

**IMPLEMENTASI METODE *CONVOLUTIONAL NEURAL*
NETWORK (CNN) UNTUK MENGLASIFIKASIKAN
PENYAKIT PADA KELAPA SAWIT**

SKRIPSI

PRISKO BANOZA

191401072



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

IMPLEMENTASI METODE CONVOLUTIONAL NEURAL
NETWORK (CNN) UNTUK MENGLASIFIKASIKAN
PENYAKIT PADA KELAPA SAWIT

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer

PRISKO BANOZA

191401072



PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

MEDAN

2024

PERSETUJUAN

Judul : IMPLEMENTASI METODE CONVOLUTIONAL
NEURAL NETWORK (CNN) UNTUK
MENGKLASIFIKASIKAN PENYAKIT PADA
KELAPA SAWIT

Kategori : SKRIPSI

Nama : PRISKO BANOZA

Nomor Induk Mahasiswa : 191401072

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI
INFORMASI UNIVERSITAS SUMATERA
UTARA

Telah diuji dan dinyatakan lulus di Medan,

Dosen Pembimbing II



Handrizal S.Si., M.Comp.Sc

NIP. 197706132017061001

Dosen Pembimbing I



Dr. T. Henny Febriani Harumy

S.kom., M.Kom

NIP. 198802192019032016

Diketahui/Disetujui Oleh
Ketua Program Studi S-1 Ilmu Komputer



Dr. Amalia, S.T., M.T

NIP. 197812212014042001

PERNYATAAN

IMPLEMENTASI METODE CONVOLUTIONAL NEURAL NETWORK (CNN) UNTUK MENGLASIFIKASIKAN PENYAKIT PADA KELAPA SAWIT

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan seumbernya.

Medan,



Prisko Banoza

191401072

UCAPAN TERIMA KASIH

Puji syukur kehadiran Tuhan Yang Maha Esa atas anugerah, berkat dan karunia yang telah dilimpahkan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini yang merupakan salah satu persyaratan untuk memperoleh gelar Sarjana Komputer dari program studi S-1 Ilmu Komputer Universitas Sumatera Utara.

Pada kesempatan ini penulis ingin memberikan ucapan terima kasih yang sebesar-besarnya kepada semua pihak yang telah mendukung dan membantu pengerjaan skripsi ini. Penulis mengucapkan terima kasih kepada:

1. Bapak Prof. Dr. Muryanto Amin, S.Sos., M.Si. Selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Ibu Dr. Amalia S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara dan selaku Dosen Pembimbing I yang membantu penulis dalam menyempurnakan skripsi ini.
4. Bapak Dr. Eng Ade Candra, S.T., M.Kom. Selaku Dosen Pembimbing Akademik penulis di Program Studi S-1 Ilmu Komputer.
5. Ibu Dr. T. Henny Febriana Harumy S.Kom., M.Kom. Selaku Dosen Pembimbing I yang telah membimbing penulis dalam penyusunan skripsi ini.
6. Bapak Handrizal S.Si., M.Comp.Sc selaku dosen Pembimbing II yang telah membimbing dan memberikan saran dan masukan kepada penulis dalam penyusunan skripsi ini.
7. Bapak Dr. Jos Timanta Tarigan S.Kom., M.Sc. selaku Dosen Pembimbing II yang membantu penulis dalam menyempurnakan skripsi ini.
8. Bapak/Ibu staff pengajar serta seluruh pegawai Program Studi S-1 Ilmu Komputer Fasilkom-TI USU.
9. Orang tua dan keluarga yang selalu mendoakan dan memberikan dukungan baik dari segi moral maupun material.

10. Sahabar-sahabat saya Annisa, Putri, Shinta, Shofia, Niko, Ivan, Sinu, Filbert, Christine, Harris, dan Vinny yang telah menemani penulis dalam proses penulisan skripsi ini.

11. Serta seluruh pihak yang terlibat langsung ataupun tidak langsung yang tidak dapat disebutkan secara rinci.

Demikianlah skripsi ini dibuat, semoga bermanfaat bagi pembaca pada umumnya dan kepada penulis khususnya, Penulis menyadari bahwa masih banyak terdapat kekurangan yang ada di dalam skripsi ini. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun dari semua pihak yang membaca skripsi ini dan untuk itu penulis mengucapkan terima kasih.

Medan

A handwritten signature in black ink, consisting of stylized, overlapping letters, positioned above a horizontal line.

Penulis

ABSTRAK

Tanaman kelapa sawit adalah sebuah tanaman produksi yang bisa memproduksi minyak kelapa sawit, yang mana ini merupakan salah satu sumber mata uang untuk menghidupi petani kelapa sawit dan keluarganya. Karenanya jika kelapa sawit terkena penyakit dan tidak dapat bertumbuh secara prima, maka penghasilan dari petani kelapa sawit tersebut bisa terancam dan berkurang. Penyakit yang sering menyerang kelapa sawit pada masa pertumbuhannya antara lain yaitu *Antraknosa*, Garis Kuning, dan bintik daun. Penyakit – penyakit ini meresahkan karena merusak daun yang mana daun merupakan cara bagi tanaman untuk berfotosintesis dan hidup. Salah satu cara mencegah agar kelapa sawit tidak mati ataupun tumbuh dengan penyakit adalah dengan mengetahui penyakit apa yang menjangkit kelapa sawit pada saat itu juga agar bisa dilakukan penanganan cepat. Oleh sebab itu dibutuhkan nya sistem yang dapat mengklasifikasikan yang sedang dijangkit oleh penyakit kelapa sawit. Penelitian yang dilakukan ini menggunakan data dari daerah perkebunan kelapa sawit di Kabupaten Mukomuko, provinsi Bengkulu. Data yang digunakan diambil menggunakan kamera *hanpphone*, data yang diambil yaitu sebanyak 300 data sawit dan 100 data kaggle, data lalu di *transpose*, *flip*, dan *roteate* sehingga data menjadi lebih banyak 4 kali lipat yaitu 1600 total data yang digunakan. Metode yang digunakan untuk mengklasifikan penyakit kelapa sawit adalah *Convolutional Neural Networks* (CNN) yang berfungsi untuk mengekstrasi fitur dari gambar daun kelapa sawit, dan *You Only Look Once* (YOLO) yang digunakan untuk mendeteksi object kelapa sawit di gambar. Data dibagi menjadi 4 kelas untuk dimasukkan ke algoritma CNN dengan masing masing kelas menggunakan 400 jumlah data, dan data dibagi menjadi 1120 atau 70% data *training* dan 480 atau 30% data validasi dengan total data 1600 dataset. Data dipilih dan diberi *bounding box* untuk dimasukkan ke dalam algoritma YOLO yang juga dibagi dengan pembagian yang sama. Hasil yang didapat dari evaluasi model adalah CNN dengan *accuracy* 0.9534, *Loss* 0.0980, *Val_accuracy* 0.9782, dan *Val_loss* 0.1085, dan YOLO dengan *mAP50* 0.992, dan *mAP50-95* 0.846.

Kata Kunci : *Antraknosa, Bintik Daun, Garis Kuning, Convolutioinal Neural Networks, You Only Look once, Machine Learning.*

*IMPLEMENTATION OF CONVOLUTIONAL NEURAL
NETWORK (CNN) METHOD TO CLASSIFY
DISEASES IN OIL PALM*

ABSTRACT

Oil palm is a production crop that can produce palm oil, which is one of the sources of currency to support oil palm farmers and their families. Therefore, if the oil palm is affected by disease and cannot grow in prime condition, the income of the oil palm farmer can be threatened and reduced. Diseases that often attack oil palms during their growth period include Anthracnose, Yellow Line, and leaf spots. These diseases are troubling because they damage the leaves which are a way for plants to photosynthesize and live. One way to prevent oil palms from dying or growing with disease is to know what diseases infect oil palms at that time so that quick treatment can be done. Therefore, a system is needed that can classify those that are infected by oil palm diseases. The research conducted uses data from oil palm plantation areas in Mukomuko Regency, Bengkulu province. The data used is taken using a cellphone camera, the data taken is 300 palm data and 100 kaggle data, the data is then transposed, flipped, and rotate so that the data becomes 4 times more, namely 1600 total data used. The method used to classify oil palm diseases is Convolutional Neural Networks (CNN) which functions to extract features from oil palm leaf images, and You Only Look Once (YOLO) which is used to detect oil palm objects in images. The data is divided into 4 classes to be entered into the CNN algorithm with each class using 400 amounts of data, and the data is divided into 1120 or 70% training data and 480 or 30% validation data with a total of 1600 datasets. Data is selected and given a bounding box to be entered into the YOLO algorithm which is also divided by the same division. The results obtained from model evaluation are CNN with accuracy 0.9534, Loss 0.0980, Val_accuracy 0.9782, and Val_loss 0.1085, and YOLO with mAP50 0.992, and mAP50-95 0.846.

