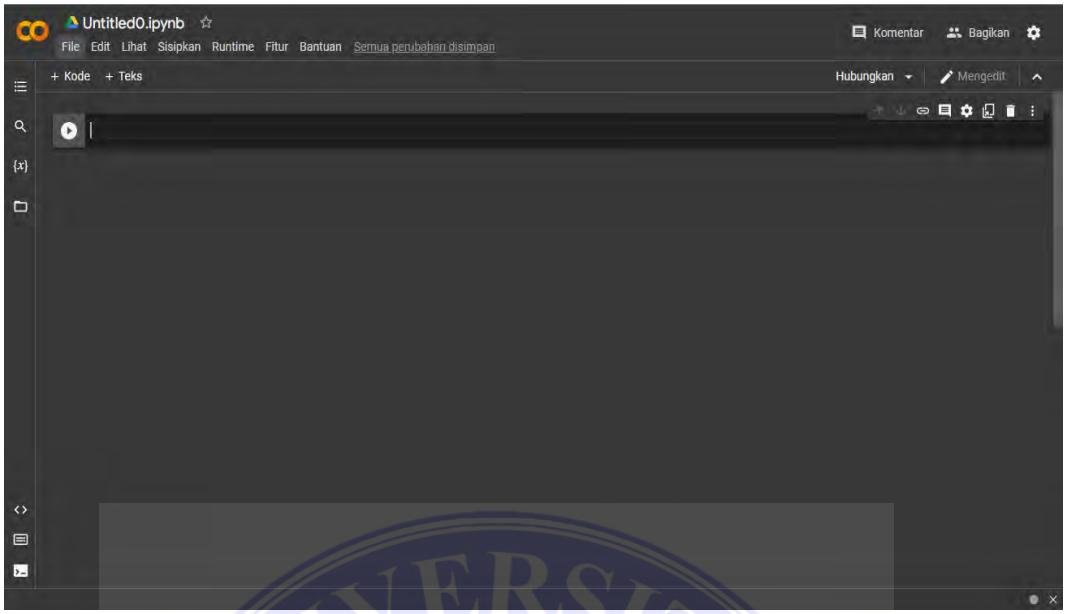
   Berguna untuk melakukan manipulasi dan analisis data. Dalam konteks penggunaan pada CNN, contohnya adalah menyimpan riwayat pelatihan model (dalam bentuk akurasi dan *loss*) ke dalam sebuah tabel CSV.

6. *Matplotlib*

   Digunakan untuk membuat grafik dan visualisasi data. Contoh penggunaannya pada CNN adalah memplot *dataset* beserta labelnya dan memplot hasil pelatihan dan pengujian model CNN ke dalam grafik untuk memudahkan pemahaman dan analisis.

2.8.2 Google Colaboratory

Google Colaboratory atau *Google Colab* adalah sebuah *platform* gratis berbasis *cloud* yang menyediakan lingkungan pengembangan *Jupyter Notebook* (*Python*). *Colab* memungkinkan pengguna untuk melatih dan menguji model *Machine Learning* dan *Deep Learning* di CPU, GPU, dan TPU tanpa memerlukan konfigurasi apapun. Ada 3 keunggulan utama Colab yaitu tidak memerlukan konfigurasi, akses gratis ke GPU, dan mudah dibagikan dengan rekan penelitian. *Colab* hanya memerlukan akun *Google dan internet browser*, serta menyediakan waktu penggunaan *notebook* maksimal 12 jam/hari dengan RAM 12GB dan disk 100GB untuk opsi gratis/tanpa berbayar. (Allaam dan Wibowo, 2021). Adapun tampilan dari *Google Colab* dapat dilihat pada Gambar 2.14.



Gambar 2.14 Tampilan Google Colab

Gambar 2.14 menunjukkan bahwa tampilan *Google Colab* mirip dengan *Jupyter Notebook*. Salah satu keuntungan penggunaan *Google Colab* adalah tidak perlu melakukan konfigurasi karena menggunakan teknologi *cloud computing*, memberikan akses gratis untuk menggunakan mesin dengan kecepatan tinggi seperti GPU, dan mudah terhubung dengan *Google Drive*. Dalam penelitian ini, *Google Colab* digunakan untuk melatih dan menguji model *Convolutional Neural Network* dengan arsitektur *MobileNetV2* dalam melakukan klasifikasi tingkat daun teh siap panen daun teh.

2.9 Penelitian Terdahulu

Dalam penelitian ini, penulis mengambil inspirasi dan merujuk pada beberapa penelitian sebelumnya yang erat kaitannya dengan latar belakang masalah. Referensi dari penelitian sebelumnya sangatlah penting untuk menghindari plagiat atau pengulangan penelitian yang sudah ada sebelumnya. Tujuan dari pengambilan referensi ini juga untuk memberikan kontribusi pada penelitian terkait agar dapat terus berkembang.

Berikut beberapa ulasan tentang penelitian-penelitian terdahulu yang pernah dilakukan sebelumnya berkenaan dengan data dan metode yang digunakan seperti disajikan pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No.	Nama Peneliti dan Tahun	Judul Penelitian	Hasil Penelitian
1.	Ari Ashari Jaelani, Fiky Yosef Supratman, Nur Ibrahim, 2020	Perancangan Aplikasi Untuk Klasifikasi Daun Teh Seri Gambung (GMB) Menggunakan Algoritma *Convolutional Neural Network*	Metode CNN dengan arsitektur *Mobilenet* menghasilkan akurasi sebesar 60% dalam melakukan klasifikasi pada daun teh sari gambung
2.	Abdul Hafiez Suherman, Nur Ibrahim, Heri Syahrian, Vitria Puspitasari Rahadi, Muhammad Khais Prayoga, 2021	Klasifikasi Daun Teh Gambung Varietas *Assamica* Menggunakan *Convolutional Neural Network* Dengan Arsitektur *Lenet-5*	Metode CNN dengan arsitektur *LeNet-5* menghasilkan akurasi sebesar 94.55% dalam melakukan klasifikasi pada daun teh gambung varietas *assamica*
3.	Baga Skara Aji Wicaksono, 2019	Identifikasi Tingkat Kematangan Daun Teh Menggunakan *Centroid Clustering* Berdasarkan Ruang Warna YCbCr	Klasifikasi tingkat kematangan daun teh menggunakan metode *Centroid Clustering* dengan jumlah centroid 10 titik berdasarkan ruang warna YCbCr dan ciri statistik minimum, maksimum, dan variansi mendapatkan nilai akurasi sebesar 80%
4.	Rahma Nur Auliasari,	Identifikasi Kematangan Daun Teh	Metode ektraksi ciri seperti *Hue Saturation Intensity*

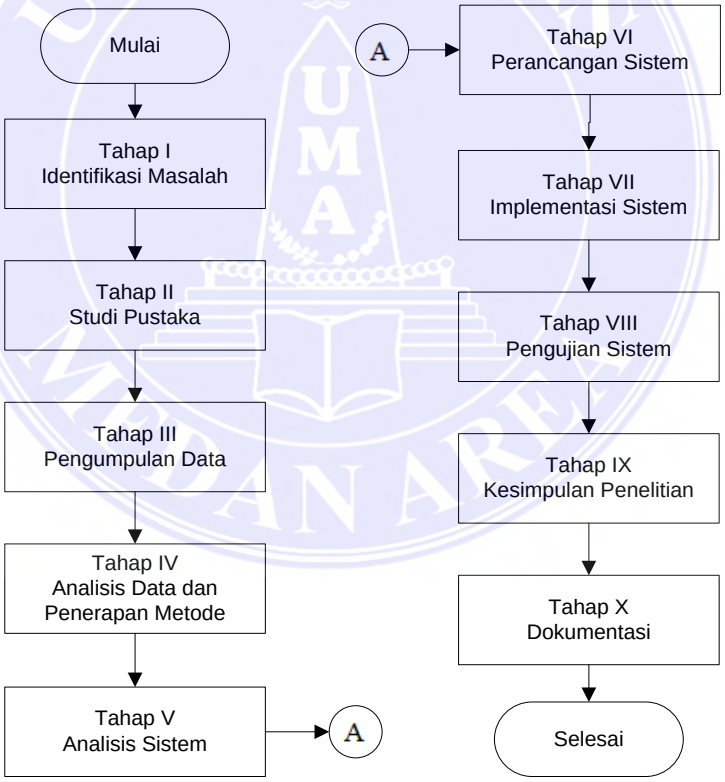
	Ledya Novamizanti, Nur Ibrahim, 2020	Berbasis Fitur Warna *Hue Saturation Intensity* (HSI) dan *Hue Saturation Value* (HSV)	(HSI) dan *Hue Saturation Value* (HSV) menghasilkan tingkat akurasi 100% pada fitur warna HIS dan 83,33% pada fitur warna HSV dalam identifikasi daun teh siap panen daun the
5.	Nur Ibrahim, Gita Ayu Lestari, Faniesa Saufana Hanafi, Khaerudin Saleh, NorKumalasari Caecar Pratiwi, Muthia Syafika Haq, Adhi Irianto Mastur, 2022	Klasifikasi Tingkat Kematangan Pucuk Daun Teh Menggunakan Metode *Convolutional Neural Network*	Membandingkan arsitektur *VGGNet19* dan arsitektur *ResNet50*. Hasil penelitian menjelaskan bahwa arsitektur *VGGNet19* memperoleh hasil yang lebih baik dengan nilai akurasi sebesar 97.5%

BAB III
METODOLOGI PENELITIAN

3.1 Kerangka Kerja Penelitian

Dalam mendukung jalannya penelitian ini agar lebih terarah dan sistematis maka dibutuhkan suatu tahapan desain penelitian atau kerangka kerja penelitian yang disusun dengan terstruktur. Kerangka kerja merupakan tahapan yang dilakukan dalam penelitian, yang mana tahapan ini bertujuan untuk memberikan ketentuan bentuk masalah dan tujuan sebagai pedoman dari tahap awal penelitian hingga selesai yang bertujuan agar tahapan-tahapan yang dilakukan berjalan secara terstruktur dan sesuai dengan tujuan yang ingin dicapai.

Secara garis besar tahapan-tahapan yang dilakukan dalam penelitian ini dapat ditampilkan pada Gambar 3.1.



Gambar 3.1 Kerangka Kerja Penelitian

Berdasarkan kerangka kerja penelitian pada Gambar 3.1, maka dapat diuraikan pembahasan pada masing-masing tahapan dalam penelitian ini yaitu:

1. Identifkasi Masalah

Identifikasi masalah merupakan tahap awal dan persiapan dari sebuah penelitian. Pada tahap ini dilakukan identifikasi yang menjadi pokok permasalahan dan mengambil permasalahan tersebut menjadi topik penelitian sehingga dapat dicari solusi yang nantinya akan menjadi tujuan dari penelitian ini. Permasalahan yang diangkat dalam penelitian ini terkait dalam hal klasifikasi daun teh siap panen. Oleh karena itu diperlukan ada pendekatan digital agar dapat mengenali daun teh siap panen dengan cepat dan akurat. Pada penelitian ini akan menggunakan metode CNN dengan arsitektur *MobileNetV2* dalam melakukan klasifikasi daun teh siap panen sebagai alternatif solusi dari permasalahan tersebut.

2. Studi Pustaka

   Studi pustaka bertujuan untuk mendapatkan landasan teori mengenai permasalahan yang akan diteliti sehingga dapat memahami permasalahan dengan benar dan sesuai dengan pembahasan yang dilakukan. Pada tahap ini dilakukan pencarian dan mengumpulkan sumber literatur pada penelitian ini dengan cara *browsing* di *internet*, membaca berbagai literatur seperti buku dan jurnal, hasil kajian dari penelitian terdahulu, serta sumber-sumber lain yang relevan dengan pokok permasalahan pada penelitian ini.

3. Pengumpulan Data

   Data yang digunakan penulis dalam penelitian ini menggunakan data primer yang didapatkan dari PTPN IV Sidamanik. Data yang digunakan adalah gambar daun teh verietas *Assamica klon Gambung* (GMB) 7 yang diambil menggunakan kamera *smartphone*. Data gambar daun teh yang dikumpulkan sebanyak 2000 buah gambar dengan format *.jpg/jpeg* dengan rincian 1000 buah gambar daun teh peko dan 1000 buah gambar daun teh kepel.

4. Analisis Data dan Penerapan Metode

   Setelah data yang dikumpulkan telah mencukupi maka tahap selanjutnya dilakukan analisis data. Analisis yang diperlukan yaitu dengan melakukan praproses pada kumpulan data gambar daun teh

(*dataset*). Setiap gambar daun teh peko dan daun teh kepel akan diseleksi guna mendapatkan kualitas yang sesuai, karena *dataset* ini nantinya yang akan menjadi *input* dalam proses *training model* pada metode CNN dengan arsitektur *MobileNetV2*.

5. Analisis Sistem

   Analisis sistem adalah fase awal sebelum merancang sebuah sistem dengan melakukan proses identifikasi komponen-komponen atau instrumen yang dibutuhkan dalam penelitian ini. Beberapa tahapan yang dapat dilakukan yaitu terkait kebutuhan perangkat keras (*hardware*) dan kebutuhan perangkat lunak (*software*). Alat yang dibutuhkan adalah sebuah laptop dengan spesifikasi kebutuhan perangkat keras yaitu:

   a. *Processor : Intel(R) Core(TM) i3-6006U*
   b. *CPU @ 2.00GHz 1.99 GHz*
   c. *RAM : 4.00 GB*
   d. *Hardisk : 1 TB*

   Sedangkan spesifikasi kebutuhan perangkat lunak yang digunakan dalam penelitian ini, yaitu:

   a. *Microsoft Windows 10* tipe *64-bit* sebagai sistem operasi
   b. *Python* versi 3 sebagai bahasa pemrograman
   c. *Google Colaboratory* sebagai *tools* untuk membangun, melatih dan menguji *model*.
   d. *Tensorflow* dan *Keras* sebagai *library Machine Learning*.
   e. *Sumblime* sebagai teks editor untuk *coding* program

6. Perancangan Sistem

   Perancangan sistem dilakukan guna mendapatkan gambaran dengan jelas tentang apa yang dikerjakan pada analisis sistem dan dilanjutkan dengan mempertimbangkan bagaimana membentuk sistem tersebut. Perancangan sistem dilakukan dengan membuat tampilan *interface* (antarmuka) sistem yang akan di integrasikan dengan aplikasi pada tahap implementasi sistem.