Keywords: *Antraknosa, leaf galls, Patch Yellow, Convolutional Neural Networks, You Only Look once, Machine Learning.*

DAFTAR ISI

PERSETUJUAN.....	ii
PERNYATAAN.....	iii
UCAPAN TERIMA KASIH	iv
ABSTRAK	vi
ABSTRACT	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR.....	xi
DAFTAR TABEL	xii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Batasan Masalah	3
1.4. Tujuan Penelitian	3
1.5. Manfaat Penelitian	3
1.6. Metodologi Penelitian.....	3
1.7. Sistem Penulisan	4
BAB II LANDASAN TEORI	6
2.1. Machine Learning	6
2.1.1. Neural Network.....	6
2.1.2. Convolutional Neural Netwrok (CNN).....	6
2.1.2.1. Convolutional Layer	7
2.1.2.2 ReLu Layer	8
2.1.2.3 Fully Connected Layer.....	8
2.1.2.4 Softmax	9

2.2 YOLO	9
2.3. Android	10
2.4 Kelapa Sawit	10
2.4.1 Antraknosa	11
2.4.2 Bintik Daun	11
2.4.3 Garis Kuning (Patch yellow).....	12
BAB III ANALISIS DAN PERANCANGAN SISTEM.....	13
3.1 Penjelasan Sistem.....	13
3.2 Sumber Dataset	15
3.3 Pra-proses Gambar.....	15
3.4 Pelatihan model CNN	15
3.5 Object detection dengan YOLO.....	34
3.5 Evaluasi model.....	34
BAB IV IMPLEMENTASI DAN PENGUJIAN	36
4.1 Perangkat yang Digunakan	36
4.1.1. Perangkat Keras	36
4.1.2 Perangkat Lunak	36
4.1.3 Library.....	36
4.2 Penerapan Tahap Pengumpulan Dataset	37
4.3 Penerapan Tahap Pra-proses dataset.....	38
4.4 Penerapan Tahap Pelatihan untuk model CNN.....	39
4.5 Penerapan Tahap Pelatihan untuk Model YOLO	41
4.6 Penerapan Tahap Evaluasi	41
4.7 Penerapan ke Dalam Android	44
4.8 Pengujian.....	51
BAB V KESIMPULAN DAN SARAN.....	54

5.1 Kesimpulan	54
5.2 Saran	54
DAFTAR PUSTAKA.....	55
LAMPIRAN.....	58
BIODATA	69

DAFTAR GAMBAR

Gambar 2.1 Struktur CNN	7
Gambar 2.2 Convolutional Layer	7
Gambar 2.3 Contoh Operasi ReLu Layer	8
Gambar 2.4 Fully Connected Layer	8
Gambar 2.5 Contoh Antraknosa	11
Gambar 2.6 Contoh Bintik Daun.....	11
Gambar 2.7 Contoh Garis Kuning.....	12
Gambar 3.1 Arsitektur Umum Sistem.....	14
Gambar 4.1 Graph untuk setiap epoch	43
Gambar 4.2 Tampak awal aplikasi untuk memilih gambar.....	45
Gambar 4.3 Hasil Bukan Sawit	50
Gambar 4.4 Hasil Antraknosa	50
Gambar 4.5 Hasil Bintik Daun	50
Gambar 4.6 Hasil Garis Kuning	50
Gambar 4.7 Hasil Sehat.....	51

DAFTAR TABEL

Tabel 4.1 Contoh gambar <i>dataset</i>	36
Tabel 4.2 Contoh Gambar Sesudah dan Sebelum pra-proses	37
Tabel 4.3 Pengujian Tanaman	51

BAB I

PENDAHULUAN

1.1 Latar Belakang

Tanaman kelapa sawit merupakan tanaman penghasil minyak sawit yang banyak digunakan dalam industri pertanian komersial secara internasional ataupun lokal. Jenis Kelapa sawit *Elaeis guineensis* Jacq atau kelapa sawit afrika merupakan jenis kelapa sawit yang paling umum dibudidayakan di dunia terutama di Indonesia.

Dalam perekonomian Indonesia, Industri minyak sawit berperan sangat penting antara lain penghasil devisa terbesar, pendorong ekonomi rakyat, kedaulatan energi, dan penyediaan tenaga kerja. Sebab itu penurunan hasil tanaman sawit yang disebabkan oleh penyakit merupakan sebuah kerugian yang besar bagi Indonesia. Oleh karenanya penulis ingin membuat sebuah aplikasi yang dapat membantu petani sawit dalam mengelola tanaman sawit menggunakan android dan *Machine Learning*.

Machine Learning merupakan salah satu bidang yang termasuk dalam kecerdasan buatan dan berperan penting dalam berbagai aspek teknologi modern. Ini merupakan teknik yang memungkinkan sistem untuk menganalisis serta mengambil keputusan berdasarkan pola yang ditemukan dalam data secara sistematis. Dalam banyaknya algoritma *Machine Learning*, CNN merupakan algoritma yang banyak digunakan. Dalam penelitian ini, penulis memilih untuk menggunakan CNN karena algoritma ini dirancang khusus untuk mengolah data dalam bentuk gambar. CNN memiliki kemampuan untuk mengenali pola visual, mengekstrak fitur penting, serta meningkatkan akurasi dalam proses klasifikasi dan analisis citra. Oleh karena itu, pemanfaatan CNN dalam penelitian ini dianggap sebagai pendekatan yang paling tepat untuk mendapatkan hasil yang optimal dalam pengolahan data gambar.

Dalam penelitian yang dilakukan Errissya Rasywir, Rudolf Sinaga, dan Yoni Pratama (2020) dari pengujian penyakit kelapa sawit menggunakan 2490 citra menunjukkan kalau klasifikasi citra menggunakan CNN menghasilkan hasil yang cukup baik, serta telah menjadi tanda bahwa pengembangan sistem klasifikasi

penyakit sawit secara otomatis dan *mobile* dapat digunakan untuk membantu para petani sawit.

Pada penelitian yang dilakukan Imha Luchman, Theresia Wati, S.Kom., MTI. , dan Desta Sandya Prasvita, S.Kom., M.Kom. (2022) untuk mengklasifikasikan pohon kelapa sawit menggunakan citra *LiDAR* dengan *Convolutional Neural Network*. Studi ini memanfaatkan data yang diperoleh melalui teknologi penginderaan jauh *LiDAR* yang dikumpulkan oleh PT Asi Pudjiastuti Geosurvey. Data yang dihimpun berbentuk citra perkebunan kelapa sawit yang diambil menggunakan sensor *LiDAR*. Setelah proses akuisisi data selesai, dataset yang terkumpul kemudian dibagi menjadi tiga bagian utama dengan proporsi 64% untuk pelatihan model (*training*), 16% untuk validasi, dan 20% untuk pengujian (*testing*), di mana pembagiannya dilakukan secara acak. Hasil penelitian ini menunjukkan bahwa CNN memiliki kinerja yang sangat baik dalam mengklasifikasikan citra kelapa sawit dari data *LiDAR*. Selain itu, pemilihan parameter yang tepat berperan krusial dalam meningkatkan tingkat akurasi selama proses pelatihan dan pengujian. Penggunaan parameter yang optimal juga mampu mengurangi beban komputasi dalam tahap pembelajaran model, sehingga memungkinkan proses yang lebih efisien dan efektif.

Dalam penelitian yang dilakukan Dieki Irfansyah, Metty Mustikasari, dan Amat Suroso (2021) untuk mengklasifikasikan hama pada citra daun tanaman kopi menggunakan arsitektur *Convolutional Neural Network (CNN) Alexnet*. *AlexNet* adalah salah satu arsitektur yang berhasil meraih gelar juara dalam ajang *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*, sebuah kompetisi berskala besar di bidang klasifikasi citra yang diselenggarakan pada tahun 2012. Penelitian ini memanfaatkan data citra daun kopi yang diperoleh dari sumber berikut: <https://data.mendeley.com/datasets/c5yvn32dzg/2>. Tiga kategori yang digunakan dalam penelitian ini meliputi *health*, *red spider mite*, dan *rust*. Kategori *health* mengacu pada daun kopi dalam kondisi sehat, sedangkan *rust* dan *red spider mite* mengindikasikan daun yang mengalami serangan hama atau penyakit. Hasil penelitian menunjukkan bahwa arsitektur CNN *AlexNet* dapat diimplementasikan secara efektif menggunakan platform MATLAB. Model yang dikembangkan mampu mencapai tingkat akurasi klasifikasi sebesar 0.816 pada data uji, sehingga

dapat disimpulkan bahwa arsitektur CNN AlexNet memiliki tingkat ketepatan yang tinggi dalam mengidentifikasi keberadaan hama pada daun kopi.

1.2. Rumusan Masalah

Kelapa sawit merupakan salah satu tanaman yang penting bagi sebagian masyarakat Indonesia. Jika kelapa sawit sakit dan masa panen terhambat, petani sawit akan mengalami kerugian. Oleh karena itu dibutuhkan aplikasi yang bisa mendeteksi penyakit kelapa sawit.

1.3. Batasan Masalah

Dalam Penelitian ini terdapat beberapa batasan, yaitu:

1. Penyakit yang diangkat hanya penyakit *Antracnosa*, penyakit bercak daun, dan penyakit garis kuning (*patch yellow*).
2. Sistem diimplementasikan di platform android.
3. *Input* yang dimasukkan oleh *user* berupa *image*.
4. *Output* yang dikeluarkan hanya berupa nama penyakit saja.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membuat aplikasi yang bisa mengklasifikasikan penyakit dari kelapa sawit menggunakan gambar dari daun kelapa sawit yang sedang terjangkit oleh penyakit.

1.5. Manfaat Penelitian

Penelitian ini dapat membantu para petani sawit untuk dapat mendeteksi penyakit yang sedang menjangkit tanaman kelapa sawit mereka secara cepat dan tepat, agar membantu dalam penurunan resiko terhambatnya panen kelapa sawit.

1.6. Metodologi Penelitian

Untuk metodologi penelitian yang digunakan yaitu sebagai berikut :

1. Studi Pustaka

Pada tahap ini, penelitian diawali dengan mengumpulkan berbagai data dari sumber-sumber yang kredibel. Selain itu, dilakukan kajian literatur melalui

berbagai referensi, seperti buku, *e-book*, artikel ilmiah, makalah, serta situs *web* yang membahas tentang tanaman kelapa sawit, algoritma *Convolutional Neural Network* (CNN), algoritma YOLO, dan pengembangan aplikasi berbasis Android.

2. Analisis dan Perancangan Sistem

Dilihat dari ruang lingkup penelitian, analisa dilakukan oleh penulis terhadap apa saja yang akan dibutuhkan dalam penelitian untuk segera dirancang dalam sebuah diagram alir (*flowchart*).

3. Implementasi

Pada tahap ini, penulis mengembangkan sebuah sistem dengan memanfaatkan bahasa pemrograman *Python* yang terintegrasi dengan *platform* Android, sesuai dengan diagram alir yang telah disusun sebelumnya.

4. Pengujian

Pada tahap ini, sistem yang telah dirancang akan diuji coba untuk mengetahui apakah sistem berjalan sesuai tujuan.