7. Implementasi Sistem

Pada tahap ini akan dilakukan pengkodean (*coding*) yang mengacu pada perancangan sistem untuk menerapakan model dari metode CNN dengan arsitektur *MobileNetV2* menggunakan bahasa pemrograman *Python*. Dalam tahapan ini akan dilakukan terlebih dahulu proses *training model* arsitektur *MobileNetV2* dengan menggunakan bantuan *Google Colaboratory*. Proses *training* dilakukan untuk mendapatkan akurasi *model* terbaik yang nantinya digunakan dalam klasifikasi tingkat daun teh siap panen. Setelah didapatkan *model* dengan akurasi yang baik, maka selanjutnya dilakukan *deploy* kedalamm sebuah aplikasi berbaiss *web*.

8. Pengujian Sistem

   Tahap pengujian dilakukan dengan tujuan untuk menjamin aplikasi yang dibuat sesuai dengan hasil analisis dan perancangan serta menghasilkan satu kesimpulan apakah aplikasi tersebut sesuai dengan yang diharapkan. Pengujian sistem dilakukan pada data *testing* untuk mengethaui tingkat akurasi sistem dalam melakukan klasifikasi daun teh siap panen.

9. Kesimpulan Penelitian

   Pada tahap ini diambil kesimpulan yang menjawab tujuan akhir dari penelitian berdasarkan hasil analisis data sampai pengujian sistem yang telah dilakukan.

10. Dokumentasi

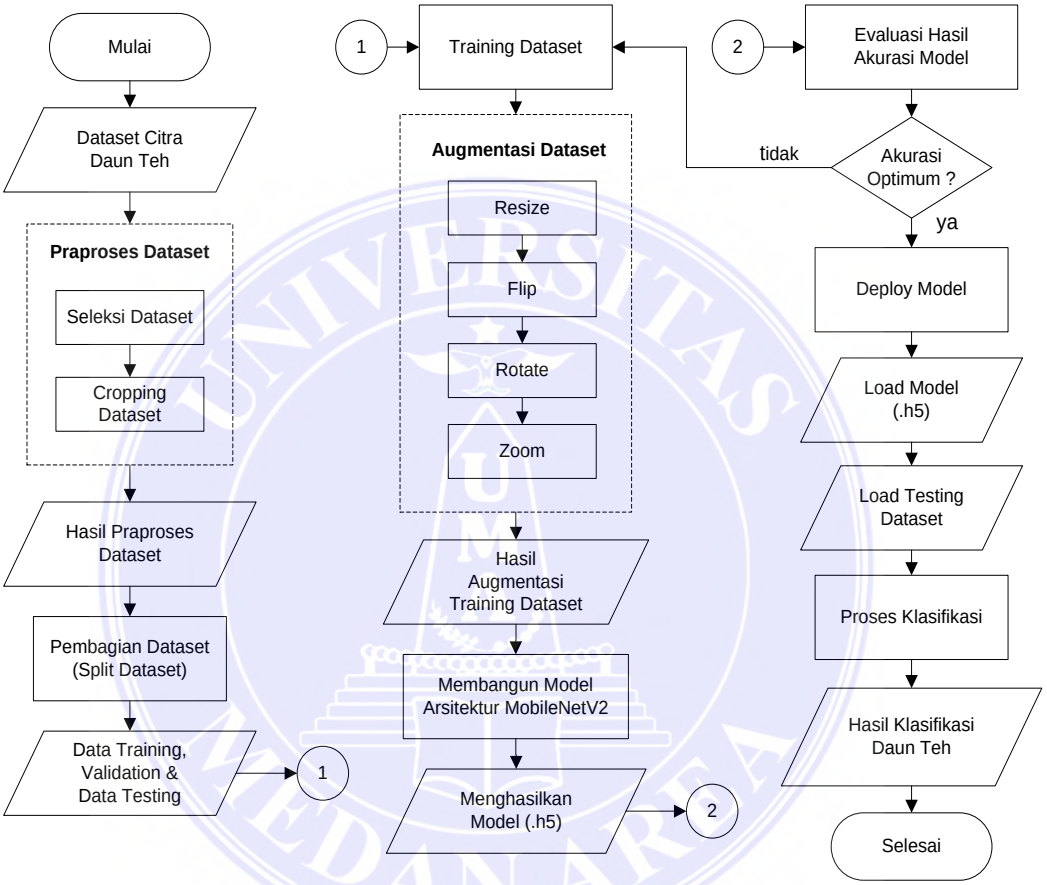
    Tahapan dokumentasi merupakan sebuah tahapan dari awal hingga akhir yang bertujuan untuk mendokumentasikan seluruh kegiatan dan hasil dari penelitian ini yang selanjutnya dibuat dalam bentuk laporan penelitian atau skripsi.

3.2 Analisis Proses Klasifikasi

Pada penelitian ini metode CNN dengan arsitektur *MobileNetV2* akan di implementasikan untuk mengklasifikasi daun teh siap panen. Metode CNN dengan arsitektur *MobileNetV2* dipilih karena merupakan salah satu algoritma *Deep Learning* yang memiliki tingkat akurasi yang cukup tinggi. Oleh karena itu

dengan menggunakan metode ini, dapat membangun sistem yang memiliki tingkat kecerdasan untuk melakukan klasifikasi daun teh siap panen dengan akurasi tinggi layaknya kemampuan manusia.

Secara garis besar tahapan yang dilakukan dalam mengklasifikasi daun teh siap panen daun teh dalam penelitian ini dapat ditampilkan dalam bentuk diagram alir (*flowchart*) seperti pada Gambar 3.2.



Gambar 3.2 Flowchart Sistem Secara Umum

Secara keseluruhan, proses klasifikasi pada Gambar 3.2 melalui beberapa tahapan. Pertama, dilakukan pengumpulan *dataset* citra daun teh (daun peko dan daun kepel). Kedua, akan dilakukan praproses *dataset* (seleksi dan *cropping*) agar data citra sesuai dengan kriteria standar saat memasuki arsitektur *MobileNetV2*. Tahap selanjutnya yang ke tiga, *dataset* akan dibagi menjadi data *training*, data *validation* dan data *testing*, dimana data *training* nantinya akan digunakan untuk melatih *model* dan data *testing* nantinya akan digunakan untuk menguji *model* arsitektur *MobileNetV2*. Ke empat, akan dilakukan perancangan arsitektur *model*

CNN dengan menggunakan data *training*, yang mana dalam penelitian ini akan menggunakan arsitektur *MobileNetV2* untuk menghasilkan *model* dengan akurasi yang tinggi. Tahap selanjutnya, yang ke lima, *model* yang dihasilkan akan diuji menggunakan data *testing* untuk mengevaluasi hasilnya lalu memilih *model* terbaik dengan akurasi yang paling tinggi, dan tahap yang terakhir adalah men-*deploy model* kedalam aplikasi siap pakai. Pada tahap *deploy* dilakukan pengembangan aplikasi berbasis *web*, dengan menggunakan *model* dengan hasil performa terbaik dalam proses *training* sebelumnya sebagai *engine* klasifikasi daun teh siap panen pada aplikasi.

3.2.1 Persiapan Dataset

Dataset yang digunakan pada penelitian ini adalah berupa sampel citra dun teh yang didapatkan dari PTPN IV Sidamanik. Data yang digunakan adalah gambar daun teh verietas *Assamica klon Gambung* (GMB) 7 yang diambil menggunakan kamera *smartphone*. *Dataset* terbagi menjadi dua kelas, yaitu gambar daun teh peko dan gambar daun teh kepel.

Dataset terbagi menjadi 2 folder sesuai dengan jumlah kelas (daun teh siap panen) dengan total masing-masing adalah 1000 citra dengan kelas daun teh peko dan 1000 citra dengan kelas dau teh kepel. Total keseluruhan *dataset* adalah 2000 buah dan disimpan dalam format *.jpg/.jpeg*. Berikut merupakan visualisasi dari sampel citra daun teh untuk kelas daun teh peko seperti ditampilkan pada Gambar 3.3.

Gambar 3.3 Sampel Dataset Daun Teh Peko

Berikut merupakan visualisasi dari sampel citra daun teh untuk kelas daun teh kepel seperti ditampilkan pada Gambar 3.4.

Gambar 3.4 Sampel Dataset Daun Teh Kepel

Selanjutnya dataset yang telah disiapkan, nantinya akan digunakan sebagai masukan (*input*) pada sistem yang dibagi menjadi dua. Pertama yaitu masukan yang akan digunakan sebagai data *training* dan yang kedua masukan sebagai data *testing*. Data *training* merupakan kumpulan *dataset* yang digunakan untuk melatih *model* terlebih dahulu. Sedangkan data *testing* adalah kumpulan *dataset* yang digunakan untuk menguji *model* sebagai tahap akhir guna mengetahui seberapa tinggi tingkat akurasi *model* yang diperoleh.

3.2.2 Praproses Dataset

Dataset yang telah disiapkan selanjutnya akan dilakukan praproses (*preprocessing*). Praproses *dataset* ini dilakukan dengan harapan menghasilkan *model* dengan akurasi yang tinggi. Adapun tahapan yang dilakukan pada bagian praproses (*preprocessing*) yaitu dapat dijabarkan sebagai berikut:

1. *Dataset* yang telah dikumpulkan akan diseleksi terlebih dahulu, untuk citra atau gambar yang berkualitas rendah dan terdapat *noise* akan dihapus. Tahapan ini dilakukan untuk menghindari atau meminimalisir gangguan pada saat proses pembelajran *model* CNN nantinya, sehingga diharapkan akan menghasilkan *model* CNN dengan akurasi yang tinggi.

(a) (b)

Gambar 3.5 Dataset Kualitas Rendah (a) Blur (b) Noise

Gambar 3.5 merupakan contoh *dataset* dengan kualitas rendah, sehinnga perlu untuk dihilangkan. Setelah dilakukan seleksi pada *dataset* maka hasilnya menjadi 2000 (dengan 1000 citra/kelas). Semua *dataset* tersimpan dalam format *.jpg/.jpeg*.

2. *Dataset* yang telah diseleksi selanjutnya akan dilakukan proses *cropping* ke bagian fitur inti dari citra daun teh dengan rasio 1x1. Proses *cropping* menjadi hal yang cukup penting, karena jika tidak dilakukan maka semua objek yang ada pada citra akan dianalisis oleh komputer, padahal fokus dari objek yang akan diteliti adalah adalah berupa fitur daun yang ada pada tanaman teh. Dalam proses *cropping* ini masih dilakukan secara manual. Adapun ilustrasi pada operasi *cropping* dapat disajikan pada Gambar 3.6.

Gambar 3.6 Proses Cropping Dataset

Sebagai langkah terakhir dari tahapan praproses *dataset*, semua citra akan dinamai ulang (*labeling*) sesuai dengan nama kelas/tingkat daun teh siap panen daun teh dengan pola sebagai berikut:

1. Untuk kelas (daun teh peko) akan dilabeli dengan nama daunpeko0001.jpg, daunpeko0002.jpg, daunpeko0003.jpg, dan seterusnya sampai daunpeko1000.jpg.
2. Untuk kelas (daun teh kepel) akan dilabeli dengan nama daunkepel0001.jpg, daunkepel0002.jpg, daunkepel0003.jpg, dan seterusnya sampai daunkepel1000.jpg.

Proses pelabelan untuk setiap citra pada dataset dalam penelitian ini dilakukan dengan menggunakan alat bantu (*tools*) bernama *Advanced Renamer*. *Tools* yang digunakan dapat di *download* secara gratis dengan mengunjungi *link https://www.advancedrenamer.com/download*.

3.2.3 Pembagian Dataset

Dalam proses klasifikasi daun teh siap panen menggunakan metode CNN dengan arsitektur *MobileNetV2* pada peneliti ini, *dataset* dibagi menjadi tiga yaitu data latih (*training*), data validasi (*validation*) dan data uji (*testing*). Data *training* digunakan untuk melatih atau membangun *model*, data *validation* digunakan untuk mengoptimasi saat melatih *model*. Sedangkan data *testing* digunakan untuk menguji (*testing*) *model* yang dihasilkan untuk dievaluasi. Pembagian *dataset* selengkapnya dapat disajikan pada Tabel 3.1.

Tabel 3.1 Pembagian Dataset

Kelas	Data Training	Data Validation	Data Testing	∑
Daun Teh Peko	700	200	100	1000
Daun Teh Kepel	700	200	100	1000
Total	**1400**	**400**	**200**	**2000**
Persentase	**70%**	**20%**	**10%**	**100%**

Berdasarkan Tabel 3.1, *dataset* dibagi menjadi tiga yaitu data *training* dengan persentase 70%, data *validation* dengan persentase 20%, dan data *testing* dengan persentase 10%.

3.2.4 Pemodelan Arsitektur MobileNetV2

Pada penelitian ini akan menggunakan metode CNN dengan arsitektur *MobileNetV2* dalam melakukan klasifikasi daun teh siap panen. *Dataset* final yang digunakan setelah dilakukan praproses adalah berjumlah 2000 buah citra dengan rincian 1000 buah citra daun teh peko dan 1000 buah citra daun teh kepel. *Dataset* terbagi menjadi tiga bagian, yaitu data *training* 70%, data *validation* 20% dan data *testing* 10%. Pembagian *dataset* ini akan mengurangi jumlah citra yang akan di *training*, sehingga perlu dilakukan augmentasi untuk memperbanyak jumlah *dataset*. Proses augmentasi ini dilakukan sebelum melatih *model* menggunakan arsitektur *MobileNetV2*.

3.2.4.1 Augmentasi Dataset

Augmentasi *dataset* dilakukan untuk meningkatkan ukuran dataset agar mendapatkan banyak gambar yang berbeda. Proses ini dilakukan untuk mencegah situasi *overfitting* (memiliki kinerja baik selama pelatihan, tetapi buruk pada data baru) dalam proses pelatihan *model* dengan arsitektur *MobileNetV2*. Augmentasi yang dilakukan yaitu pencerminan (*flipping*) secara horizontal adalah untuk membalik gambar secara horizontal, rotasi (*rotating*) dengan sudut 0.1 derajat adalah untuk memutar gambar dengan sudut 10 derajat secara acak, dan memperbesar citra (*zoom*) dengan perbesaran 0.1 adalah untuk memperbesar citra dengan perbesaran sebanyak 1+0.1 dari luas gambar.

Adapun tahapan yang dilakukan pada bagian augmentasi *dataset* yaitu dapat dijabarkan sebagai berikut:

1. Pencerminan (*Flipping*)

   Proses *flipping* mengakibatkan adanya perubahan orientasi citra, baik secara horizontal, vertikal, maupun keduanya. Pada penelitian ini menggunakan *horizontal flip* untuk membalik citra secara horizontal yang dilakukan secara acak. Adapun ilustrasi pada proses *flipping* dapat disajikan pada Gambar 3.7.

Gambar 3.7 Proses Flipping Dataset

2. Rotasi (*Rotating*)

   Proses *rotating* dilakukan untuk memutar citra dalam derajat terhadap titik pusatnya, baik searah jarum jam maupun berlawanan arah jarum jam. Pada penelitian ini, citra akan dirotasi sebesar 0.2 derajat adalah untuk memutar gambar dengan sudut 20 derajat secara acak. Adapun ilustrasi pada proses *rotating* dapat disajikan pada Gambar 3.8.

Gambar 3.8 Proses Rotating Dataset

3. Memperbesar Citra (*Zoom*)

Proses *zoom* dilakukan untuk memperbesar citra dengan perbesaran 0.1 adalah untuk memperbesar citra dengan perbesaran sebanyak 1+0.1 dari luas gambar yang dilakukan secara acak. Adapun ilustrasi pada proses *zoom* dapat disajikan pada Gambar 3.9.

Gambar 3.9 Proses Zoom Dataset

4. Mengubah Resolusi (*Resize*)

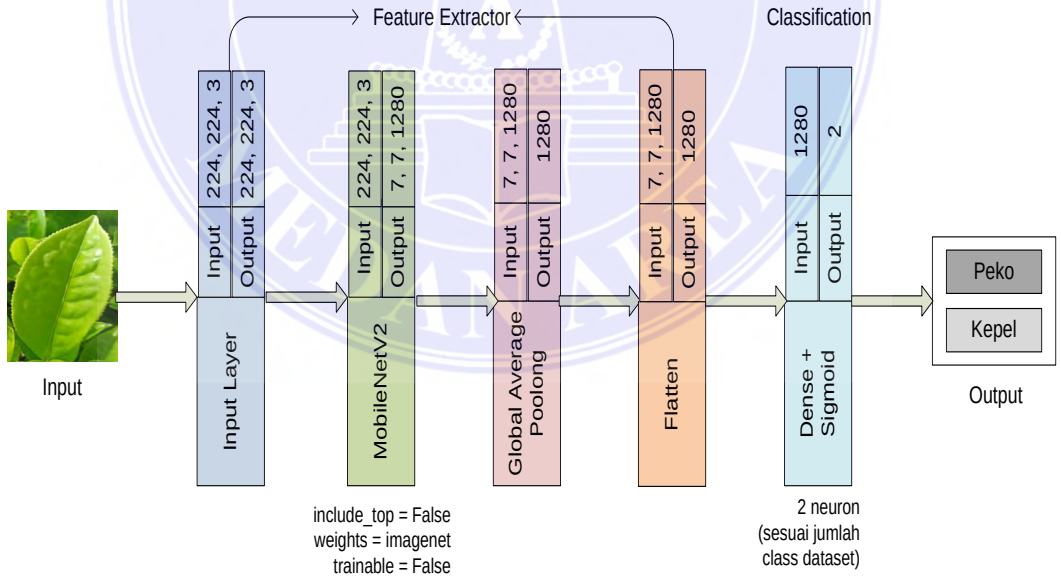
Proses *resize* dimaksudkan untuk mengubah resolusi citra sesuai dengan yang di inginkan. Pada penelitian ini, citra *dataset* akan mengalami perubahan ukuran (*resizing*) menjadi 224x224 piksel. Proses ini bertujuan untuk mempercepat proses klasifikasi dan menyamaratakan ukuran citra setiap data dan juga untuk menyesuaikan dengan *input shape* didalam arsitektur *MobileNetV2* yang akan digunakan. Adapun ilustrasi pada proses *resize* dapat disajikan pada Gambar 3.10.

Gambar 3.10 Proses Resize Dataset

Dataset yang telah diperkaya atau diperbanyak dengan cara melakukan augmentasi, maka *dataset* telah siap untuk di *training* dengan menggunakan arsitektur *MobileNetV2*.

3.2.4.2 Rancangan Arsitektur MobileNetV2

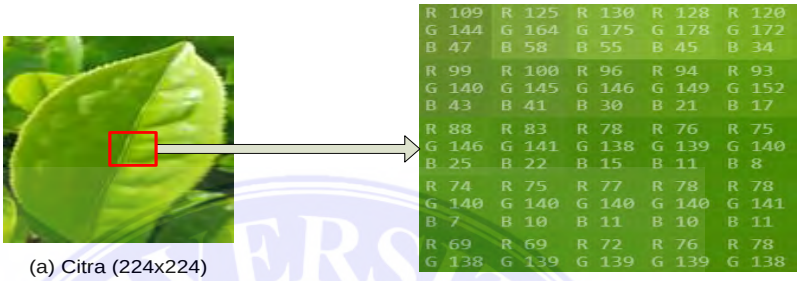
Pada penelitian ini akan dibuat *image classification model* yang dapat mengenali tingkat daun teh siap panen daun teh dengan menggunakan *pre-trained model MobileNetV2*. Arsitektur *MobileNetV2* dipilih karena memiliki arsitektur yang ringan, sehingga lebih cepat dalam proses pelatihan (*training*), dan memberikan akurasi yang baik dalam klasifikasi citra. Terdapat 2 *hyperparameter* yang akan diuji pada arsitektur *MobileNetV2*, yaitu skenario berdasarkan perubahan jumlah *epoch* dan perubahan nilai *Learning Rate*. Setiap skenario model arsitektur akan dilatih (*training*) terhadap skenario *dataset*, sehingga *output model* yang dihasilkan nanti berjumlah 6 *model*, yang selanjutnya masing-masing *model* ini akan diujikan (*testing*) terhadap data *testing* lalu mengevaluasi hasilnya, memilih *model* terbaik dan terakhir menerapkannya ke aplikasi siap pakai.



Gambar 3.11 Rancangan Arsitektur MobileNetV2

Gambar 3.11 menjelaskan proses atau tahapan dari arsitektur *MobileNetV2* yang akan digunakan pada klasifikasi daun teh siap panen. Adapun penjelasannya dapat diuraikan sebagai berikut:

1. Pada bagian *Input Layer*, citra yang akan digunakan berukuran 224x224x3. Angka 3 pada ukuran citra tersebut menandakan *channel imag*e (RGB) pada citra warna (*true color*). Sebelum citra di inputkan, maka terlebih dahulu akan diaugmentasi seperti yang sudah dijelaskan pada bagian augmentasi *dataset* di atas. Hasil augmentasi kemudian dipanggil untuk dijadikan *input* pada *model*.



(a) Citra (224x224)

(a) Pixel Citra (5x5)

Gambar 3.12 Sampel Citra Input

2. Citra yang sudah diinputkan tersebut masuk pada tahap *feature extractor* untuk mengekstraksi fitur dari citra *dataset* menggunakan *pre-trained model* dengan arsitektur *MobileNetV2*. Pada Gambar 3.11 terdapat beberapa parameter dari arsitektur *MobileNetV2* yang disesuaikan dalam penelitian ini, yaitu:
   a. *include_top=False*

      Berfungsi untuk menghilangkan *layer* terakhir (*layer output*) pada *model MobileNetV2*. *Layer output default* dari *MobileNetV2* adalah 1000 (sesuai jumlah *class MobileNetV2*). Pada penelitian ini, jumlah *class* dari *output*-nya adalah 2 (*class* daun teh peko dan *class* daun teh kepel). Oleh karena itu, menggunakan parameter *include_top=False* untuk menghilangkan *layer* terakhir pada *model MobileNetV2*.

   b. *weights='imagenet'*

      Berfungsi agar dapat menggunakan *weights model* dari *MobileNetV2* yang sudah dilatih pada *dataset imagenet*.

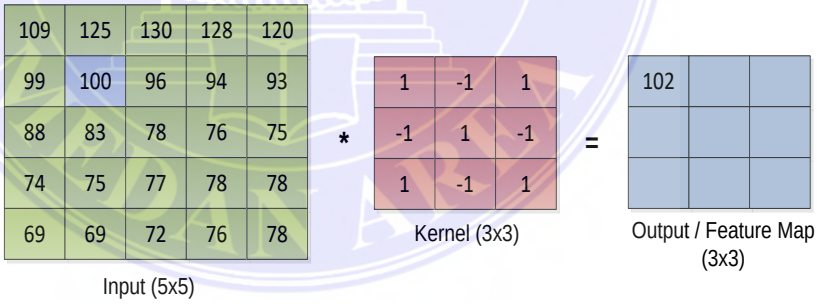
   c. *trainable=False*

      Berfungsi agar semua *layer* pada *MobileNetV2* tidak dapat di-*training* ulang. Hal ini karena hanya ingin menggunakan *model*

MobileNetV2 sebagai *feature extractor* untuk mengekstraksi fitur dari citra *dataset*, dan tidak ingin mengubah bobot-bobot (*weights*) yang sudah dilatih pada *model MobileNetV2*.

Berdasarkan pustaka *Keras*, arsitektur *MobileNetV2* terdiri dari 155 *layer* yang terbagi ke dalam 16 blok. Pada *layer* pertama akan dilakukan operasi konvolusi untuk mengekstraksi fitur dari *dataset* citra *input*. Tahapan ini adalah proses kombinasi antara dua buah matriks yang berbeda untuk menghasilkan nilai matriks yang baru. Konvolusi pertama menggunakan *input* citra berukuran 224x224,3 dengan jumlah *filter* 32 serta kernel yang digunakan berukuran 3x3. Untuk mempermudah perhitungan manual dalam proses konvolusi, maka citra *input* yang digunakan hanya mengambil piksel citra input berukuran 5x5 serta hanya menghitung hasil konvolusi pada *channel Red* (R). Adapun proses konvolusi pada *layer* pertama dengan *stride* 1 dan *zero padding* menggunakan rumus pada persamaan (2.2) yang dapat disimulasikan sebagai berikut:

a. Tempatkan kernel pada sudut kiri atas piksel citra, kemudian hitung menggunakan rumus pada persamaan (2.2).



Gambar 3.13 Hasil Konvolusi Stride ke-1

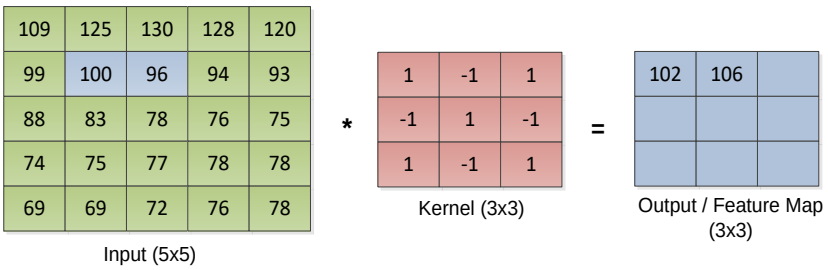
Hasil konvolusi pada *stride*/pergeseran pertama adalah 102. Nilai ini dihitung dengan cara mengalikan antara piksel citra *input* dengan nilai *kernel*, yaitu:

$$S_{(i,j)} = (K * I)_{(i,j)} = \sum_{i=1}^{m}\sum_{j=1}^{n}(I_{(i-m,j-n)} * K)_{(m,n)}$$

$$= (1 * 109) + (-1 * 125) + (1 * 130) + (-1 * 99) +$$

$$(1 * 100) + (-1 * 96) + (1 * 88) + (-1 * 83) + (1 * 78)$$
$$= 102$$

b. Geser *kernel* satu piksel ke kanan, kemudian hitung menggunakan rumus pada persamaan (2.2).

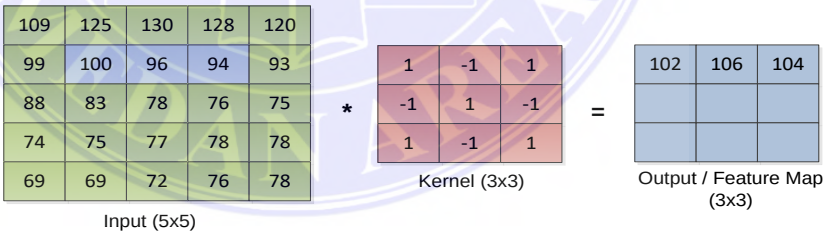


Gambar 3.14 Hasil Konvolusi Stride ke-2

Hasil konvolusi pada *stride*/pergeseran kedua adalah 106. Nilai ini dihitung dengan cara mengalikan antara citra dengan nilai *kernel*, yaitu:

$$S_{(i,j)} = (1 * 125) + (-1 * 130) + (1 * 128) + (-1 * 100) +$$
$$(1 * 96) + (-1 * 94) + (1 * 83) + (-1 * 78) + (1 * 76)$$
$$= 106$$

c. Geser *kernel* satu piksel ke kanan, kemudian hitung menggunakan rumus pada persamaan (2.2).

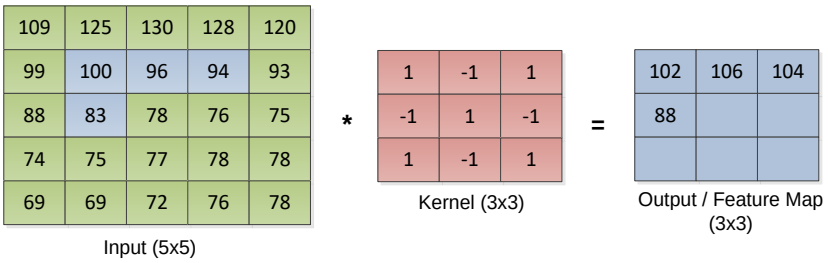


Gambar 3.15 Hasil Konvolusi Stride ke-3

Hasil konvolusi pada *stride*/pergeseran ketiga adalah 104. Nilai ini dihitung dengan cara mengalikan antara citra dengan nilai *kernel*, yaitu:

$$S_{(i,j)} = (1 * 130) + (-1 * 128) + (1 * 120) + (-1 * 96) +$$
$$(1 * 94) + (-1 * 93) + (1 * 78) + (-1 * 76) + (1 * 75)$$
$$= 104$$

d. Geser *kernel* keposisi baris kedua piksel citra, kemudian hitung menggunakan rumus pada persamaan (2.2).

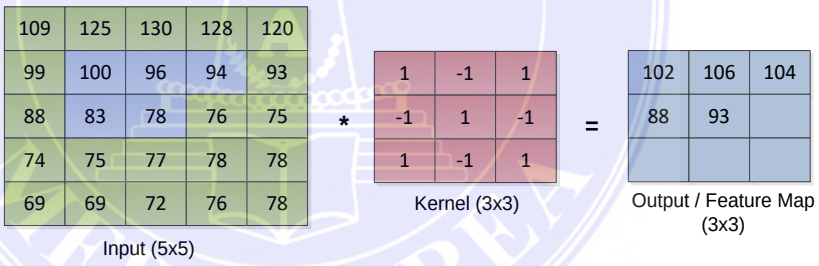


Gambar 3.16 Hasil Konvolusi Stride ke-4

Hasil konvolusi pada *stride* ke empat adalah 88. Nilai ini dihitung dengan cara mengalikan antara piksel citra dengan nilai *kernel*.

$$S_{(i,j)} = (1 * 99) + (-1 * 100) + (1 * 96) + (-1 * 88) + (1 * 83) + (-1 * 78) + (1 * 74) + (-1 * 75) + (1 * 77) = 88$$

e. Geser *kernel* satu piksel ke kanan, kemudian hitung menggunakan rumus pada persamaan (2.2).