5. Dokumentasi

Tahap dokumentasi ini penulis melakukan dokumentasi yang diawali dari tahap analisis menuju kepada tahap pengujian sistem dengan format penulisan skripsi.

1.7. Sistem Penulisan

Sistematika penulisan skripsi ini dibagi dalam beberapa bab, yaitu:

BAB I PENDAHULUAN

Bab pendahuluan ini mengulas berbagai aspek penting dalam penelitian, termasuk latar belakang, perumusan masalah, batasan penelitian, tujuan yang ingin dicapai, manfaat penelitian, metode yang digunakan, serta sistematika penulisan skripsi.

BAB II LANDASAN TEORI

Bab landasan teori ini membahas mengenai penjelasan mengenai kelapa sawit, *Android*, *Machine Learning*, *YOLO*, *Convolutional Neural Network* (CNN), *Neural Network*, dan penelitian – penelitian yang relevan.

BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab analisis dan perancangan sistem ini membahas mengenai analisis dari masalah – masalah penelitian dan solusi yang diberikan untuk permasalahan yang ada dengan sistem yang akan dibangun.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab implementasi dan pengujian sistem ini membahas mengenai implementasi dari sistem yang telah dirancang dan hasil dari pengujian sistem yang berfungsi untuk memeriksa kesesuaian dan keberhasilan sistem.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang diperoleh dari hasil penelitian yang telah dilakukan serta saran berdasarkan hasil pengujian. Saran yang diberikan diharapkan dapat menjadi referensi atau masukan bagi penelitian selanjutnya agar dapat dikembangkan lebih lanjut pada penelitian selanjutnya.

BAB II

LANDASAN TEORI

2.1. Machine Learning

Machine learning merupakan bidang studi yang mempelajari algoritma dan model statistik yang memungkinkan sistem komputer untuk menyelesaikan tugas tertentu tanpa memerlukan pemrograman secara langsung. Teknologi ini banyak diterapkan dalam kehidupan sehari-hari. Misalnya, setiap kali kita menggunakan mesin pencari seperti *Google* untuk menjelajah internet, salah satu alasan mesin pencari dapat berfungsi dengan baik adalah berkat algoritma *machine learning* yang telah mempelajari cara menentukan peringkat situs *web* berdasarkan kata kunci yang dimasukkan. Algoritma *machine learning* ini digunakan untuk berbagai keperluan, seperti analisis data, pemrosesan gambar, prediksi, dan masih banyak lagi.. Keuntungan terbesar dari *machine learning* adalah setelah sistem mempelajari algoritma apa yang harus dilakukan maka pekerjaan bisa dilakukan secara otomatis. *Machine Learning* merupakan salah satu topik dari kecerdasan buatan.

2.1.1. Neural Network

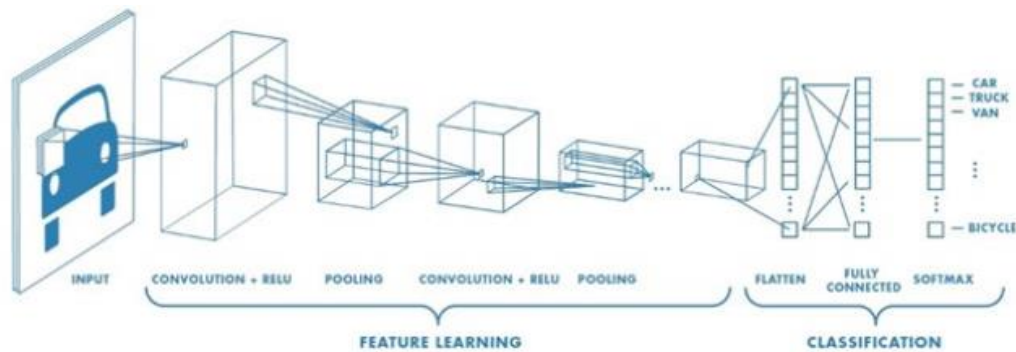
Neural Network adalah salah satu metode dalam kecerdasan buatan yang mengadopsi konsep jaringan saraf manusia. Model ini dibangun dari kumpulan node yang saling terhubung, mencerminkan cara kerja otak manusia. Pendekatan ini dirancang untuk meniru karakteristik otak, seperti pemrosesan paralel, jumlah elemen pemrosesan yang besar, serta ketahanan terhadap kesalahan (*fault tolerance*).

2.1.2. Convolutional Neural Network (CNN)

Convolutional Neural Network secara fungsi mirip dengan Jaringan Saraf Tiruan (JST) tradisional yang mana keduanya terdiri dari neuron yang dapat mengoptimasi secara mandiri melalui proses pembelajaran atau *learning*. Perbedaan antara CNN dan JST tradisional adalah CNN sering digunakan pada bidang pengenalan pola pada citra. CNN pada saat ini sangat banyak digunakan untuk

melakukan penelitian yang berkaitan dengan *object detection* ataupun *image classification*. Banyak penelitian terdahulu yang menunjukkan bahwa CNN merupakan metode dengan akurasi yang sangat tinggi.

Arsitektur LeNet5 menunjukkan bahwa ada 4 jenis *layer* utama pada sebuah CNN yaitu, *convolutional layer*, *relu layer*, *subsampling layer* atau *pooling layer*, dan *fully connected layer* (Yuliany, S. , Aradea, & Rachman, A., N. 2022). Gambar 2.1 dibawah adalah gambar Struktur CNN.

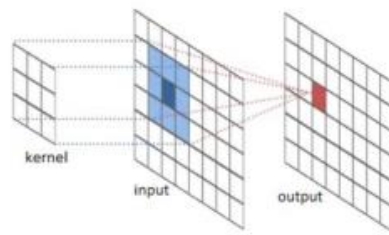


Gambar 2.1 Struktur CNN

2.1.2.1. Convolutional Layer

Lapisan konvolusi (*convolutional layer*) adalah fondasi utama dalam struktur arsitektur CNN. Dalam konteks pemrosesan citra, konvolusi merujuk pada penerapan kernel pada setiap posisi yang memungkinkan dalam citra. Di sebagian besar *library* yang menggunakan CNN, konvolusi sering disebut sebagai *cross-correlation*, yang pada dasarnya juga merupakan bentuk konvolusi, namun tanpa proses pembalikan kernel. (Asrianda, Aidilof, H., A., & Pangetsu, Y. 2021). Persamaan dan contoh dari operasi konvolusi dapat dilihat dibawah. Pada gambar 2.2 dibawah, bisa kita lihat cara kerja *Convolutional Layer*.

$$s(t) = (x * w)(t)$$

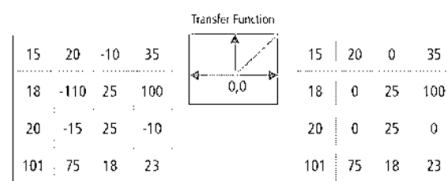


Gambar 2.2 *Convolutional Layer*

2.1.2.2 ReLu Layer

Lapis *Rectified Linear Unit* (ReLU) merupakan sebuah lapisan yang ditujukan sehingga *nonlinearitas* fungsi keputusan bisa meningkat (Alamsyah, Derry & Pratama, Dicky. 2020). Digunakan fungsi aktivasi sebagai dibawah. Gambar 2.3 dibawah menunjukkan contoh operasi *ReLU Layer*

$$f(x) = \max(0, x)$$



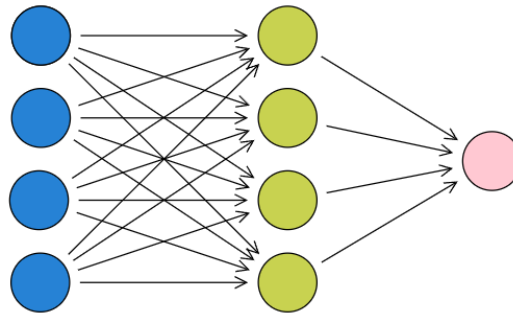
Gambar 2.3 Contoh operasi *ReLU layer*

2.1.2.3 Fully Connected Layer

Fully Connected Layer adalah bentuk *multi-layer* perceptron klasik yang menerapkan fungsi aktivasi *softmax* pada lapisan *outputnya*. Sebutan "*Fully Connected*" mengacu pada kondisi di mana setiap *neuron* di lapisan sebelumnya terhubung secara langsung dengan seluruh *neuron* di lapisan selanjutnya. (Muhammad, S. & Wibowo, A., T. 2021).

Output dari *convolutional* dan *pooling layer* memprementasikan *high level features* yang mana yaitu hasil ekstrasi dari input gambar. Pada dasarnya *layer* ini mendapatkan *volume* input (*convolutional*, *ReLU* atau *pooling Layer*) serta menghasilkan *vector* dimensi N yang mana N adalah jumlah kelas, tujuan dari menggunakan fitur-fitur tersebut adalah untuk mengklasifikasikan gambar kedalam kelas-kelas yang mana berdasarkan inisiani pada data *training* (Muhammad, S. &

Wibowo, A., T. 2021). Gambar 2.4 dibawah adalah Contoh *Fully Connected Layer*



Gambar 2.4 *Fully Connected Layer*

2.1.2.4 *Softmax*

Softmax adalah fungsi aktivasi yang digunakan sehingga mendapatkan hasil kelas. Nilai kelas dihitung menggunakan fungsi aktivasi *softmax* (Asrianda, Aidilof, H., A., & Pangetsu, Y. 2021) .

Keuntungan yang paling terbaik untuk softmax digunakan adalah output yang dihasilkan menggunakan rentang probabilitas dari 0 hingga 1, serta jika setiap kemungkinan kelas kelasnya ditambahkan maka sama dengan 1. Dalam kasus model *multi-classification*, *softmax* dapat mengembalikan probabilitas pada setiap kelas dan kelas tertentu akan memiliki probabilitas tertinggi (Muhammad, S. & Wibowo, A., T. 2021). Fungsi aktivasi *softmax* dapat ditulis sebagai berikut

$$S(y) = \frac{e^y}{\sum e^y}$$

2.2 *YOLO*

Yolo merupakan sistem pendeteksian objek yang baru yang bisa digunakan secara *real time* atau pun tidak. *YOLO* merupakan sebuah sistem yang menggunakan sebuah jaringan syaraf tunggal (*single neural network*) untuk bisa melakukan pengenalan dan pendeteksian objek yang memprediksi menggunakan *bounding box* dan kelas probability(Dwiyanto, R., Widodo, D. W., & Kasih, P. 2022).