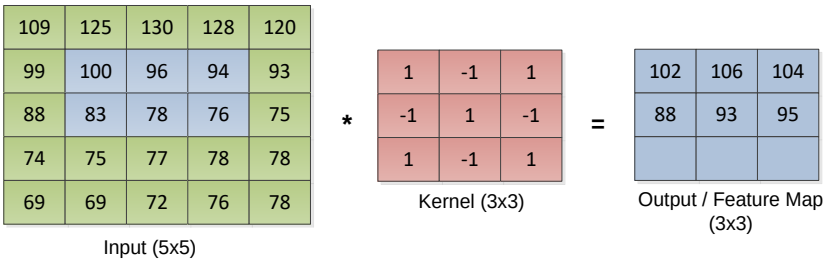


Gambar 3.17 Hasil Konvolusi Stride ke-5

Hasil konvolusi pada *stride* ke lima adalah 93. Nilai ini dihitung dengan cara mengalikan antara piksel citra dengan nilai *kernel*, yaitu:

$$S_{(i,j)} = (1 * 100) + (-1 * 96) + (1 * 94) + (-1 * 83) + (1 * 78) + (-1 * 76) + (1 * 75) + (-1 * 77) + (1 * 78) = 93$$

f. Geser *kernel* satu piksel ke kanan, kemudian hitung menggunakan rumus pada persamaan (2.2).

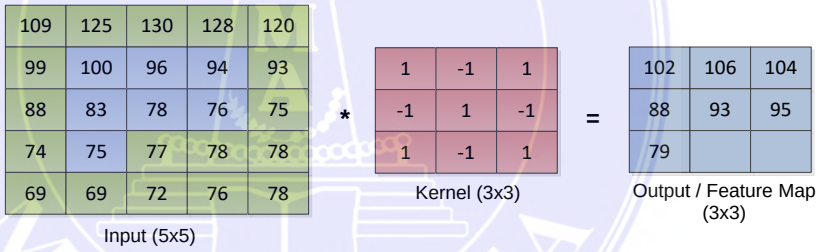


Gambar 3.18 Hasil Konvolusi Stride ke-6

Hasil konvolusi pada *stride*/pergeseran ke enam adalah 95. Nilai ini dihitung dengan cara mengalikan antara piksel citra dengan nilai *kernel*, yaitu:

$$S_{(i,j)} = (1 * 96) + (-1 * 94) + (1 * 93) + (-1 * 78) + (1 * 76) + (-1 * 75) + (1 * 77) + (-1 * 78) + (1 * 78) = 95$$

g. Geser *kernel* keposisi baris ketga piksel citra, kemudian hitung menggunakan rumus pada persamaan (2.2).

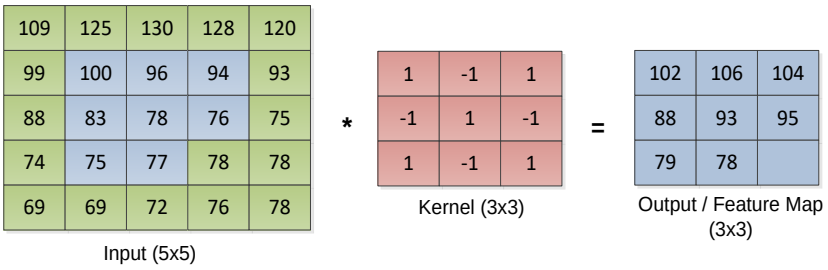


Gambar 3.19 Hasil Konvolusi Stride ke-7

Hasil konvolusi pada *stride*/pergeseran ke tujuh adalah 79. Nilai ini dihitung dengan cara mengalikan antara piksel citra dengan nilai *kernel*, yaitu:

$$S_{(i,j)} = (1 * 88) + (-1 * 83) + (1 * 78) + (-1 * 74) + (1 * 75) + (-1 * 77) + (1 * 69) + (-1 * 69) + (1 * 72) = 79$$

h. Geser *kernel* satu piksel ke kanan, kemudian hitung menggunakan rumus pada persamaan (2.2).

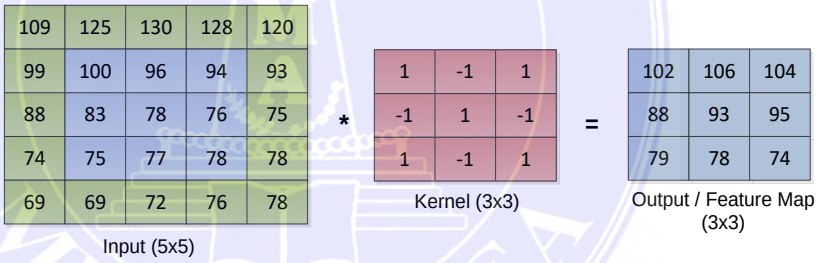


Gambar 3.20 Hasil Konvolusi Stride ke-8

Hasil konvolusi pada *stride*/pergeseran ke delapan adalah 78. Nilai ini dihitung dengan cara mengalikan antara piksel citra dengan nilai *kernel*, yaitu:

$$S_{(i,j)} = (1 * 83) + (-1 * 78) + (1 * 76) + (-1 * 75) + (1 * 77) + (-1 * 78) + (1 * 69) + (-1 * 72) + (1 * 76) = 78$$

i. Geser *kernel* satu piksel ke kanan, kemudian hitung menggunakan rumus pada persamaan (2.2).

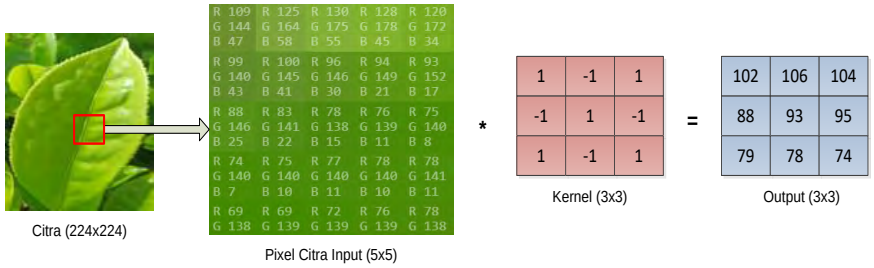


Gambar 3.21 Hasil Konvolusi Stride ke-9

Hasil konvolusi pada *stride*/pergeseran ke sembilan adalah 74. Nilai ini dihitung dengan cara mengalikan antara piksel citra dengan nilai *kernel*, yaitu:

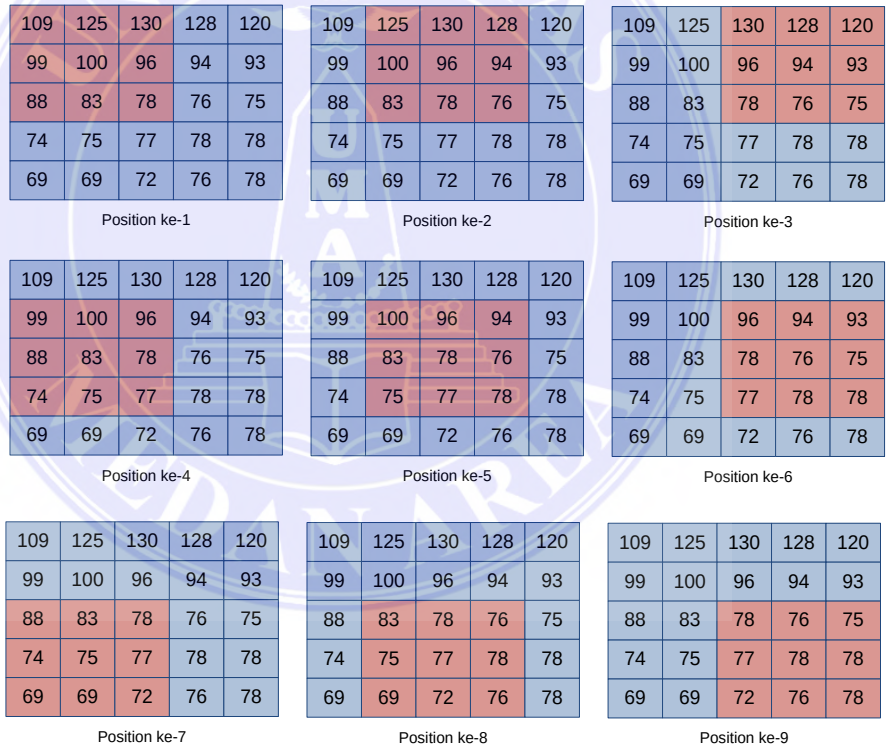
$$S_{(i,j)} = (1 * 78) + (-1 * 76) + (1 * 75) + (-1 * 77) + (1 * 78) + (-1 * 78) + (1 * 72) + (-1 * 76) + (1 * 78) = 74$$

Berdasarkan hasil perhitungan konvolusi pada citra *input* berukuran $5x5$, maka diperoleh citra *output* berukuran $3x3$ seperti ditampilkan pada gambar 3.22.



Gambar 3.22 Output Pixel Citra Hasil Konvolusi

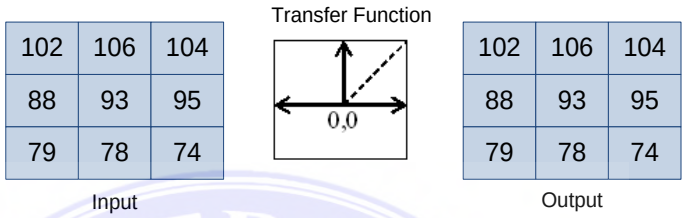
Pada Gambar 3.22, *stride* (pergeseran) yang digunakan pada kernel adalah 1 dengan ukuran 3x3. *Stride* ini memperlihatkan pergeseran jumlah kernel dalam matriks yang bernilai satu. Adapun visualisasi dari *stride* kernel pada proses konvolusi dapat ditampilkan pada Gambar 3.23.



Gambar 3.23 Posisi Kernel Dalam Proses Konvolusi Dengan Stride 1

Gambar 3.23 menunjukkan perhitungan dari piksel citra *input* dengan kernel yang terdapat dalam proses konvolusi dengan ukuran kernelnya 3x3. Adapun proses perhitungannya seperti yang sudah dijelaskan pada bagian sebelumnya.

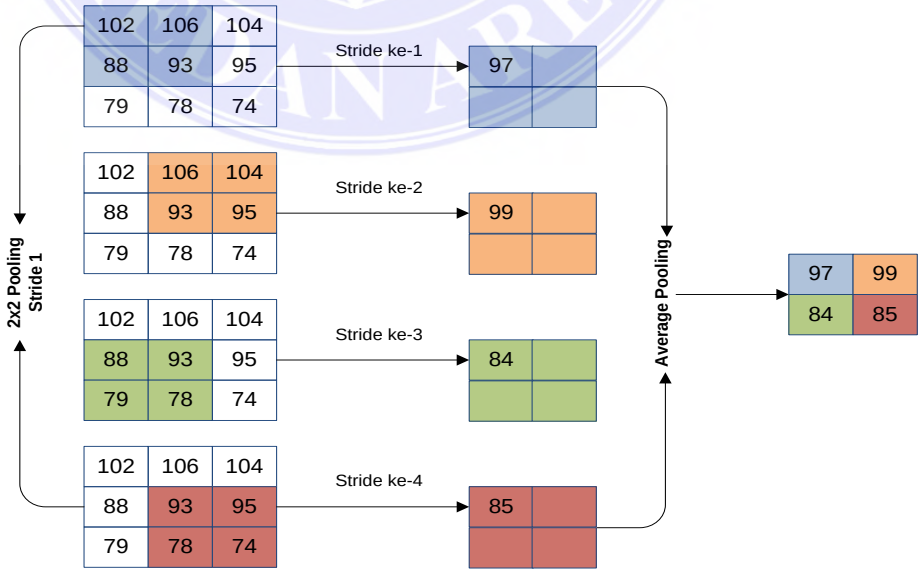
Setelah proses konvolusi selesai, maka akan dilanjutkan dengan menambahkan aktivasi fungsi *ReLU* (*Rectified Linear Unit*). Fungsi aktivasi *ReLU* dipakai setelah melakukan proses konvolusi/sebelum melakukan *pooling* (di tahap *feature extractor*). Adapun hasil operasi *ReLU* dengan menggunakan rumus pada persamaan (2.3) dapat ditampilkan pada Gambar 3.24.



Gambar 3.24 Hasil Proses ReLu

Fungsi *ReLU* adalah fungsi yang nilai *output* dari *neuron* bisa dinyatakan dengan 0 jika nilai *input* adalah negatif. Jika nilai *input* adalah positif, maka *output* dari *neuron* adalah nilai *input* aktivasi itu sendiri.

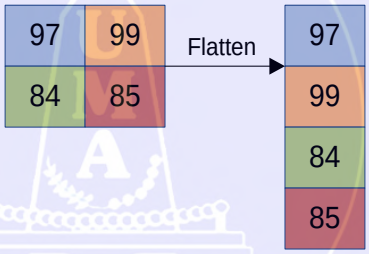
3. *Output* dari *MobileNetV2* akan diambil kemudian dilakukan proses *pooling*. Dalam proses *pooling* dengan *MobileNetV2* menggunakan *Global Average Pooling*. Adapun hasil operasi *pooling* dengan menggunakan persamaan (2.4) dapat ditampilkan pada Gambar 3.25.



Gambar 3.25 Hasil Proses Average Pooling

Ukuran *pooling* yang dipakai seperti pada Gambar 3.25 berukuran 2x2 dengan penggunaan *stride* 1, dimana pergeseran karnel pada *input* matriks berjumlah satu. Dalam proses *pooling* digunakan *average pooling*, yang nantinya akan bergeser sesuai dengan *stride* dan ukurannya, yang bertujuan untuk mengambil nilai rata-rata. Seperti pada Gambar 3.26 *output* yang dihasilkan berasal dari nilai rata-rata yang diambil dari matrik *feature map* hasil konvolusi. *Output* dari *average pooling* berukuran 2x2.

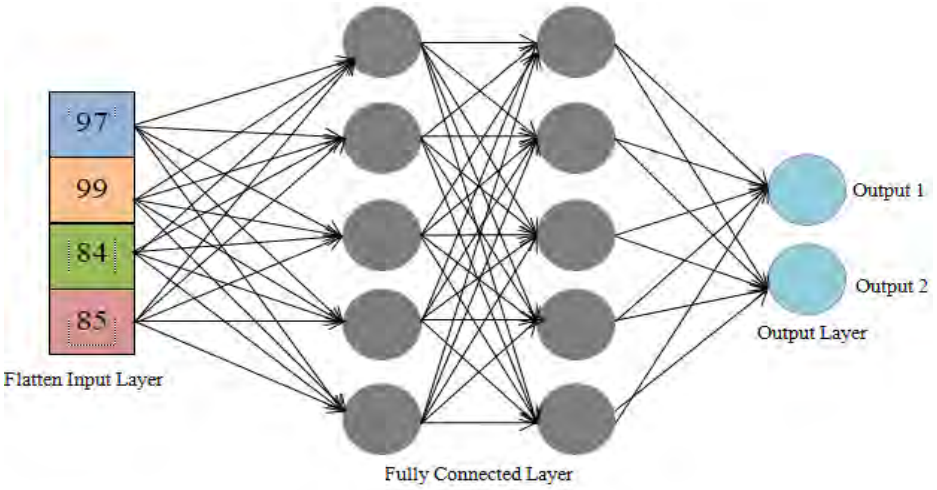
4. *Output* dari *Global Average Pooling* masih berbentuk *array* multidimensi, oleh karena itu perlu dilakukan *Flatt*en, yaitu membentuk ulang fitur (*reshape feature map*) menjadi sebuah *vector* (*array* satu dimensi) agar bisa digunakan sebagai *input* ke tahap *fully connected/classification*. Adapun hasil dari proses *flatten* dapat ditampilkan pada Gambar 3.26.



Gambar 3.26 Hasil Proses Flatten Layer

Gambar 3.26 merupakan hasil akhir dari *flatten* yang akan dibuat menjadi 1 vektor saja, dimana hasil dari *flattening* selanjutnya akan digunakan sebagai *input* pada *Fully Connected Layer*.

5. Setelah penggunaan fungsi *flatten* ditambahkan, selanjutnya akan disambungkan dengan *dense layer* berukuran 2 untuk melakukan klasifikasi pada 2 *class* yang ada pada *dataset* dalam penelitian ini, yaitu *class* daun teh peko dan *class* daun teh kepel. Fungsi ini akan menambahkan *layer* pada *fully connected* yang akan dijadikan sebagai *layer classification*. Gambar 3.27 merupakan tampilan *fully connected layer* serta *output layer*.



Gambar 3.27 Fully Connected Layer

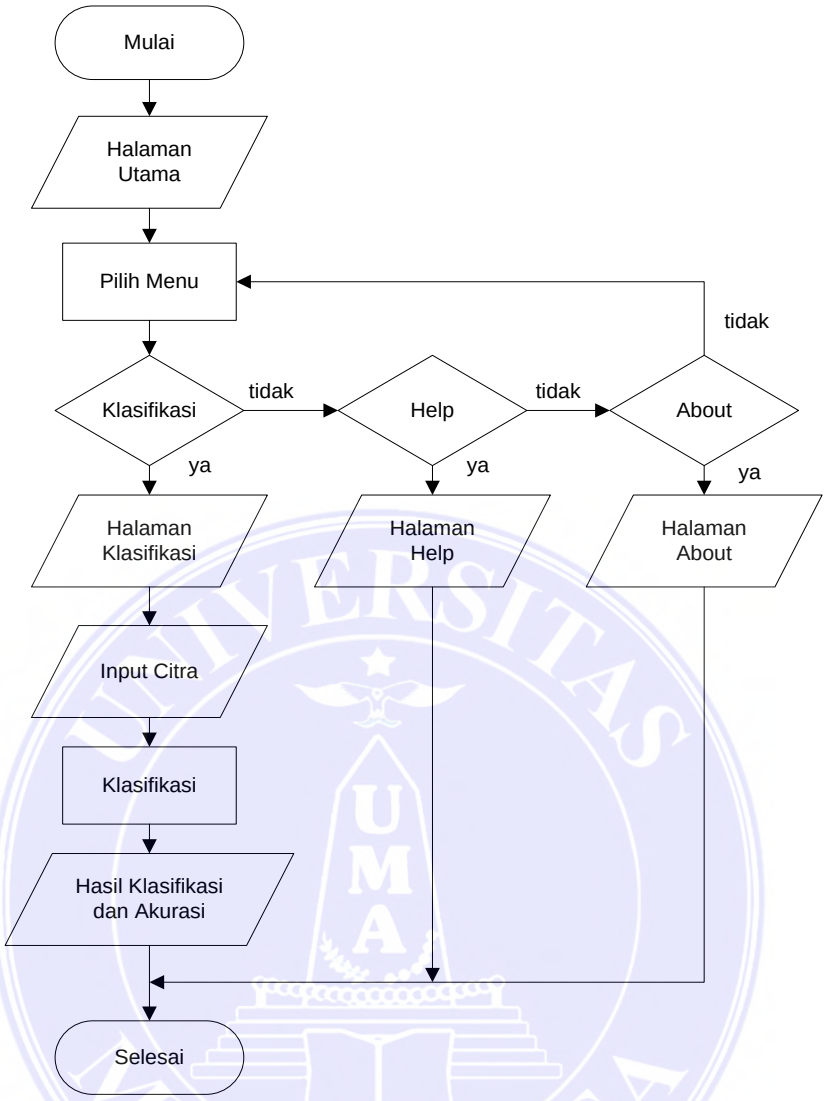
Fully Connected Layer pada Gambar 3.27 berperan sebagai *hidden layer* yang merupakan lapisan yang menentukan ke *output* manakah citra akan dikelompokan.

3.3 Perancangan Sistem

Perancangan sistem merupakan wujud dari implementasi sistem secara teknis. Perancangan sistem dalam penelitian ini dilakukan dengan membuat *flowchart* sistem dan merancang antarmuka (*interface*) sistem. *Flowchart* sistem digunakan untuk menggambarkan urutan proses secara detail dan hubungan antara suatu proses atau instruksi dengan proses lainnya dalam suatu ptogram. Sedangkan perancangan antarmuka (*interface*) sistem bertujuan untuk membuat tampilan sistem yang sederhana dan mudah digunakan (*user friendly*) sehingga *user* dapat lebih mudah dalam menggunakan sistem klasifikasi daun teh siap panen.

3.3.1 Flowchart Sistem

Flowchart sistem pada penelitian ini digunakan untuk menggambarkan urutan proses secara detail dan hubungan antara suatu proses atau instruksi dengan proses lainnya dalam aplikasi klasifikasi daun teh siap panen. *Flowchart* sistem pada aplikasi yang dirancang dapat ditampilkan pada Gambar 3.28.



Gambar 3.28 Flowchart Sistem

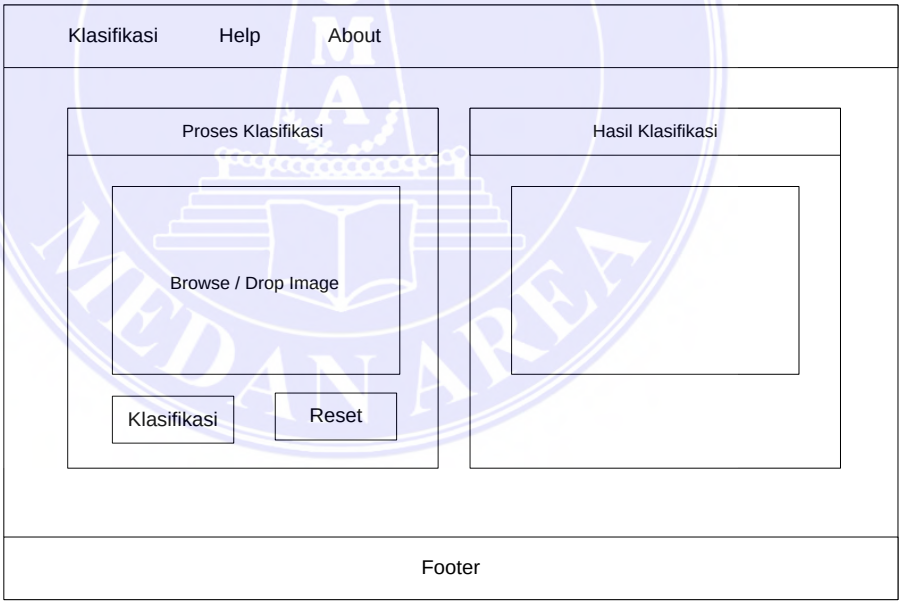
Gambar 3.28 menunjukkan *flowchart* dari alur proses pada aplikasi yang dimulai dengan menampilkan halaman utama. Pada halaman utama, terdapat tiga buah menu yang dapat diakses oleh *user*. Jika memilih menu klasifikasi maka sistem akan menampilkan halaman klasifikasi, dimana pada halaman ini akan di inputkan sebuah citra dan sistem akan melakukan proses lalu menampilkan hasil klasifikasi serta nilai akurasinya. Jika menu yang dipilih adalah menu *help* maka sistem akan menampilkan halaman *help*, dimana pada halaman ini akan ditampilkan informasi mengenai cara menggunakan aplikasi. Jika yang dipilh menu *about* maka sistem akan menampilkan halaman *about*, dimana pada halaman ini sistem akan menampilkan informasi mengenai profil pembuat sistem.

3.3.2 Rancangan Antarmuka Sistem

Perancangan antarmuka sistem yaitu membuat tampilan *interface* (antarmuka) sistem yang akan di integrasikan dengan aplikasi pada tahap implementasi sistem. Rancangan antarmuka sistem dibuat dengan tampilan yang sederhana dan mudah digunakan (*user friendly*) sehingga *user* dapat lebih mudah dalam menggunakan sistem. *Interface* yang akan dirancang pada sistem ini memiliki tiga bagian utama, yaitu halaman klasifikasi, halaman *help*, dan halaman *about*. Adapun bentuk rancangan antarmuka sistem dapat diuraikan yaitu:

1. Rancangan Halaman Klasifikasi

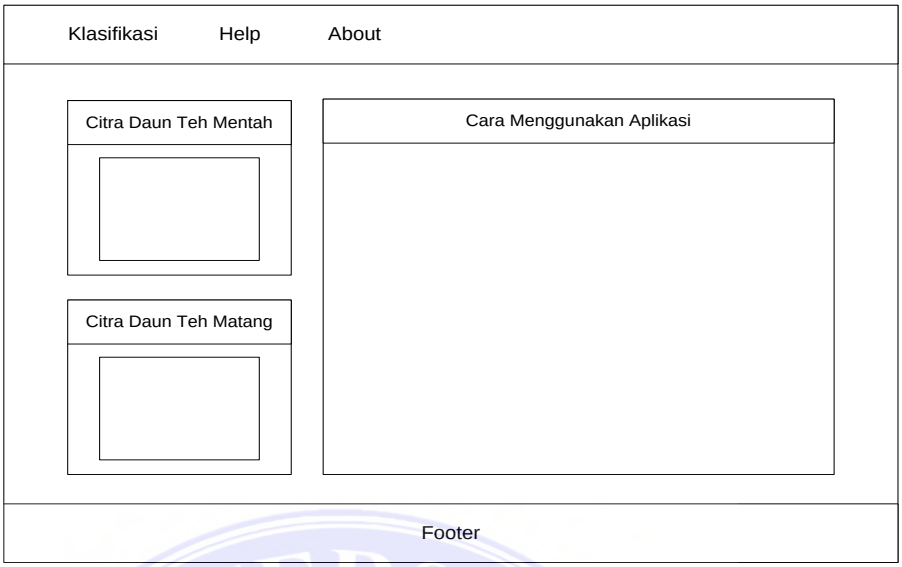
   Halaman klasifikasi dirancang untuk menampilkan proses klasifikasi daun teh siap panen menggunakan metode CNN dengan arsitektur *MobileNetV2*. Pada halaman ini, *user* dapat memilih citra daun teh yang akan diklasifikasi untuk kemudian di *upload* kedalam sistem. Selanjutnya sistem akan mengidentifikasi untuk dikenali lalu menampilkan hasil klasifikasi beserta nilai akurasinya.



Gambar 3.29 Rancangan Halaman Klasifikasi

2. Rancangan Halaman *Help*

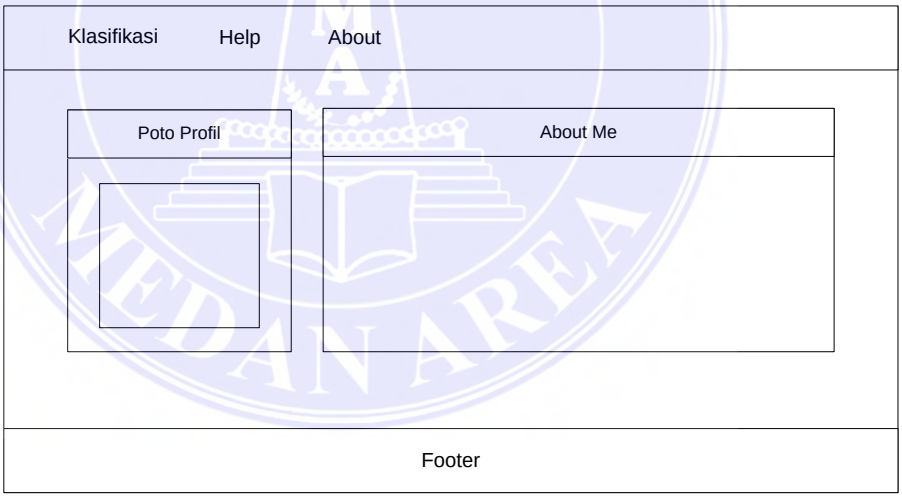
   Halaman *help* dirancang untuk menampilkan informasi panduan cara menggunakan aplikasi dalam melakukan proses klasifikasi daun teh siap panen. Adapun tampilan dari rancangan halaman *help* dapat dilihat pada Gambar 3.30.



Gambar 3.30 Rancangan Halaman Help

3. Rancangan Halaman *About*

Halaman *about* dirancang untuk menampilkan informasi mengenai profil pembuat sistem yang mencakup judul, nama, npm dan program studi. Rancangan halaman *about* dapat dilihat pada Gambar 3.31.



Gambar 3.31 Rancangan Halaman About

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari hasil pengujian sistem dalam melakukan klasifikasi daun teh siap panen dengan menggunakan arsitektur *MobileNetV2* adalah sebagai berikut:

1. Penerapan metode *Convolutional Neural Network* dengan arsitektur *MobileNetV2* menghasilkan performa yang baik terhadap semua *dataset* sehingga dapat menghasilkan klasifikasi daun teh siap panen secara akurat.
2. Berdasarkan hasil *training* pada 6 skenario model yang diuji, diperoleh tingkat akurasi tertinggi pada pengujian skenario *model* 2 sebesar 100% menggunakan *hyperparameter epoch* 50, *input shape* : 224x224x3 (*RGB channel*), *batch size* : 32, dan *optimizer* : *Adam*.
3. Hasil akurasi dari model setelah diuji menggunakan *data testing* (100 citra per *class*) didapatkan hasil akurasi sebesar 100% dengan nilai presisi (*precision*) sebesar 100%, *recall* 100%, dan *f1-score* 100%.