Yolo Memanfaatkan *single neural network* pada arsitekturnya yang mana itu digunakan untuk menggabungkan proses klasifikasi yang mana pada model-model

lainnya dilakukan secara terpisah. Dengan menggunakan algoritma seperti ini Yolo untuk pengenalan objek menjadi lebih cepat dari pada model-model lainnya sehingga YOLO mampu mengalahkan model-model lainnya (Kusunah, H., Zahran, M. S., Rifqi, K. N., Putri, D. A., & Hapsari, E. M. W. 2023).

2.3. Android

Android adalah sistem operasi berbasis *Linux* dengan kode sumber terbuka yang menggunakan lisensi APACHE 2.0, dirancang khusus untuk perangkat bergerak dengan layar sentuh, seperti *smartphone* dan *tablet*. Keunggulan terbaik pada Android terletak di kode sumber terbukanya, yang memungkinkan para pengembang perangkat lunak untuk dengan bebas memodifikasi dan mendistribusikan aplikasi yang mereka buat.

Pada zaman sekarang penggunaan *smarthphone* sudah menjadi hal yang wajib untuk hampir segala sesuatu yang kita lakukan, di Indonesia sendiri penggunaan *smarthphone* sangat tinggi dan 88.44% dari *smartphone* yang beredar di pasar Indonesia menggunakan sistem operasi Android, oleh karena itu Android dikenal oleh seluruh warga Indonesia dan hampir setiap orang dari berbagai kalangan memilikinya

2.4 Kelapa Sawit

Tanaman kelapa sawit merupakan tanaman penghasil minyak sawit yang banyak digunakan dalam industri pertanian komersial secara internasional ataupun lokal. Tanaman kelapa sawit sendiri memiliki dua spesies anggota, *Elaeis guineensis* dan *Elaeis oleifera*. Jenis Kelapa sawit *Elaeis guineensis* Jacq atau kelapa sawit afrika merupakan jenis kelapa sawit yang paling umum dibudidayakan di dunia terutama di Indonesia. Dalam perekonomian Indonesia, Industri minyak sawit berperan sangat penting antara lain penghasil devisa terbesar, pendorong ekonomi rakyat, kedaulatan energi, dan penyediaan tenaga kerja. Banyak penyakit yang dapat menggunakan perkembangan dan hasil panen dari kelapa sawit, penyakit yang di angkat oleh penulis adalah penyakit pada daun kelapa sawit yaitu *antraknosa*, bintik daun, dan Garis Kuning.

2.4.1 Antraknosa

Penyakit *Antraknosa* pada tanaman kelapa sawit disebabkan oleh infeksi jamur, dengan beberapa jenis jamur yang terlibat, seperti *Glomerella Cingulata*, *Botryodiplodia palmarum*, dan *Melanconium sp.* Penyakit ini dapat menyerang bagian daun serta tulang daun pada tanaman kelapa sawit. (Isnaini, Siregar, I. K., & Ihsan, M. 2022). Gambar 2.5 dibawah adalah contoh penyakit *Antraknosa*.



Gambar 2.5 Contoh *Antraknosa*

2.4.2 Bintik Daun

Penyakit bintik daun pada kelapa sawit telah berkembang menjadi ancaman signifikan bagi kelangsungan pembibitan tanaman kelapa sawit. Dalam dua tahun terakhir, penyakit ini telah menyebabkan kerusakan parah di sejumlah pusat pembibitan kelapa sawit di Kalimantan dan Sumatera, bahkan mengakibatkan kematian pada bibit kelapa sawit yang sedang dibudidayakan. Di Indonesia, penyakit bintik daun pada pembibitan kelapa sawit umumnya disebabkan oleh jamur dari genus *Culvularia*. (Priwiratama, H., Eris, D. D., Pradana, M. G., & Rozziansha, T. A. P. 2023). Gambar 2.6 dibawah adalah contoh sawit yang terkena bintik daun.



Gambar 2.6 Contoh Bintik daun

2.4.3 Garis Kuning (Patch yellow)

Penyakit garis kuning pada kelapa sawit disebabkan oleh jamur *Fusarium oxysporum*. Jamur ini cenderung menyerang tanaman yang sedang dalam kondisi lemah, yang biasanya dipicu oleh faktor-faktor seperti kekeringan, kekurangan nutrisi, paparan sinar matahari berlebihan, atau beban buah yang terlalu banyak. (Effendi, Z., Manurung, S., & Ayu, S., M. 2020). Penyakit ini meresahkan karena menyerang bagian daun yang mana bagian daun penting bagi tanaman untuk berfotosintesis yang dibutuhkan oleh tanaman untuk membuat nutrisi – nutrisi bagi mereka untuk hidup dan tumbuh. Gambar 2.7 dibawah adalah contoh sawit yang terkena penyakit garis kuning.



Gambar 2.7 Contoh garis kuning

BAB III

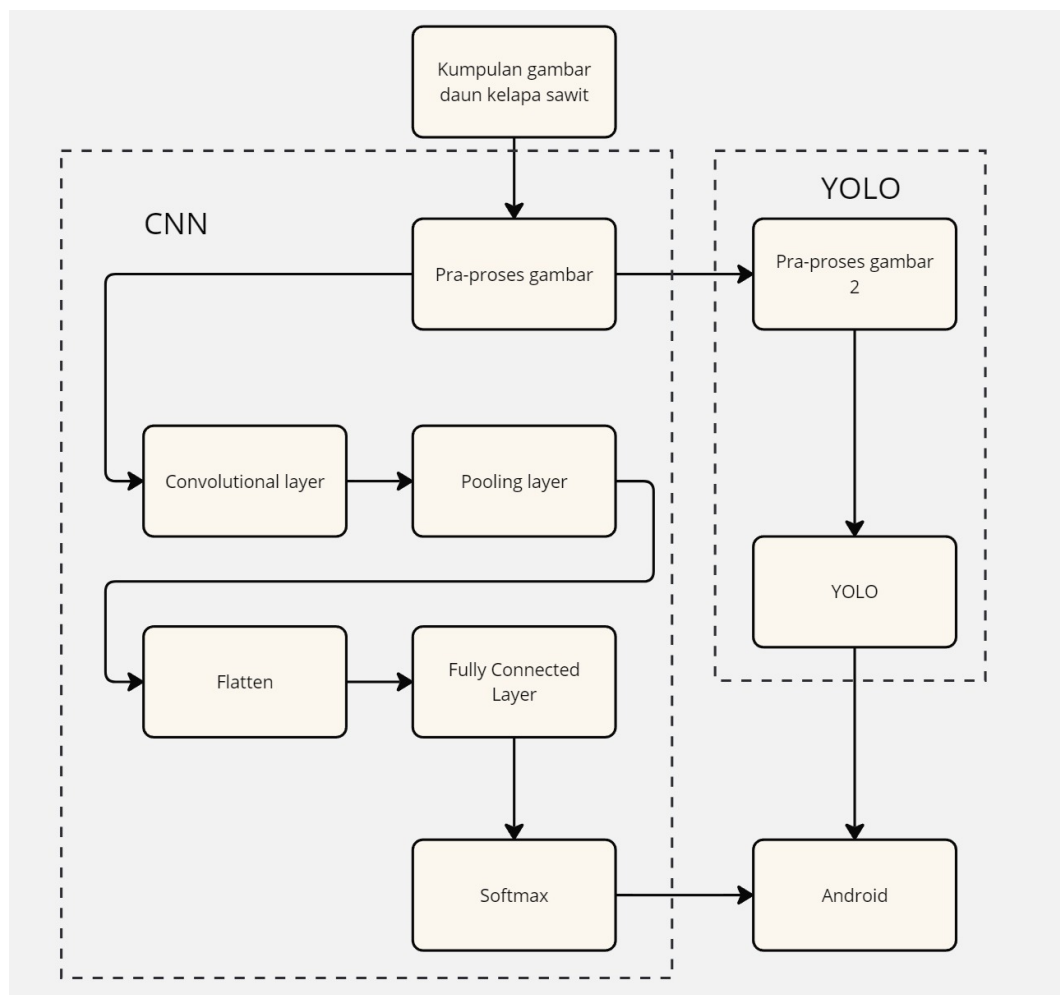
ANALISIS DAN PERANCANGAN SISTEM

3.1 Penjelasan Sistem

Pada penelitian ini, penulis mengembangkan suatu sistem yang mana sistem ini digunakan untuk mengklasifikasikan jenis penyakit pada kelapa sawit dengan cara menganalisis *image* dari daun kelapa sawit yang diberikan. Sistem dikembangkan dengan menggunakan algoritma *Convolutional Neural Network* (CNN) untuk mengklasifikasikan jenis penyakit nya, dan algoritma *You Only Look Once* (YOLO) untuk mendeteksi apakah *image* yang di berikan oleh user adalah sebuah gambar dari daun kelapa sawit atau bukan. Model CNN dan YOLO akan dilatih menggunakan gambar yang telah di pra-proses sebelumnya dengan metode *resize*.