5.2 Saran

Saran yang dapat penulis berikan untuk pengembangan selanjutnya adalah sebagai berikut:

1. Penelitian selanjutnya dapat mengembangkan (*deploy*) *model* kedalam aplikasi berbasis *mobile* dan proses klasifikasi dapat dilakukan secara *real time*.
2. Menggunakan spesifikasi perangkat yang lebih tinggi, khusunya pada *Random Access Memory* (RAM) dan *Graphics Processing Unit* (GPU) untuk mempercepat proses *training* model.

DAFTAR PUSTAKA

Akmal Hariz, F., Nurma Yulita, I., & Suryana, I. (2022). Human Activity Recognition Berdasarkan Tangkapan Webcam Menggunakan Metode Convolutional Neural Network (CNN) Dengan Arsitektur MobileNet Human Activity Recognition Berdasarkan Tangkapan Webcam Menggunakan Metode Convolutional Neural Network (CNN) Dengan . *Ilmiah Teknologi Sistem Informasi*, *3*(4), 103–115. http://jurnal-itsi.org

Allaam, M. R. R., & Wibowo, A. T. (2021). Klasifikasi Genus Tanaman Anggrek Menggunakan Convolutional Neural Network (CNN). *Proceeding of Engineering*, *8*(2), 1153. . https://doi.org/https://doi.org/10.34818/eoe.v8i2.14708

Asrianda, A., Aidilof, H. A. K., & Pangestu, Y. (2021). Machine Learning for Detection of Palm Oil Leaf Disease Visually using Convolutional Neural Network Algorithm. *Journal of Informatics and Telecommunication Engineering*, *4*(2), 286–293. https://doi.org/10.31289/jite.v4i2.4185

Auliasari, R. N., Novamizanti, L., & Ibrahim, N. (2020). Identifikasi Kematangan Daun Teh Berbasis Fitur Warna Hue Saturation Intensity (HSI) dan Hue Saturation Value (HSV). *JUITA: Jurnal Informatika*, *8*(2), 217. https://doi.org/10.30595/juita.v8i2.7387

Damanik, Y., Okprana, H., & Sormin, R. K. (2022). Sistem Pendukung Keputusan Penentuan Produk Teh Terbaik Menggunakan Metode Moora Pada PTPN IV Sidamanik. *Buletin Big Data, Data Science and Artificial Intelligence Sistem*, *1*(1), 24–33 https://ejurnal.pdsi.or.id/index.php/zahra/article/view/11

Darwis, M. R. (2022). *Pengenalan Ekspresi Wajah Pada Video Berbasis Deep Learning Menngunakan MObileNetV2*. UIN SUSKA RIAU.

Dwijayana, I. G. D., & Wibawa, I. G. A. (2022). Implementasi Transfer Learning Dalam Klasifikasi Penyakit Pada Daun Teh Menggunakan MobileNetV2. *Jurnal Nasional Teknologi Informasi Dan Aplikasinya*, *1*(1), 379–387.

Ibrahim, N., Lestary, G. A., & Hanafi, F. S. (2022). Klasifikasi Tingkat Kematangan Pucuk Daun Teh menggunakan Metode Convolutional Neural

Network. *Jurnal Energi Elektrik, Teknik Telekomunikasi & Teknik Elektronika.*, *10*(1), 162–176.

Jaelani, A. A., Supratman, F. Y., & Ibrahim, N. (2020). *Perancangan Aplikasi Untuk Klasifikasi Klon Daun Teh Seri Gambung (Gmb) Menggunakan Algoritma Convolutional Neural Network*. *7*(2), 2920–2928. https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/13008

Juliansyah, S., & Laksito, A. D. (2021). Klasifikasi Citra Buah Pir Menggunakan Convolutional Neural Networks. *Jurnal Telekomunikasi Dan Komputer*, *11*(1), 65. https://doi.org/10.22441/incomtech.v11i1.10185

Kementan RI. (2015). Pedoman Produksi, Sertifikasi, Peredaran dan Pengawasan Benih Tanaman Teh (Camellia sinensis (L) O. Kuntze). *Kementrian Pertaninan RI*, *L*, 1–28.

Kholik, A. (2021). Klasifikasi Menggunakan Convolutional Neural Network (Cnn) Pada Tangkapan Layar Halaman Instagram. *Jdmsi*, *2*(2), 10–20.

Mudzakir, I., & Arifin, T. (2022). Klasifikasi Penggunaan Masker dengan Convolutional Neural Network Menggunakan Arsitektur MobileNetv2. *EXPERT: Jurnal Manajemen Sistem Informasi Dan Teknologi*, *12*(1), 76. https://doi.org/10.36448/expert.v12i1.2466

Nadila, A., Saragih, L., & Tarigan, W. J. (2022). Pengaruh Reactional Satisfacation Terhadap Minat Kunjung Ulang Wisata Kebun Teh Sidamanik Dengan Citra Destinasi Sebagai Variabel Moderasi. *Jurnal Ekonomi Integra*, *12*(2), 133–142. http://journal.stieip.ac.id/index.php/iga

Rahmat, S. (2020). Pengaruh Pemberdayaan Petani Terhadap Penerapan Teknologi Budidaya Serta Implikasinya Pada Produktivitas Kebun Teh Rakyat (Camellia Sinensis). *Tersedia Pada: Repository. Unwim. Ac. Id*, *c*. https://repository.unwim.ac.id/file/mahasiswa/1383152668.pdf

Ramayanti, D., Asri, D., & Lionie, L. (2022). Implementasi Model Arsitektur VGG16 dan MobileNetV2 Untuk Klasifikasi Citra Kupu-Kupu. *JSAI: Journal Scientific and Applied Informatics*, *5*(3), 182–187. https://doi.org/10.36085/jsai.v5i3.2864

Suherman, A. H., Ibrahim, N., Syahrian, H., Rahadi, V. P., & Prayoga, M. K.

(2021). *Klasifikasi Daun Teh Gambung Varietas Assamica menggunakan Convotional Neural Network dengan Arsitektur Lenet-5*. *4*(2), 63–71. 10.31289/jesce.v4i2.4136

Wicaksono, B. A. (2019). *Identifikasi Kematangan Daun Teh Menggunakan Centroid Clustering Berbasis Ruang Warna YCbCr* [Universitas Telkom]. https://openlibrary.telkomuniversity.ac.id/pustaka/153829/identifikasi-tingkat-kematangan-daun-teh-menggunakan-centroid-clustering-berdasarkan-ruang-warna-ycbcr.html

Zaelani, F., & Miftahuddin, Y. (2022). Perbandingan Metode EfficientNetB3 dan MobileNetV2 Untuk Identifikasi Jenis Buah-buahan Menggunakan Fitur Daun. *Jurnal Ilmiah Teknologi Infomasi Terapan*, *9*(1), 1–11. https://doi.org/10.33197/jitter.vol9.iss1.2022.911

LAMPIRAN

Lampiran 1 Kode Program

Kode Program Klasifikasi Menggunakan Google Colab

```
# cek versi tensorflow, keras dan python
import tensorflow as tf
from tensorflow import keras
import sys

print("Versi Tensorflow :", tf.__version__)
print("Versi Keras      :", keras.__version__)
print("Versi Python     :", sys.version)
```

```
# import library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from tensorflow.keras.applications.mobilenet_v2 import
  MobileNetV2
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,
  GlobalAveragePooling2D
from tensorflow.keras.preprocessing import
  image_dataset_from_directory
from tensorflow.keras.preprocessing.image import
  ImageDataGenerator
from tensorflow.keras.optimizers import Adam

from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix,
  classification_report
import seaborn as sns

from IPython.display import clear_output
import warnings
warnings.filterwarnings('ignore')
```

1. DATASET

```
# mount ke GoogleDrive (dataset disimpan dalam google drive)
from google.colab import drive
drive.mount('/content/drive')
```

```
# persiapan direktori dataset (data training, data validation,
  dan data testing)
```

```
train_dir = '/content/drive/MyDrive/Dataset/daun-teh/train'
test_dir = '/content/drive/MyDrive/Dataset/daun-teh/test'
val_dir = '/content/drive/MyDrive/Dataset/daun-teh/val'
# menampilkan jumlah dataset
data_train = image_dataset_from_directory(train_dir)
data_validation = image_dataset_from_directory(val_dir)
data_test = image_dataset_from_directory(test_dir)

# visualisasi dataset
classes_train = data_train.class_names
plt.figure(figsize = (10,10))
for img, label in data_train.take(1):
    for i in range(10):
        ax = plt.subplot(4,5,i+1)
        plt.imshow(img[i].numpy().astype('uint8'))
        plt.title(classes_train[int(label[i])])
        plt.axis('off')

clear_output()
# Augmentasi Dataset

IMG_SIZE = (224, 224) # menentukan size citra/input shape

# menggunakan ImageDataGenerator untuk proses augmentasi
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   rotation_range = 20,
                                   horizontal_flip = True,
                                   zoom_range = 0.2)


test_datagen = ImageDataGenerator(rescale = 1./255)

train_dataset = train_datagen.flow_from_directory(train_dir,
    target_size = (IMG_SIZE), # target_size = dimensi dari citra
    yang akan digunakan dalam proses training
                                                  color_mode = "rgb",
                                                  batch_size = 32, #
    batch_size = banyaknya citra yang dimasukan dalam setiap
    steps training
                                                  shuffle = True, #
    untuk acak data sehingga data tidak akan urut
                                                  class_mode =
    "categorical") # class_mode = metode pemilihan klasifikasi
    (categorical=1,2,3 atau binary=[1,0,1])

```

```
test_dataset = test_datagen.flow_from_directory(test_dir,
    target_size = (IMG_SIZE),
                                                color_mode = "rgb",
                                                batch_size = 32,
                                                shuffle = True,
                                                class_mode =
    "categorical")

validation_dataset = train_datagen.flow_from_directory(val_dir,
    target_size = (IMG_SIZE),
                                                color_mode = "rgb",
                                                batch_size = 32,
                                                shuffle = True,
                                                class_mode =
    "categorical")
```

2. Build Model MobileNetV2

```
# mendefinisikan terlebih dahulu base-model yang akan di
  gunakan
# dalam hal ini model dibangun menggunakan arsitektur
  MobileNetV2

mobilenet =
  MobileNetV2(weights='imagenet',include_top=False,input_shape=(2
  24,224,3))

# pre trained model MobileNetV2 dibuat jadi non trainable
for layer in mobilenet.layers:
    layer.trainable = False
```

2.1 Model dengan MobileNetV2 Original

Skenario 1 : input shape=224,224; batch_size=32; Optimizer=Adam; epoch=20

```
# Inisialisai model jadi sequential
model_skenario1 = Sequential()

# Feature Extractor Layer
model_skenario1.add(mobilenet) # tambahkan model MobileNetV2
  kedalam sequence model
model_skenario1.add(GlobalAveragePooling2D())

# Flatten Layer
# model_skenario1.add(Flatten())

   # Fully Connected Layer
```

```
    # model_skenario1.add(Dense(128, activation="relu",
kernel_initializer="he_uniform"))

    # output layer
    model_skenario1.add(Dense(2,activation='sigmoid'))

    # summary model
    model_skenario1.summary()
```

```
# compile model
model_skenario1.compile(optimizer=Adam(),
                loss='categorical_crossentropy',
                metrics = ['accuracy'])
```

```
# fitting model (training model arsitektur yang sudah dibuat)
history = model_skenario1.fit(train_dataset,
validation_data=validation_dataset,
                              epochs=20)
```

```
# evaluasi model hasil training
model_skenario1.evaluate(validation_dataset)
```

```
# menampilkan grafik akurasi model
acc_model_skenario1 = history.history['accuracy']
val_acc_model_skenario1 = history.history['val_accuracy']
loss_model_skenario1 = history.history['loss']
val_loss_model_skenario1 = history.history['val_loss']
epochs_range = range(len(acc_model_skenario1))

plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
    plt.plot(epochs_range, acc_model_skenario1,"go-",
label='Training Accuracy')
    plt.plot(epochs_range, val_acc_model_skenario1,"ro-",
label='Validation Accuracy')
    plt.title('Training and Validation Accuracy Model Skenario
1')
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.legend(loc='lower right')

    plt.subplot(2, 2, 2)
    plt.plot(epochs_range, loss_model_skenario1,"go-",
label='Training Loss')
```

```
19.   plt.plot(epochs_range, val_loss_model_skenario1,"ro-",
  label='Validation Loss')
20.   plt.title('Training and Validation Loss Model Skenario 1')
21.   plt.xlabel("Epoch")
22.   plt.ylabel("Loss")
23.   plt.legend(loc='upper right')
24.
25.   plt.show()
```

```
1. # simpan model skenario 1
2. model_skenario1.save("/content/drive/MyDrive/Dataset/model_sken
  ario1.h5")
```

2.2 Model dengan MobileNetV2 Original

Skenario 2 : input shape=224,224; batch_size=32; Optimizer=Adam; epoch=50

```
1.  # Inisialisai model jadi sequential
2.  model_skenario2 = Sequential()
3.
4.  # Feature Extractor Layer
5.  model_skenario2.add(mobilenet) # tambahkan model MobileNetV2
    model kedalam sequence model
6.  model_skenario2.add(GlobalAveragePooling2D())
7.
8.  # Flatten Layer
9.  # model_skenario2.add(Flatten())
10.
11. # Fully Connected Layer
12. # model_skenario2.add(Dense(128, activation="relu",
    kernel_initializer="he_uniform"))
13. # output layer
14. model_skenario2.add(Dense(2,activation='sigmoid'))
15.
16. # summary model
17. model_skenario2.summary()
```

```
1.  # compile model
2.  model_skenario2.compile(optimizer=Adam(),
3.                 loss='categorical_crossentropy',
4.                 metrics = ['accuracy'])
5.
```

```
1.  # fitting model (training model arsitektur yang sudah dibuat)
2.  history = model_skenario2.fit(train_dataset,
3.
    validation_data=validation_dataset,
4.                                  epochs=50)
```

```
# evaluasi model hasil training
model_skenario2.evaluate(validation_dataset)
```

```
# menampilkan grafik akurasi model
acc_model_skenario2 = history.history['accuracy']
val_acc_model_skenario2 = history.history['val_accuracy']
loss_model_skenario2 = history.history['loss']
val_loss_model_skenario2 = history.history['val_loss']
epochs_range = range(len(acc_model_skenario2))

plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc_model_skenario2,"go-",
label='Training Accuracy')
plt.plot(epochs_range, val_acc_model_skenario2,"ro-",
label='Validation Accuracy')
plt.title('Training and Validation Accuracy Model Skenario 2')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(loc='lower right')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss_model_skenario2,"go-",
label='Training Loss')
plt.plot(epochs_range, val_loss_model_skenario2,"ro-",
label='Validation Loss')
plt.title('Training and Validation Loss Model Skenario 2')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc='upper right')

plt.show()
```

```
# simpan model skenario 2
model_skenario2.save("/content/drive/MyDrive/Dataset/model_ske
nario2.h5")
```