Penelitian ini dimulai dengan mengumpulkan data-data yang berupa gambar daun kelapa sawit dari sawit di perkebunan di Kota Mukomuko. Gambar yang dikumpulkan berupa gambar daun kelapa sawit yang memiliki penyakit bintik daun, penyakit kuning daun, penyakit *Antraknosa* , dan daun kelapa sawit yang sehat. Gambar yang sudah dikumpulkan dilakukan pra-proses *resize* dan *crop* menjadi gambar berukuran 500 x 500 agar seragam yang kemudia dataset disimpan didalam google drive dan diakses menggunakan google *colaboratory*. Dataset lalu dipisahkan menjadi data untuk *training* dan data untuk *validation*. Setelah dataset dikumpulkan, model CNN mulai menerima dataset yang akan diproses oleh *Convolutional Layer*, yang mana pada proses ini *convlutional layer* membagi gambar menjadi beberapa bagian menggunakan filter kernel yang akan melewati seluruh bagian gambar, pada proses ini akan mengambil fitur-fitur penting pada gambar. Kemudia *pooling layer* akan mengurangi ukuran *feature map* dengan menggunakan *max pooling*. Setelah melakukan proses ini beberapa kali data akan masuk ke proses *flatten* untuk dirubah kedalam bentuk yang bisa dimasukkan ke dalam *fully connected layer* yang mana pada proses ini, data akan digunakan untuk membuat keputusan akhir dalam klasifikasi dengan bantuan aktivasi *softmax*.

Proses berikutnya adalah untuk membuat model *object detection* yang mana yang akan membedakan gambar yang *diinput* kan oleh user apakah sebuah daun kelapa sawit atau bukan. Pembuatan model menggunakan algoritma *YOLO*, pembuatan model dimulai dengan mengambil data dari dataset drive lalu gambar yang diambil di pra-proses sehingga gambar yang memiliki *object* daun sawit didalamnya di edit sehingga memiliki kotak yang mana menandakan *object* didalam kotak tersebut adalah sebuah daun kelapa sawit. Setelah itu gambar yang sudah di pra-proses akan di masukkan kedalam algoritma *YOLO* yang sudah disiapkan untuk dilatih sehingga bisa membedakan yang mana gambar memiliki *object* daun kelapa sawit atau tidak. Setelah ke dua model selesai di latih, model – model tersebut akan dimasukkan kedalam android untuk di implementasikan. Arsitektur umum sistem bisa dilihat pada gambar 3.1



Gambar 3.1 Arsitektur umum sistem

3.2 Sumber Dataset

Dataset yang digunakan merupakan gambar – gambar yang diambil penulis di perkebunan sawit di Kota Mukomuko menggunakan *handphone* yang diambil langsung dengan kamera bawaan *handphone* dan beberapa diambil dari *website kaggle*. Dataset yang diambil yaitu gambar dari daun sawit yang terjangkit oleh penyakit *Antraknosa*, Kuning daun, dan bintik daun dengan jumlah gambar yang diambil yaitu 300 data gambar sendiri, dan 100 dari data *kaggle* untuk daun sawit yang sehat. Dataset diambil berdasarkan pengawasan ahli yaitu bapak Samsudin, seorang pensiunan Ahli Pengendalian Hama Terpadu (APHT) di Kabupaten Mukomuko

3.3 Pra-proses Gambar

Gambar – gambar yang sudah diambil lalu di pra-proses sebelum dimasukkan untuk dilatih kedalam model agar data yang dilatih oleh model seragam. Pra-proses dilakukan dengan *me-resize* dan *me-crop* gambar agar menjadi 500 x 500 pixel serta gambar ditambah menjadi 4 kali lipat dengan cara setiap gambar yang digunakan untuk dijadikan dataset di *transpose*, *flip*, dan *rotate*. Untuk algoritma YOLO gambar dipilih kembali dan di edit dengan *bounding box* untuk menandai bahwa gambar memiliki *object* daun kelapa sawit pada gambar. Labelisasi dilakukan secara manual oleh penulis dengan mengelompokkan sesuai dengan kelas masing – masing sesuai arahan dari *expert* yaitu bapak samsudin.

3.4 Pelatihan model CNN

Selanjutnya gambar yang sudah di pra-proses akan diambil masuk ke dalam algoritma CNN untuk dilatih agar menjadi model yang bisa digunakan. Pertama gambar masuk ke *convolutional layer* dimana pada *layer* ini fitur – fitur penting pada gambar di ekstraksi. Lalu setelah dari *convolutional layer* gambar masuk ke *pooling layer* dimana pada lapisan ini akan mengurangi ukuran *feature map* agar membuat model lebih efisien. Setelah dilakukan proses *convolutional* dan *pooling* beberapa kali, gambar lalu masuk ke *flatten layer* yang mana pada lapisan ini

gambar dirubah dari *output* 2D/3D menjadi vektor 1D yang mana bisa dimasukkan ke dalam *fully connected layer* yang membutuhkan input 1D, setelah itu gambar memasuki aktivasi *softmax* yang mana dengan aktivasi ini yang menentukan probabilitas kelas dari gambar dan data yang di *training*.

Model yang akan kita gunakan merupakan model CNN yang disusun menggunakan arsitektur *LeNet* dengan 5 *convolutional layer* dengan fungsi aktivasi *relu*, 4 *pooling layer*, *flatten layer*, 2 *dense layer* yang memiliki fungsi aktivasi *relu* dan *softmax*, dan 1 *dropout layer*. Proses *training* ini pun dilakukan sekian kali sesuai epoch yang kita tentukan. Contoh perhitungan sederhana terhadap model CNN ini bisa kita buat sebagai berikut. Dengan menggunakan arsitektur *LeNet* kita akan menyusun kode kita seperti ini:

```
tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(256, 256, 3)),
tf.keras.layers.MaxPooling2D(2, 2),
tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(4, activation='softmax')
```

untuk mempermudah perhitungan kita kan menggunakan *input* 5x5 yang disetiap kali konvolusional kita akan menambah ukurannya, hal ini diperlukan karena kita memerlukan matrik untuk dihitung sebab kita tidak menggunakan *padding* pada kode kita jadi output yang dihasilkan akan mengecil. Misalkan kita memiliki *image input* yaitu

1	0	1	2	1
0	2	0	0	2
2	0	1	2	1
1	2	0	2	1
0	1	0	2	1

Dan kita memiliki kernel *filter* yaitu

1	0	1
0	1	0
1	0	1

Dengan menggunakan rumus

$$f(i,j) = (I * K)_{[i,j]} = \sum_x^h \sum_y^w K_{[x,y]} I_{[i-x,j-y]}$$

Yang mana f merupakan *output*, I adalah matrik *input*, K merupakan *kernel filter*, i dan j merupakan posisi pada *output*, h merupakan tinggi *input*, dan w merupakan lebar dari *input*. Dengan ini kita bisa merubah *input* dan *filter* yang kita punya menjadi

		-2	-1	0	1	2			
-2		1	0	1	2	1			
-1		0	2	0	0	2			
0		2	0	1	2	1			
1		1	2	0	2	1			
2		0	1	0	2	1			
		I							

*

		-1	0	1	
-1		1	0	1	
0		0	1	0	
1		1	0	1	
		K			

=

		-1	0	1	
-1					
0					
1				5	
		f			

$$f(i,j) = (I * K)_{[i,j]} = \sum_x^h \sum_y^w K_{[x,y]} I_{[i-x,j-y]}$$

$$f(1,1) = \sum_{x=-2}^5 \sum_{y=-2}^5 K_{[x,y]} I_{[i-x,j-y]}$$

$$\begin{aligned} f(1,1) = & K_{[-2,-2]} I_{[3,3]} + K_{[-1,-2]} I_{[2,3]} + K_{[0,-2]} I_{[1,3]} + K_{[1,-2]} I_{[0,3]} + \\ & K_{[2,-2]} I_{[-1,3]} + K_{[-2,-1]} I_{[3,2]} + K_{[-1,-1]} I_{[2,2]} + K_{[0,-1]} I_{[1,2]} + K_{[1,-1]} I_{[0,2]} + \\ & K_{[2,-1]} I_{[-1,2]} + K_{[-2,0]} I_{[3,1]} + K_{[-1,0]} I_{[2,1]} + K_{[0,0]} I_{[1,1]} + K_{[1,0]} I_{[0,1]} + \\ & K_{[2,0]} I_{[-1,1]} + K_{[-2,1]} I_{[3,0]} + K_{[-1,1]} I_{[2,0]} + K_{[0,1]} I_{[1,0]} + K_{[1,1]} I_{[0,0]} + \\ & K_{[2,1]} I_{[-1,0]} + K_{[-2,2]} I_{[3,-1]} + K_{[-1,2]} I_{[2,-1]} + K_{[0,2]} I_{[1,-1]} + K_{[1,2]} I_{[0,-1]} + \\ & K_{[2,2]} I_{[-1,-1]} = 0 + 0 + 0 + 0 + 0 + 0 + (1 * 1) + (0 * 1) + (1 * \\ & 1) + 0 + 0 + (0 * 2) + (1 * 2) + (0 * 2) + 0 + 0 + (1 * 0) + (0 * 0) + (1 * \\ & 1) + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 1 + 0 + 1 + 0 + 2 + 0 + 0 + 0 + 1 = 5 \end{aligned}$$

Dari rumus diatas kita bisa melihat perkalian yang dihasilkan adalah seperti dibawah, bisa dilihat pada warna di matrik bawah, bahwa perhitungan dari rumus diatas sama dengan perkalian perkalian dari *input* dikalikan dengan rotasi 180° dari kernel *filter*. Jadi untuk kedepannya kita akan menghitung menggunakan

$$f(i,j) = (I * K)_{[i,j]} = \sum_x^h \sum_y^w K_{[x,y]} ROT\ 180^\circ I_{[i+x,i+x]}$$

h = tinggi kernel

w = lebar kernel

x = - (h-1)/2

y = - (w-1)/2

	-2	-1	0	1	2
-2	1	0	1	2	1
-1	0	2	0	0	2
0	2	0	1	2	1
1	1	2	0	2	1
2	0	1	0	2	1

I

* Rot 180°

	-1	0	1
-1	1	0	1
0	0	1	0
1	1	0	1

K

=

	-1	0	1
-1	7	4	4
0	1	7	5
1	5	5	5

f

	-2	-1	0	1	2
-2	1	0	1	2	1
-1	0	2	0	0	2
0	2	0	1	2	1
1	1	2	0	2	1
2	0	1	0	2	1

I

*

	-1	0	1
-1	1	0	1
0	0	1	0
1	1	0	1

K

=

	-1	0	1
-1	7	4	4
0	1	7	5
1	5	5	5

f

$$f(1,1) = K[-1,-1]I[0,0] + K[-1,0]I[0,1] + K[-1,1]I[0,2] + K[0,-1]I[1,0] \\ + K[0,0]I[1,1] + K[0,1]I[1,2] + K[1,-1]I[2,0] + K[1,0]I[2,1] \\ + K[1,1]I[2,2]$$

$$= 1 * 1 + 2 * 0 + 1 * 1 + 0 * 0 + 2 * 1 + 1 * 0 + 0 * 1 + 2 * 0 \\ + 1 * 1 = 1 + 0 + 1 + 0 + 2 + 0 + 0 + 0 + 1 = 5$$

$$f(1,0) = 0 * 1 + 1 * 0 + 2 * 1 + 2 * 0 + 0 * 1 + 2 * 0 + 1 * 1 + 0 * 0 + 2 * 1 = \\ 0 + 0 + 2 + 0 + 0 + 0 + 1 + 2 = 5$$

$$f(1,-1) = 2 * 1 + 0 * 0 + 1 * 1 + 1 * 0 + 2 * 1 + 0 * 0 + 0 * 1 + 1 * 0 + 0 * 1 \\ = 2 + 0 + 1 + 0 + 2 + 0 + 0 + 0 + 0 = 5$$

$$f(0,1) = 0 * 1 + 0 * 0 + 2 * 1 + 1 * 0 + 2 * 1 + 1 * 0 + 0 * 1 + 2 * 0 + 1 * 1 \\ = 0 + 0 + 2 + 0 + 2 + 0 + 0 + 0 + 1 = 5$$

$$f(0,0) = 2 * 1 + 0 * 0 + 0 * 1 + 0 * 0 + 1 * 1 + 2 * 0 + 2 * 1 + 0 * 0 + 2 * 1 \\ = 2 + 0 + 0 + 0 + 1 + 0 + 2 + 0 + 2 = 7$$

$$f(0,-1) = 0 * 1 + 2 * 0 + 0 * 1 + 2 * 0 + 0 * 1 + 1 * 0 + 1 * 1 + 2 * 0 + 0 * 1 \\ = 0 + 0 + 0 + 0 + 0 + 0 + 1 + 0 + 0 = 1$$

$$f(-1,1) = 1 * 1 + 2 * 0 + 1 * 1 + 0 * 0 + 0 * 1 + 2 * 0 + 1 * 1 + 2 * 0 + 1 * 1 = 1 + 0 + 1 + 0 + 0 + 0 + 1 + 0 + 1 = 4$$

$$f(-1,0) = 0 * 1 + 1 * 0 + 2 * 1 + 2 * 0 + 0 * 1 + 0 * 0 + 0 * 1 + 1 * 0 + 2 * 1 = 0 + 0 + 2 + 0 + 0 + 0 + 0 + 0 + 2 = 4$$

$$f(-1,-1) = 1 * 1 + 0 * 0 + 1 * 1 + 0 * 0 + 2 * 1 + 0 * 0 + 2 * 1 + 0 * 0 + 1 * 1 = 1 + 0 + 1 + 0 + 2 + 0 + 2 + 0 + 1 = 7$$

Setelah kita mendapatkan perhitungan menggunakan konvolusi kita akan lanjut ke fungsi aktivasi yang kita gunakan, yaitu ReLu dengan rumus $f(x) = \max(0, x)$ yang mana semua hasil negatif kita rubah menjadi nol.

$$f(x) = \max(0, x)$$

$$f(1,1) = \max(0,5) = 5$$

$$f(1,0) = \max(0,5) = 5$$

$$f(1,-1) = \max(0,5) = 5$$

$$f(0,1) = \max(0,5) = 5$$

$$f(0,0) = \max(0,7) = 7$$

$$f(0,-1) = \max(0,1) = 1$$

$$f(-1,1) = \max(0,4) = 4$$

$$f(-1,0) = \max(0,4) = 4$$

$$f(-1,-1) = \max(0,7) = 7$$

Setelah itu kita akan melakukan *max pooling* pada hasil sebelumnya, *max pooling* ditentukan dengan menarik angka yang paling besar diantara dimensi yang sudah ditentukan, yang mana dalam kasus ini yaitu 2 x 2, bisa kita lihat contoh max pooling dibawah.

7	4	4
1	7	5
5	5	5

7	

7	4	4
1	7	5
5	5	5

7	7

7	4	4
1	7	5
5	5	5

7	7
7	

7	4	4
1	7	5
5	5	5

7	7
7	7

Setelah kita mendapat hasilnya, kita akan melanjutkan kembali ke *layer* kita yang berikutnya yaitu konvolusional *layer* yang kedua. Matrix *input* akan ditambah supaya menjadi lebih mudah untuk dihitung.

7	7	8	9	4
7	7	2	8	4
7	8	9	5	7
8	9	3	1	5
1	7	5	6	7

Diatas adalah matrix yang akan kita hitung untuk *layer* kita selanjutnya, perhitungan yang akan kita lakukan adalah seperti berikut

	-2	-1	0	1	2
-2	7	7	8	9	4
-1	7	7	2	8	4
0	7	8	9	5	7
1	8	9	3	1	5
2	1	7	5	6	7

	-1	0	1
-1	1	0	1
0	0	1	0
1	1	0	1

*

$$\begin{aligned}
 f(1,1) &= K[-1,-1] I[0,0] + K[-1,0] I[0,1] + K[-1,1] I[0,2] + K[0,-1] I[1,0] \\
 &\quad + K[0,0] I[1,1] + K[0,1] I[1,2] + K[1,-1] I[2,0] + K[1,0] I[2,1] \\
 &\quad + K[1,1] I[2,2]
 \end{aligned}$$

$$\begin{aligned}
 &= 9 * 1 + 5 * 0 + 7 * 1 + 3 * 0 + 1 * 1 + 5 * 0 + 5 * 1 + 6 * 0 + \\
 7 * 1 &= 9 + 0 + 7 + 0 + 1 + 0 + 5 + 0 + 7 = 29
 \end{aligned}$$

$$\begin{aligned}
 f(1,0) &= 8 * 1 + 9 * 0 + 5 * 1 + 9 * 0 + 3 * 1 + 1 * 0 + 7 * 1 + 5 * 0 + 6 * 1 \\
 &= 8 + 0 + 5 + 0 + 3 + 0 + 7 + 0 + 6 = 29
 \end{aligned}$$

$$\begin{aligned}
 f(1,-1) &= 7 * 1 + 8 * 0 + 9 * 1 + 8 * 0 + 9 * 1 + 3 * 0 + 1 * 1 + 7 * 0 + 5 * 1 \\
 &= 7 + 0 + 9 + 0 + 9 + 0 + 1 + 0 + 5 = 31
 \end{aligned}$$

$$\begin{aligned}
 f(0,1) &= 2 * 1 + 8 * 0 + 4 * 1 + 9 * 0 + 5 * 1 + 7 * 0 + 3 * 1 + 1 * 0 + 5 * 1 \\
 &= 2 + 0 + 4 + 0 + 5 + 0 + 3 + 0 + 5 = 19
 \end{aligned}$$

$$\begin{aligned}
 f(0,0) &= 7 * 1 + 2 * 0 + 8 * 1 + 8 * 0 + 9 * 1 + 5 * 0 + 9 * 1 + 3 * 0 + 1 * 1 \\
 &= 7 + 0 + 8 + 0 + 9 + 0 + 9 + 0 + 1 = 34
 \end{aligned}$$

$$\begin{aligned}
 f(0,-1) &= 7 * 1 + 7 * 0 + 2 * 1 + 7 * 0 + 8 * 1 + 9 * 0 + 8 * 1 + 9 * 0 + 3 * 1 \\
 &= 7 + 0 + 2 + 0 + 8 + 0 + 8 + 0 + 3 = 28
 \end{aligned}$$

$$\begin{aligned}
 f(-1,1) &= 8 * 1 + 9 * 0 + 4 * 1 + 2 * 0 + 8 * 1 + 4 * 0 + 9 * 1 + 5 * 0 + 7 * 1 \\
 &= 8 + 0 + 4 + 0 + 8 + 0 + 9 + 0 + 7 = 36
 \end{aligned}$$

$$\begin{aligned}
 f(-1,0) &= 7 * 1 + 8 * 0 + 9 * 1 + 7 * 0 + 2 * 1 + 8 * 0 + 8 * 1 + 9 * 0 + 5 * 1 \\
 &= 7 + 0 + 9 + 0 + 2 + 0 + 8 + 0 + 5 = 31
 \end{aligned}$$

$$\begin{aligned}
 f(-1,-1) &= 7 * 1 + 7 * 0 + 8 * 1 + 7 * 0 + 7 * 1 + 2 * 0 + 7 * 1 + 8 * 0 + 9 \\
 &\quad * 1 = 7 + 0 + 8 + 0 + 7 + 0 + 7 + 0 + 9 = 38
 \end{aligned}$$

$$f = \begin{array}{|c|c|c|} \hline 38 & 31 & 36 \\ \hline 28 & 34 & 19 \\ \hline 31 & 29 & 29 \\ \hline \end{array}$$

Setelah kita mendapatkan perhitungan menggunakan konvolusi kita akan lanjut ke fungsi aktivasi yang kita gunakan, yaitu ReLu dengan rumus $f(x) = \max(0, x)$ yang mana semua hasil negatif kita rubah menjadi nol.

$$f(x) = \max(0, x)$$

$$f(1,1) = \max(0,29) = 29$$

$$f(1,0) = \max(0,29) = 29$$

$$f(1,-1) = \max(0,31) = 31$$

$$f(0,1) = \max(0,19) = 19$$

$$f(0,0) = \max(0,34) = 34$$

$$f(0,-1) = \max(0,28) = 28$$

$$f(-1,1) = \max(0,36) = 36$$

$$f(-1,0) = \max(0,31) = 31$$

$$f(-1,-1) = \max(0,38) = 38$$

Jadi f untuk konvolusi ke dua adalah seperti yang diatas,lalu kita akan lanjut ke *max pooling* ke dua, dibawah adalah hasil dari *max pooling* ke dua.