2.3 Model dengan MobileNetV2 Original

Skenario 3 : input shape=224,224; batch_size=32; Optimizer=Adam; epoch=20; learning_rate=0.0003

```
# Inisialisai model jadi sequential
model_skenario3 = Sequential()

# Feature Extractor Layer
# tambahkan model MobileNetV2 model kedalam sequence model
```

```
model_skenario3.add(mobilenet)
model_skenario3.add(GlobalAveragePooling2D())

# Flatten Layer
# model_skenario3.add(Flatten())

# Fully Connected Layer
# model_skenario3.add(Dense(128, activation="relu",
kernel_initializer="he_uniform"))
# model_skenario3.add(Dropout(0.3))

# output layer
model_skenario3.add(Dense(2,activation='sigmoid'))
# model_skenario3.add(Dropout(0.3))

# summary model
model_skenario3.summary()
```

```
 # compile model
 opt = Adam(learning_rate=0.0003)
 model_skenario3.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics = ['accuracy'])
```

```
 # fitting model (training model arsitektur yang sudah dibuat)
 history = model_skenario3.fit(train_dataset,

validation_data=validation_dataset,
                              epochs=20)
```

```
 # fitting model (training model arsitektur yang sudah dibuat)
 history = model_skenario3.fit(train_dataset,

validation_data=validation_dataset,
                        # evaluasi model hasil training
 model_skenario3.evaluate(validation_dataset)       epochs=20)
```

```
 # fitting model (training model arsitektur yang sudah dibuat)
 history = model_skenario3.fit(train_dataset,

validation_data=validation_dataset,
                        # evaluasi model hasil training
 model_skenario3.evaluate(validation_d# menampilkan grafik
akurasi model
 acc_model_skenario3 = history.history['accuracy']
 val_acc_model_skenario3 = history.history['val_accuracy']
```

```
loss_model_skenario3 = history.history['loss']
val_loss_model_skenario3 = history.history['val_loss']
epochs_range = range(len(acc_model_skenario3))

plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc_model_skenario3,"go-",
label='Training Accuracy')
plt.plot(epochs_range, val_acc_model_skenario3,"ro-",
label='Validation Accuracy')
plt.title('Training and Validation Accuracy Model Skenario
3')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(loc='lower right')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss_model_skenario3,"go-",
label='Training Loss')
plt.plot(epochs_range, val_loss_model_skenario3,"ro-",
label='Validation Loss')
plt.title('Training and Validation Loss Model Skenario 3')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(loc='upper right')

plt.show()ataset)        epochs=20)
```

```
# simpan model
model_skenario3.save("/content/drive/MyDrive/Dataset/model_sk
enario3.h5")
```

2.4 Model dengan MobileNetV2 Original

Skenario 4 : input shape=224,224; batch_size=32; Optimizer=Adam; epoch=20; learning_rate=0.0005

```
# Inisialisai model jadi sequential
model_skenario4 = Sequential()

# Feature Extractor Layer
# tambahkan model MobileNetV2 model kedalam sequence model
model_skenario4.add(mobilenet)
model_skenario4.add(GlobalAveragePooling2D())

# Flatten Layer
# model_skenario4.add(Flatten())
```

```
11.
12. # Fully Connected Layer
13. # model_skenario4.add(Dense(128, activation="relu",
    kernel_initializer="he_uniform"))
14. # model_skenario4.add(Dropout(0.3))
15.
16. # output layer
17. model_skenario4.add(Dense(2,activation='sigmoid'))
18. # model_skenario4.add(Dropout(0.3))
19.
20. # summary model
21. model_skenario4.summary()
```

```
1.  # compile model
2.  opt = Adam(learning_rate=0.0005)
3.  model_skenario4.compile(optimizer=opt,
4.                loss='categorical_crossentropy',
5.                metrics = ['accuracy'])
```

```
1.  # fitting model (training model arsitektur yang sudah dibuat)
2.  history = model_skenario4.fit(train_dataset,
3.
    validation_data=validation_dataset,
4.                                epochs=20)
```

```
1.  # evaluasi model hasil training
2.  model_skenario4.evaluate(validation_dataset)
```

```
1.  # menampilkan grafik akurasi model
2.  acc_model_skenario4 = history.history['accuracy']
3.  val_acc_model_skenario4 = history.history['val_accuracy']
4.  loss_model_skenario4 = history.history['loss']
5.  val_loss_model_skenario4 = history.history['val_loss']
6.  epochs_range = range(len(acc_model_skenario4))
7.
8.  plt.figure(figsize=(15, 10))
9.  plt.subplot(2, 2, 1)
10. plt.plot(epochs_range, acc_model_skenario4,"go-",
    label='Training Accuracy')
11. plt.plot(epochs_range, val_acc_model_skenario4,"ro-",
    label='Validation Accuracy')
12. plt.title('Training and Validation Accuracy Model Skenario 4')
13. plt.xlabel("Epoch")
14. plt.ylabel("Accuracy")
15. plt.legend(loc='lower right')
16.
```

```
plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss_model_skenario4,"go-",
label='Training Loss')
plt.plot(epochs_range, val_loss_model_skenario4,"ro-",
label='Validation Loss')
plt.title('Training and Validation Loss Model Skenario 4')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc='upper right')

plt.show()
```

```
# simpan model
model_skenario4.save("/content/drive/MyDrive/Dataset/model_ske
nario4.h5")
```

2.5 Model dengan MobileNetV2 Original

Skenario 5 : input shape=224,224; batch_size=32; Optimizer=Adam; epoch=50; learning_rate=0.0003

```
2.5 Model dengan MobileNetV2 Original

Skenario 5 : input shape=224,224; batch_size=32;
Optimizer=Adam; epoch=50; learnin# Inisialisai model jadi
sequential
model_skenario5 = Sequential()

# Feature Extractor Layer
# tambahkan model MobileNetV2 model kedalam sequence model
model_skenario5.add(mobilenet)
model_skenario5.add(GlobalAveragePooling2D())

# Flatten Layer
# model_skenario5.add(Flatten())

# Fully Connected Layer
# model_skenario5.add(Dense(128, activation="relu",
kernel_initializer="he_uniform"))
# model_skenario5.add(Dropout(0.3))

# output layer
model_skenario5.add(Dense(2,activation='sigmoid'))
# model_skenario5.add(Dropout(0.3))

# summary model
model_skenario5.summary()g_rate=0.0003
```

```
# compile model
opt = Adam(learning_rate=0.0003)
model_skenario5.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics = ['accuracy'])
```

```
# fitting model (training model arsitektur yang sudah dibuat)
history = model_skenario5.fit(train_dataset,
validation_data=validation_dataset,
                              epochs=50)
```

```
# evaluasi model hasil training
model_skenario5.evaluate(validation_dataset)
# menampilkan grafik akurasi model
acc_model_skenario5 = history.history['accuracy']
val_acc_model_skenario5 = history.history['val_accuracy']
loss_model_skenario5 = history.history['loss']
val_loss_model_skenario5 = history.history['val_loss']
epochs_range = range(len(acc_model_skenario5))

plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc_model_skenario5,"go-",
label='Training Accuracy')
plt.plot(epochs_range, val_acc_model_skenario5,"ro-",
label='Validation Accuracy')
plt.title('Training and Validation Accuracy Model Skenario 5')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(loc='lower right')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss_model_skenario5,"go-",
label='Training Loss')
plt.plot(epochs_range, val_loss_model_skenario5,"ro-",
label='Validation Loss')
plt.title('Training and Validation Loss Model Skenario 5')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc='upper right')

plt.show()
```

```
# simpan model
```

```
2. model_skenario5.save("/content/drive/MyDrive/Dataset/model_ske
   nario5.h5")
```

2.6 Model dengan MobileNetV2 Original

Skenario 6 : input shape=224,224; batch_size=32; Optimizer=Adam; epoch=50; learning_rate=0.0005

```
1. # Inisialisai model jadi sequential
2. model_skenario6 = Sequential()
3.
4. # Feature Extractor Layer
5. # tambahkan model MobileNetV2 model kedalam sequence model
6. model_skenario6.add(mobilenet)
7. model_skenario6.add(GlobalAveragePooling2D())
8.
9. # Flatten Layer
10. # model_skenario6.add(Flatten())
11.
12. # Fully Connected Layer
13. # model_skenario6.add(Dense(128, activation="relu",
    kernel_initializer="he_uniform"))
14. # model_skenario6.add(Dropout(0.3))
15.
16. # output layer
17. model_skenario6.add(Dense(2,activation='sigmoid'))
18. # model_skenario6.add(Dropout(0.3))
19.
20. # summary model
21. model_skenario6.summary()
```

```
1. # compile model
2. opt = Adam(learning_rate=0.0005)
3. model_skenario6.compile(optimizer=opt,
4.                 loss='categorical_crossentropy',
5.                 metrics = ['accuracy'])
```

```
1. # fitting model (training model arsitektur yang sudah dibuat)
2. history = model_skenario6.fit(train_dataset,
3.
   validation_data=validation_dataset,
4.                                 epochs=50)
```

```
1. # evaluasi model hasil training
2. model_skenario6.evaluate(validation_dataset)
```

```
1. # menampilkan grafik akurasi model
```

```
acc_model_skenario6 = history.history['accuracy']
val_acc_model_skenario6 = history.history['val_accuracy']
loss_model_skenario6 = history.history['loss']
val_loss_model_skenario6 = history.history['val_loss']
epochs_range = range(len(acc_model_skenario6))

plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc_model_skenario6,"go-",
label='Training Accuracy')
plt.plot(epochs_range, val_acc_model_skenario6,"ro-",
label='Validation Accuracy')
plt.title('Training and Validation Accuracy Model Skenario 6')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(loc='lower right')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss_model_skenario6,"go-",
label='Training Loss')
plt.plot(epochs_range, val_loss_model_skenario6,"ro-",
label='Validation Loss')
plt.title('Training and Validation Loss Model Skenario 6')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc='upper right')

plt.show()
```

```
# simpan model
model_skenario6.save("/content/drive/MyDrive/Dataset/model_ske
nario6.h5")
```

3. Perbandingan Akurasi Semua Model

```
model_skenario1 =
load_model("/content/drive/MyDrive/Dataset/model_skenario1.h5"
)
val_acc_model_skenario1 =
model_skenario1.evaluate(validation_dataset)
print(val_acc_model_skenario1)

model_skenario2 =
load_model("/content/drive/MyDrive/Dataset/model_skenario2.h5"
)
```

```
val_acc_model_skenario2 =
model_skenario2.evaluate(validation_dataset)
print(val_acc_model_skenario2)

model_skenario3 =
load_model("/content/drive/MyDrive/Dataset/model_skenario3.h5"
)
val_acc_model_skenario3 =
model_skenario3.evaluate(validation_dataset)
print(val_acc_model_skenario3)

model_skenario4 =
load_model("/content/drive/MyDrive/Dataset/model_skenario4.h5"
)
val_acc_model_skenario4 =
model_skenario4.evaluate(validation_dataset)
print(val_acc_model_skenario4)

model_skenario5 =
load_model("/content/drive/MyDrive/Dataset/model_skenario5.h5"
)
val_acc_model_skenario5 =
model_skenario5.evaluate(validation_dataset)
print(val_acc_model_skenario5)

model_skenario6 =
load_model("/content/drive/MyDrive/Dataset/model_skenario6.h5"
)
val_acc_model_skenario6 =
model_skenario6.evaluate(validation_dataset)
print(val_acc_model_skenario6)
```

```
plt.figure(figsize = (15, 5))
plt.plot(val_acc_model_skenario1)
plt.plot(val_acc_model_skenario2)
plt.plot(val_acc_model_skenario3)
plt.plot(val_acc_model_skenario4)
plt.plot(val_acc_model_skenario5)
plt.plot(val_acc_model_skenario6)

plt.title('Model Validation Accuracy')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(['Model Skenario 1','Model Skenario 2','Model
Skenario 3','Model Skenario 4','Model Skenario 5','Model
Skenario 6'], loc='lower right')
```

```
plt.grid(True)
plt.show()
```

4. Evaluasi Model

```
batch_size=32
target_size=(224,224)
test_path = '/content/drive/MyDrive/Dataset/daun-teh/test'

test_generator = test_datagen.flow_from_directory(
                  test_path,
                  target_size=target_size,
                  batch_size=batch_size,
                  class_mode=None,
                  shuffle=False)
test_generator.reset()

# Calling the saved model for making predictions
model_klasifikasi =
load_model("/content/drive/MyDrive/Dataset/model_skenario2.h5"
)
pred = model_klasifikasi.predict(test_generator, verbose=1)
predicted_class_indices=np.argmax(pred,axis=1)
labels = (test_generator.class_indices)
labels = dict((v,k) for k,v in labels.items())
predictions = [labels[k] for k in predicted_class_indices]
filenames = test_generator.filenames
results = pd.DataFrame({"Filename":filenames,
                        "Predictions":predictions})

# create a function for visualizing model performance
def PerformanceReports(conf_matrix, class_report, labels):
    ax = plt.subplot(2, 2, 1)
    ax = plt.subplot()
    sns.heatmap(conf_matrix, cmap="crest_r", annot=True,
fmt='.4g', linewidths=2, linecolor='white', cbar=True, ax=ax)
    #labels, title and ticks
    ax.set_xlabel('Predicted labels', fontsize=12,
color="darkblue", labelpad=24)
    ax.set_ylabel('True labels', fontsize=12, color="darkblue",
labelpad=24)
    ax.set_title('Confusion Matrix Model Skenario 1',
fontsize=12, pad=24)
    ax.xaxis.set_ticklabels(labels)
    ax.yaxis.set_ticklabels(labels)
    plt.show()
```

```
36.
37.     # ax= plt.subplot(2, 2, 2)
38.     # ax = plt.subplot()
39.     # sns.heatmap(pd.DataFrame(class_report).iloc[:-1, :].T,
40.                 # annot=True,ax=ax)
41.     # ax.set_title('Classification Report Model Skenario 1',
    fontsize=12, pad=24)
42.     plt.show()
43.
44.
45. labels = ['kepel', 'peko']
46. test_labels = [fn.split('/')[0] for fn in filenames]
47. cm = confusion_matrix(test_labels, predictions)
48. print(cm)
49.
50. cr = classification_report(test_labels, predictions)
51. class_report = classification_report(test_labels,
52.                                       predictions,
53.                                       target_names=labels,
54.                                       output_dict=True)
55. print(cr)
56. PerformanceReports(cm, class_report, labels)
```

5. Prediksi

```
1.  import requests
2.  from io import BytesIO
3.
4.  from PIL import Image
5.  import numpy as np
6.
7.  # Parameters
8.  input_size = (224,224)
9.
10. #define input shape
11. channel = (3,)
12. input_shape = input_size + channel
13.
14. #define labels
15. labels = ['matang', 'mentah']
16.
17. # Define preprocess function
18. def preprocess(img,input_size):
19.     nimg = img.convert('RGB').resize(input_size, resample= 0)
20.     img_arr = (np.array(nimg))/255
21.     return img_arr
```

```
22.
23. def reshape(imgs_arr):
24.     return np.stack(imgs_arr, axis=0)
```

```
1.  # Load model
2.  model =
    load_model("/content/drive/MyDrive/Dataset/model_skenario2.h5"
    ) # model yang di load adalah model sudah disimpan sebelumnya
3.
4.  # Predict the image
5.  # read image
6.  im = Image.open('/content/drive/MyDrive/Dataset/daun-
    teh/test/matang/0904matang.jpg') # sesuaikan direktori untuk
    data testing yang akan diprediksi
7.  X = preprocess(im,input_size)
8.  X = reshape([X])
9.  y = model.predict(X)
10.
11. print( labels[np.argmax(y)], np.max(y) )
```

Kode Progam Aplikasi Web

app.py

```
1.  import os
2.  import sys
3.  import numpy as np
4.  from util import base64_to_pil
5.  from flask import Flask, redirect, url_for, request,
    render_template, Response, jsonify, redirect
6.  from werkzeug.utils import secure_filename
7.  #from gevent.pywsgi import WSGIServer
8.  import tensorflow as tf
9.  from tensorflow import keras
10. from tensorflow.keras.applications.imagenet_utils import
    preprocess_input, decode_predictions
11. from tensorflow.keras.models import load_model
12. from tensorflow.keras.preprocessing import image
13. from tensorflow.keras.utils import get_file
14.
15.
16. app = Flask(__name__)
17.
18.
19. # LOAD TRAINED MODEL FROM LOCAL FOLDER
20. model = load_model('models/model_skenario2.h5') #    SESUAIKAN

21.
```

```
22. def model_predict(img, model):
23.     img = img.resize((224, 224))              #    SESUAIKAN
24.
25.     x = image.img_to_array(img)
26.     x = x.reshape(-1, 224, 224, 3)
27.     x = x.astype('float32')
28.     x = x / 255.0
29.     preds = model.predict(x)
30.     return preds
31.
32. @app.route('/', methods=['GET'])
33. def index():
34.     return render_template('index.html')
35.
36. @app.route('/predict', methods=['GET', 'POST'])
37. def predict():
38.     if request.method == 'POST':
39.         # Get the image from post request
40.         img = base64_to_pil(request.json)
41.
42.         # Save the image to ./uploads
43.         # img.save("./uploads/image.png")
44.
45.         # Make prediction
46.         preds = model_predict(img, model)
47.
48.
    #=============================================================
    ======================#
49.
50.         target_names = ['kepel', 'peko']     #    SESUAIKAN
51.
52.         hasil_label = target_names[np.argmax(preds)]
53.         hasil_prob = "{:.2f}".format(100 * np.max(preds)) # 2f
    adalah presisi angka dibelakang koma (coba ganti jadi 0f, 3f,
    dst)
54.
55.
    #=============================================================
    ======================#
56.
57.         return jsonify(result=hasil_label,
    probability=hasil_prob)
58.
59.     return None
60.
```

```
61. if __name__ == '__main__':
62.     # OPTION 1: NORMAL SERVE THE APP
63.     app.run(debug=True)
64.     # app.run(port=5002, threaded=False)
65.
66.     # OPTION 2: SERVE THE APP WITH GEVENT
67.     # Setiap merubah file di main.js, ubah juga 5000, menjadi
    5001, dst (alasan: cache)
68.     #http_server = WSGIServer(('0.0.0.0', 5050), app)
69.     #http_server.serve_forever()
```

Lampiran 2 Hasil Turnitin

turnitin

Similarity Report ID: oid:29477:44186949

PAPER NAME
Skripsi Febriady Marpaung (Plagiat).docx

AUTHOR
Febriady Marpaung

WORD COUNT
14824 Words

CHARACTER COUNT
94353 Characters

PAGE COUNT
82 Pages

FILE SIZE
33.0MB

SUBMISSION DATE
Oct 2, 2023 2:00 AM GMT+7

REPORT DATE
Oct 2, 2023 2:03 AM GMT+7

25% Overall Similarity
The combined total of all matches, including overlapping sources, for each database.

- 24% Internet database
- Crossref database
- 11% Submitted Works database
- 3% Publications database
- Crossref Posted Content database

Excluded from Similarity Report

- Small Matches (Less then 10 words)

Summary

Lampiran 3 SK Pembimbing

UNIVERSITAS MEDAN AREA
FAKULTAS TEKNIK
Kampus I : Jalan Kolam Nomor 1 Medan Estate/Jalan PBSI Nomor 1☎(061) 7366878, 7360168, 7364348, 7366781, Fax. 061) 7366998 Medan 20223
Kampus II : Jalan Setiabudi Nomor 79 / Jalan Sei Serayu Nomor 70 A, ☎ (061) 8225602, Fax. (061) 8226331 Medan 201 2
Website: www.teknik.uma.ac.id E-mail: univ_medanarea@uma.ac.id

Nomor : 127/FT.6/01.10/II/2023 16 Februari 2023
Lamp : -
H a l : **Perubahan Judul Tugas Akhir**

Yth, Pembimbing Tugas Akhir
Nurul Khairina, S.Kom, M.Kom
di
Tempat

Dengan hormat, Sehubungan dengan adanya perubahan judul tugas akhir maka perlu diterbitkan kembali SK Pembimbing Skripsi baru atas nama mahasiswa tersebut :

N a m a : Febriady Marpaung
N P M : 188160032
Jurusan : Teknik Informatika

Maka dengan hormat kami mengharapkan kesediaan saudara :

1. Nurul Khairina, S.Kom, M.Kom (Sebagai Pembimbing)

Adapun Tugas Akhir Skripsi berjudul :

"Klasifikasi Tingkat Kematangan Pucuk Daun Teh menggunakan *MobileNetV2*".

SK Pembimbing ini berlaku selama enam bulan terhitung sejak SK ini diterbitkan. Jika proses pembimbing melebihi batas waktu yang telah ditetapkan, SK ini dapat ditinjau ulang.

Demikian kami sampaikan, atas kesediaan saudara diucapkan terima kasih.

Dekan,

Dr. Rahmad Syah, S. Kom, M. Kom

Lampiran 4 SK Penelitian dan Pengambilan Data Tugas Akhir

UNIVERSITAS MEDAN AREA
FAKULTAS TEKNIK
Kampus I : Jalan Kolam Nomor 1 Medan Estate/Jalan PBSI Nomor 1☎(061) 7366878, 7360168, 7364348, 7366781, Fax.(061) 7366998 Medan 20223
Kampus II : Jalan Setiabudi Nomor 79 / Jalan Sei Serayu Nomor 70 A, ☎ (061) 8225602, Fax. (061) 8226331 Medan 20122
Website: www.teknik.uma.ac.id E-mail: univ_medanarea@uma.ac.id

Nomor : 290/FT.6/01.10/X/2022 5 Oktober 2022
Lamp : -
H a l : **Penelitian Dan Pengambilan Data Tugas Akhir**

Yth. Pimpinan PTPN IV Unit Kebun Sidamanik
Kec. Sidamanik
Di
Simalungun

Dengan hormat,
Kami mohon kesediaan Bapak/Ibu berkenan untuk memberikan izin dan kesempatan kepada mahasiswa kami tersebut dibawah ini :

NO	N A M A	N P M	PRODI
1	Febriady Marpaung	188160032	Informatika

Untuk melaksanakan Penelitian dan Pengambilan Data Tugas Akhir pada perusahaan/Instansi yang Bapak/Ibu Pimpin.

Perlu kami jelaskan bahwa Pengambilan Data tersebut adalah semata-mata untuk tujuan ilmiah dan Skripsi yang merupakan salah satu syarat bagi mahasiswa tersebut untuk mengikuti ujian sarjana lengkap pada Fakultas Teknik Universitas Medan Area dan tidak untuk dipublikasikan, dengan judul penelitian :

Klasifikasi Tingkat Kematangan Pucuk Daun Teh menggunakan *Least-Squares Support Vector Machine*

Atas perhatian dan kerja sama yang baik diucapkan terima kasih.

Dekan,

Dr. Rahmad Syah, S. Kom, M. Kom

Tembusan :
1. Ka. BAMAI
2. Mahasiswa
3. File

Lampiran 5 Surat Izin Riset

PT PERKEBUNAN NUSANTARA IV
MEDAN - SUMATERA UTARA - INDONESIA

- KANTOR PUSAT: JL LETJEND SUPRAPTO NO.2 MEDAN — TELP.: (061) 4154666 – FAX.: (061) 4573117
- KANTOR PERWAKILAN JAKARTA — TELP.: (021) 7231662 – FAX.: (021) 7231663

Nomor : 04.07/X/021640/XI/2022 — Medan, 04 November 2022
Lamp : -
Hal : IZIN RISET SARJANA

Kepada Yth :
DEKAN
UNIVERSITAS MEDAN AREA
JALAN KOLAM NO 1 MEDAN ESTATE / JALAN PBSI NO 1
MEDAN
Di - MEDAN

Membalas surat saudara/i nomor 290/FT.6/01.10/X/2022 tanggal : 05 Oktober 2022, Mahasiswa/Siswa/i TEKNIK Jurusan TEKNIK INFORMATIKA atas nama :

No.	Nama	NPM	Program Studi / Judul
1.	FEBRIADY MARPAUNG	188160032	KLASIFIKASI TINGKAT KEMATANGAN PUCUK DAUN TEH MENGGUNAKAN LEAST-SQUARES SUPPORT VECTOR MACHINE

Diizinkan untuk melakukan RISET dengan metode *ONLINE* (tanpa tatap muka) di PT Perkebunan Nusantara IV sebagai berikut :
Tempat : UNIT USAHA TEH
Bagian / Bidang : TANAMAN
Terhitung mulai tgl : 03 November 2022 s/d 03 Januari 2023

Sesuai dengan ketentuan yang berlaku di perusahaan disampaikan sebagai berikut :

1. Telah mengisi dan mengunggah kembali surat pernyataan yang menjadi persyaratan dalam proses riset secara *online*.
2. Semua biaya ditanggung oleh siswa/mahasiswa/i yang bersangkutan.
3. Yang bersangkutan wajib menjaga kerahasiaan data perusahaan yang digunakan dalam riset, serta semata-mata dipergunakan untuk kepentingan ilmiah pada Perguruan Tinggi yang bersangkutan.
4. Selambat-lambatnya 1 (satu) bulan setelah pelaksanaan diwajibkan mengirimkan 1 bundel laporan kepada Direksi PTPN IV cq Bagian SDM untuk dimasukkan ke dalam perpustakaan PTPN IV.
5. Yang bersangkutan agar berkoordinasi dengan Penanggung Jawab Riset di Unit Kerja yang menjadi tempat penelitian selama proses riset dilaksanakan.
6. Khusus bagi peserta Riset yang harus melakukan konfirmasi data riset dalam bentuk tatap muka ke unit kerja terkait, maka diwajibkan
   a. Menggunakan pakaian kemeja putih, bawahan hitam serta memakai jaket almamater dan sepatu.
   b. Membawa Surat Izin Riset dari PTPN IV, Surat Pernyataan Kesediaan dan Surat Pernyataan yang sudah dilengkapi dengan hasil pemeriksaan dokter.
7. Pelaksanaan kunjungan dalam betuk tatap muka ke unit kerja tempat pelaksanaan riset hanya dilakukan selama 1 (satu) hari, dan yang bersangkutan harus berprilaku sopan, mematuhi peraturan dan ketentuan protokol kesehatan yang berlaku di tempat pelaksanaan riset.
8. Surat keterangan selesai pelaksanaan riset dikeluarkan oleh Bagian/Distrik/Kebun/Pabrik dimana tempat pelaksanaan riset tersebut.
9. Apabila selama waktu pelaksanaan terjadi kecelakaan baik di dalam/di luar PTPN IV maka sepenuhnya menjadi tanggung jawab yang bersangkutan.
10. Bagi yang melanggar aturan tersebut, maka Perusahaan akan memberikan sanksi berupa dikeluarkan dari program riset.

GM/Manajer/Kepala Bagian yang menerima tembusan surat ini agar dapat membantu segala sesuatunya yang berkaitan dengan keperluan tersebut diatas, serta menjaga kerahasiaan data perusahaan.
Demikian disampaikan.

PT PERKEBUNAN NUSANTARA IV
Bagian Sumber Daya Manusia

Misran, SP
Kepala Bagian

Tembusan :
- UNIT USAHA TEH TANAMAN
- Mahasiswa/Siswa Ybs
(Email : febriadymarpaung@gmail.com) / (No.HP : 12345678910)

Lampiran 6 Surat Selesai Riset

UNIT USAHA TEH
PT PERKEBUNAN NUSANTARA IV
SIMALUNGUN-SUMATERA UTARA-INDONESIA

KANTOR UNIT USAHA : BAH BUTONG TELP : (0622) 25617
KANTOR PUSAT : JL. LETJEND SUPRAPTO NO. 2 MEDAN TELP : (061) 45773117

SURAT KETERANGAN
BUT/SK/ 08 / I / 2023

Yang bertanda tangan dibawah ini :

Nama : Hwin Dwi Putera
Jabatan : Manajer Unit Usaha Teh
Alamat : PT Perkebunan Nusantara IV Unit Usaha Teh

Menerangkan dengan sebenarnya bahwa Mahasiswa Fakultas Teknik Universitas Medan Area atas nama sbb :

NO	NAMA	Program Study	NIM
1	Febriady Marpaung	Informatika	188160032

Telah selesai melakukan Penelitian dan Pengambilan Data Tugas Akhir di PT.Perkebunan Nusantara IV Unit Usaha Teh Sidamanik, terhitung mulai tanggal 03 November 2022 s/d 03 Januari 2023 dengan judul Penelitian :

"Klasifikasi Tingkat Kematangan Pucuk Daun Teh menggunakan Least–Squares Support Vector Machine"

Demikian surat keterangan ini diperbuat untuk dapat dipergunakan seperlunya.

Sidamanik, 30 Januari 2023

PT Perkebunan Nusantara IV
Unit Usaha Teh

Hwin Dwi Putera
Manajer

AKHLAK - Amanah, Kompeten, Harmonis, Loyal, Adaptif, Kolaboratif.