38	31	36	~>	38	
28	34	19			
31	29	29			

38	31	36	~>	38	36
28	34	19			
31	29	29			

38	31	36	~>	38	36
28	34	19		34	
31	29	29			

38	31	36	~>	38	36
28	34	19		34	34
31	29	29			

Setelah kita mendapat hasilnya, kita akan melanjutkan kembali ke *layer* kita yang berikutnya yaitu konvolusional *layer* yang ketiga. Matrix *input* akan ditambah supaya menjadi lebih mudah untuk dihitung. *input* yang akan kita masukkan dan akan langsung kita hitung untuk *layer* konvolusi ketiga bisa kita lihat dibawah.

	-2	-1	0	1	2		
-2	38	36	14	11	5	*	
-1	34	34	5	8	6		
0	10	14	7	4	8		
1	7	14	6	2	13		
2	6	15	8	2	11		

	-1	0	1
-1	1	0	1
0	0	1	0
1	1	0	1

$$\begin{aligned}
 f(1,1) &= K[-1,-1]I[0,0] + K[-1,0]I[0,1] + K[-1,1]I[0,2] + K[0,-1]I[1,0] \\
 &\quad + K[0,0]I[1,1] + K[0,1]I[1,2] + K[1,-1]I[2,0] + K[1,0]I[2,1] \\
 &\quad + K[1,1]I[2,2] \\
 &= 7 * 1 + 4 * 0 + 8 * 1 + 6 * 0 + 2 * 1 + 13 * 0 + 8 * 1 + 2 * 0 + \\
 11 * 1 &= 7 + 0 + 8 + 0 + 2 + 0 + 8 + 0 + 11 = 36 \\
 f(1,0) &= 14 * 1 + 7 * 0 + 4 * 1 + 14 * 0 + 6 * 1 + 2 * 0 + 15 * 1 + 8 * 0 + 2 \\
 * 1 &= 14 + 0 + 4 + 0 + 6 + 0 + 15 + 0 + 2 = 41 \\
 f(1,-1) &= 10 * 1 + 14 * 0 + 7 * 1 + 7 * 0 + 14 * 1 + 6 * 0 + 6 * 1 + 15 * 0 \\
 + 8 * 1 &= 10 + 0 + 7 + 0 + 14 + 0 + 6 + 0 + 8 = 45 \\
 f(0,1) &= 5 * 1 + 8 * 0 + 6 * 1 + 7 * 0 + 4 * 1 + 8 * 0 + 6 * 1 + 2 * 0 + 13 * 1 \\
 &= 5 + 0 + 6 + 0 + 4 + 0 + 6 + 0 + 13 = 34 \\
 f(0,0) &= 34 * 1 + 5 * 0 + 8 * 1 + 14 * 0 + 7 * 1 + 4 * 0 + 14 * 1 + 6 * 0 + 2 \\
 * 1 &= 34 + 0 + 8 + 0 + 7 + 0 + 14 + 0 + 2 = 65
 \end{aligned}$$

$$f(0, -1) = 34 * 1 + 34 * 0 + 5 * 1 + 10 * 0 + 14 * 1 + 7 * 0 + 7 * 1 + 14 * 0 + 6 * 1 = 34 + 0 + 5 + 0 + 14 + 0 + 7 + 0 + 6 = 66$$

$$f(-1, 1) = 14 * 1 + 11 * 0 + 5 * 1 + 5 * 0 + 8 * 1 + 6 * 0 + 7 * 1 + 4 * 0 + 8 * 1 = 14 + 0 + 5 + 0 + 8 + 0 + 7 + 0 + 8 = 42$$

$$f(-1, 0) = 36 * 1 + 14 * 0 + 11 * 1 + 34 * 0 + 5 * 1 + 8 * 0 + 14 * 1 + 7 * 0 + 4 * 1 = 36 + 0 + 11 + 0 + 5 + 0 + 14 + 0 + 4 = 70$$

$$f(-1, -1) = 38 * 1 + 36 * 0 + 14 * 1 + 34 * 0 + 34 * 1 + 5 * 0 + 10 * 1 + 14 * 0 + 7 * 1 = 38 + 0 + 14 + 0 + 34 + 0 + 10 + 0 + 7 = 103$$

f =

103	70	42
66	65	34
45	41	36

Setelah kita mendapatkan perhitungan menggunakan konvolusi kita akan lanjut ke fungsi aktivasi yang kita gunakan, yaitu ReLu dengan rumus $f(x) = \max(0, x)$ yang mana semua hasil negatif kita rubah menjadi nol.

$$f(x) = \max(0, x)$$

$$f(1, 1) = \max(0, 36) = 36$$

$$f(1, 0) = \max(0, 41) = 41$$

$$f(1, -1) = \max(0, 45) = 45$$

$$f(0, 1) = \max(0, 34) = 34$$

$$f(0, 0) = \max(0, 65) = 65$$

$$f(0, -1) = \max(0, 66) = 66$$

$$f(-1, 1) = \max(0, 42) = 42$$

$$f(-1, 0) = \max(0, 70) = 70$$

$$f(-1, -1) = \max(0, 103) = 103$$

Setelah kita mendapatkan hasil dari konvolusi ke tiga kita, kita akan lanjut menggunakan *max pooling* kembali. Hasil dari *max pooling* bisa kita lihat dibawah.

103	70	42
66	65	34
45	41	36

 $\sim >$

103	

103	70	42
66	65	34
45	41	36

 $\sim >$

103	70

103	70	42
66	65	34
45	41	36

 $\sim >$

103	70
66	

103	70	42
66	65	34
45	41	36

 $\sim >$

103	70
66	65

Setelah kita mendapat hasilnya, kita akan melanjutkan kembali ke *layer* kita yang berikutnya yaitu konvolusional *layer* yang keempat. Matrix *input* akan ditambah supaya menjadi lebih mudah untuk dihitung. *input* yang akan kita masukkan dan akan langsung kita hitung untuk *layer* konvolusi keempat bisa kita lihat dibawah.

	-2	-1	0	1	2
-2	103	70	13	10	5
-1	66	65	5	5	7
0	15	3	16	2	14
1	10	14	10	13	24
2	17	21	3	15	9

 $*$

	-1	0	1
-1	1	0	1
0	0	1	0
1	1	0	1

$$\begin{aligned}
 f(1,1) &= K[-1,-1] I[0,0] + K[-1,0] I[0,1] + K[-1,1] I[0,2] + K[0,-1] I[1,0] \\
 &\quad + K[0,0] I[1,1] + K[0,1] I[1,2] + K[1,-1] I[2,0] + K[1,0] I[2,1] \\
 &\quad + K[1,1] I[2,2] \\
 &= 16 * 1 + 2 * 0 + 14 * 1 + 10 * 0 + 13 * 1 + 24 * 0 + 3 * 1 + 15 \\
 &\quad * 0 + 9 * 1 = 16 + 0 + 14 + 0 + 13 + 0 + 3 + 9 = 55
 \end{aligned}$$

$$f(1,0) = 3 * 1 + 16 * 0 + 2 * 1 + 14 * 0 + 10 + 1 + 13 * 0 + 21 * 1 + 3 * 0 \\ + 15 * 1 = 3 + 0 + 2 + 0 + 10 + 0 + 21 + 0 + 15 = 51$$

$$f(1,-1) = 15 * 1 + 3 * 0 + 16 * 1 + 10 * 0 + 14 * 1 + 10 + 0 + 17 * 1 + 21 \\ * 0 + 3 * 1 = 15 + 0 + 16 + 0 + 14 + 0 + 17 + 0 + 3 = 65$$

$$f(0,1) = 5 * 1 + 5 * 0 + 7 * 1 + 16 * 0 + 2 * 1 + 14 * 0 + 10 * 1 + 13 * 0 \\ + 24 * 1 = 5 + 0 + 7 + 0 + 2 + 0 + 10 + 0 + 24 = 48$$

$$f(0,0) = 65 * 1 + 5 * 0 + 5 * 1 + 3 * 0 + 16 * 1 + 2 * 0 + 14 * 1 + 10 * 0 \\ + 13 * 1 = 36 + 0 + 5 + 0 + 16 + 0 + 14 + 0 + 13 = 113$$

$$f(0,-1) = 66 * 1 + 65 * 0 + 5 * 1 + 15 * 0 + 3 * 1 + 16 * 0 + 10 * 1 + 14 * 0 \\ + 10 * 1 = 66 + 0 + 5 + 0 + 3 + 0 + 10 + 0 + 10 = 94$$

$$f(-1,1) = 13 * 1 + 10 * 0 + 5 * 1 + 5 * 0 + 5 * 1 + 7 * 0 + 16 * 1 + 2 * 0 \\ + 14 * 1 = 13 + 0 + 5 + 0 + 5 + 0 + 16 + 0 + 14 = 53$$

$$f(-1,0) = 70 * 1 + 13 * 0 + 10 * 1 + 65 * 0 + 5 * 1 + 5 * 0 + 3 * 1 + 16 * 0 \\ + 2 * 1 = 70 + 0 + 10 + 0 + 5 + 0 + 3 + 0 + 2 = 90$$

$$f(-1,-1) = 103 * 1 + 70 * 0 + 13 * 1 + 66 * 0 + 65 * 1 + 5 * 0 + 15 * 1 + 3 \\ * 0 + 16 * 1 = 103 + 0 + 13 + 0 + 65 + 0 + 15 + 0 + 16 \\ = 212$$

f	=	212	90	53
		94	113	48
		65	51	55

Setelah kita mendapatkan perhitungan menggunakan konvolusi kita akan lanjut ke fungsi aktivasi yang kita gunakan, yaitu ReLu dengan rumus $f(x) = \max(0, x)$ yang mana semua hasil negatif kita rubah menjadi nol.

$$f(x) = \max(0, x)$$

$$f(1,1) = \max(0, 55) = 55$$

$$f(1,0) = \max(0, 51) = 51$$

$$f(1, -1) = \max(0, 64) = 65$$

$$f(0, 1) = \max(0, 48) = 48$$

$$f(0, 0) = \max(0, 113) = 113$$

$$f(0, -1) = \max(0, 94) = 94$$

$$f(-1, 1) = \max(0, 53) = 53$$

$$f(-1, 0) = \max(0, 90) = 90$$

$$f(-1, -1) = \max(0, 212) = 212$$

Setelah kita mendapatkan hasil dari konvolusi keempat kita, kita akan lanjut menggunakan *max pooling* kembali. Hasil dari *max pooling* bisa kita lihat dibawah.

212	90	53	~>	212	
94	113	48			
65	51	55			

212	90	53	~>	212	113
94	113	48			
65	51	55			

212	90	53	~>	212	113
94	113	48		113	
65	51	55			

212	90	53	~>	212	113
94	113	48		113	113
65	51	55			

Setelah kita mendapat hasilnya, kita akan melanjutkan kembali ke *layer* kita yang berikutnya yaitu konvolusional *layer* yang kelima dan terakhir. Matrix *input* akan ditambah supaya menjadi lebih mudah untuk dihitung. *input* yang akan kita masukkan dan akan langsung kita hitung untuk *layer* konvolusi kelima bisa kita lihat dibawah.

	-2	-1	0	1	2
-2	212	113	19	9	3
-1	113	113	26	11	1
0	25	26	16	13	13
1	3	4	1	1	11
2	20	26	9	3	23

	-1	0	1
-1	1	0	1
0	0	1	0
1	1	0	1

*

$$f(1,1) = K[-1,-1]I[0,0] + K[-1,0]I[0,1] + K[-1,1]I[0,2] + K[0,-1]I[1,0] \\ + K[0,0]I[1,1] + K[0,1]I[1,2] + K[1,-1]I[2,0] + K[1,0]I[2,1] \\ + K[1,1]I[2,2]$$

$$= 16 * 1 + 13 * 0 + 13 * 1 + 1 * 0 + 1 * 1 + 11 * 0 + 9 * 1 + 3 \\ * 0 + 13 * 1 = 16 + 0 + 13 + 0 + 1 + 0 + 9 + 0 + 13 = 62$$

$$f(1,0) = 26 * 1 + 16 * 0 + 13 * 1 + 4 * 0 + 1 * 1 + 1 * 0 + 16 * 1 + 9 * 0 + 3 \\ * 1 = 26 + 0 + 13 + 0 + 1 + 0 + 16 + 0 + 3 = 69$$

$$f(1,-1) = 25 * 1 + 26 * 0 + 16 * 1 + 3 * 0 + 4 * 1 + 1 * 0 + 20 * 1 + 26 * 0 \\ + 9 * 1 = 25 + 0 + 16 + 0 + 4 + 0 + 20 + 0 + 9 = 74$$

$$f(0,1) = 26 * 1 + 11 * 0 + 1 * 1 + 16 * 0 + 13 * 1 + 13 * 0 + 1 * 1 + 1 * 0 \\ + 11 * 1 = 26 + 0 + 1 + 0 + 13 + 0 + 1 + 0 + 11 = 52$$

$$f(0,0) = 113 * 1 + 26 * 0 + 11 * 1 + 26 * 0 + 16 * 1 + 13 * 0 + 4 * 1 + 1 * 0 \\ + 1 * 1 = 113 + 0 + 11 + 0 + 16 + 0 + 4 + 0 + 1 = 145$$

$$f(0,-1) = 113 * 1 + 113 * 0 + 26 * 1 + 15 * 0 + 26 * 1 + 16 * 0 + 3 * 1 + 4 \\ * 0 + 1 * 1 = 113 + 0 + 26 + 0 + 26 + 0 + 3 + 0 + 1 = 169$$

$$f(-1,1) = 19 * 1 + 9 * 0 + 3 * 1 + 26 * 0 + 11 * 1 + 1 * 0 + 16 * 1 + 13 * 0 \\ + 13 * 1 = 19 + 0 + 3 + 0 + 11 + 0 + 16 + 0 + 13 = 62$$

$$f(-1,0) = 113 * 1 + 19 * 0 + 9 * 1 + 113 * 0 + 26 * 1 + 11 * 0 + 26 * 1 + 16 \\ * 0 + 13 * 1 = 113 + 0 + 9 + 0 + 26 + 0 + 26 + 0 + 13 = 187$$

$$f(-1,-1) = 212 * 1 + 113 * 0 + 19 * 1 + 113 * 0 + 113 * 1 + 26 * 0 + 25 \\ * 1 + 26 * 0 + 16 * 1 \\ = 212 + 0 + 19 + 0 + 113 + 0 + 25 + 0 + 16 = 385$$

$$f = \begin{array}{|c|c|c|} \hline 385 & 187 & 62 \\ \hline 169 & 145 & 52 \\ \hline 74 & 69 & 62 \\ \hline \end{array}$$

Setelah kita mendapatkan perhitungan menggunakan konvolusi kita akan lanjut ke fungsi aktivasi yang kita gunakan, yaitu ReLu dengan rumus $f(x) = \max(0, x)$ yang mana semua hasil negatif kita rubah menjadi nol.

$$f(x) = \max(0, x)$$

$$f(1,1) = \max(0,62) = 62$$

$$f(1,0) = \max(0,69) = 69$$

$$f(1,-1) = \max(0,74) = 74$$

$$f(0,1) = \max(0,52) = 52$$

$$f(0,0) = \max(0,145) = 145$$

$$f(0,-1) = \max(0,169) = 169$$

$$f(-1,1) = \max(0,62) = 62$$

$$f(-1,0) = \max(0,187) = 187$$

$$f(-1,-1) = \max(0,385) = 385$$

$$f = \begin{array}{|c|c|c|} \hline 385 & 187 & 62 \\ \hline 169 & 145 & 52 \\ \hline 74 & 69 & 62 \\ \hline \end{array}$$

Setelah mendapatkan hasil dari konvolusi kita, kita akan lanjut ke tahap selanjutnya yaitu ke *flatten layer* yang mana mengubah hasil kita menjadi matriks 1 dimensi agar bisa dibaca untuk tahap tahap selanjutnya. Dibawah adalah hasil perubahan ke matrix 1 dimensi.

$$f = \begin{array}{|c|c|c|} \hline 385 & 187 & 62 \\ \hline 169 & 145 & 52 \\ \hline 74 & 69 & 62 \\ \hline \end{array}$$

$$f = [385, 187, 62, 169, 145, 52, 74, 69, 62]$$

layer selanjutnya yang akan kita hitung adalah *fully connected layer* yaitu *layers dense* dengan bias 128 dan aktivasi relu. Untuk perhitungan pada layer ini bisa dilihat dibawah.

$$z = x.W + b$$

z = *output* dari layer ini.

x = *input* pada layer ini atau *output* dari layer sebelumnya.

W = matriks bobot yang memiliki ukuran 9 x 128 (9 dari *input*, 128 dari jumlah neuron/bias).

b = bias (vektor yang berukuran 1x128)

untuk mempermudah perhitungan kita akan mengecilkan ukuran bias/neuron kita menjadi 2. *Input* dari layer sebelumnya adalah sebagai berikut.

$$x = [385, 187, 62, 169, 145, 52, 74, 69, 62]$$

Untuk matrik bobot, kita akan misalkan menjadi

$$W = \begin{bmatrix} 0.3 & -0.4 & 0.6 & 0.3 & 0.2 & 0.4 & 0.8 & 0.9 & -0.2 \\ 0.0 & 0.3 & 0.4 & -0.9 & 0.9 & 0.9 & 0.9 & -0.7 & 0.4 \end{bmatrix}$$

Dan kita misalkan bias kita menjadi

$$b = [0.1, -0.1]$$

Maka perhitungan kita akan menjadi seperti yang bisa kita lihat dibawah.

z

$$= [385, 187, 62, 169, 145, 52, 74, 69, 62] \times \begin{bmatrix} 0.3 & -0.4 & 0.6 & 0.3 & 0.2 & 0.4 & 0.8 & 0.9 & -0.2 \\ 0.0 & 0.3 & 0.4 & -0.9 & 0.9 & 0.9 & 0.9 & -0.7 & 0.4 \end{bmatrix} \\ + [0.1, -0.1]$$

$$z_1 = (385.0,3) + (187. -0.4) + (62.0,6) + (169.0,3) + (145.0,2) + (52.0,4) \\ + (74.0,8) + (69.0,9) + (62. -0,2) + 0.1 \\ = 115.5 - 74.8 + 37.2 + 50.7 + 29 + 20.8 + 59.2 + 62.1 \\ - 12.4 + 0.1 = 287.4$$

$$z_2 = (385.0,0) + (187. 0,3) + (62.0,4) + (169. -0,9) + (145.0,9) \\ + (52.0,9) + (74.0,9) + (69. -0,7) + (62.0,4) - 0.1 \\ = 0 + 56.1 + 24.8 - 176.4 + 130.5 + 46.8 + 66.6 - 48.3 \\ + 24.8 - 0.1 = 124.8$$

$$f(x) = \max(0, x)$$

$$f(287.4) = \max(0, 287.4) = 287.4$$

$$f(124.8) = \max(0, 124.8) = 124.8$$

$$z = [287.4, 124.8]$$

Diatas adalah hasil dari *dense layer* kita, selanjutnya kita masuk ke *dropout layer* dengan parameter 0,2, yang artinya 20% secara acak dari *output* sebelumnya akan dirubah menjadi 0, contohnya misal *output* kita adalah

$$z = [10, 3, 5, 4, 23, 48, 23, 23, 57, 1]$$

hasil yang akan dikeluarkan setelah *dropout layer* adalah

$$z = [0, 3, 5, 4, 23, 0, 23, 23, 57, 1]$$

bisa kita lihat 2 dari 10 angka yang ada berubah menjadi nol. Tapi untuk contoh ini kita tidak akan menggunakan *dropout layer* karena sebelumnya kita memisalkan *output* dari *dense layer* kita cuman memiliki 2 neuron untuk memudahkan perhitungan, selanjutnya kita akan masuk ke layer terakhir kita yaitu *dense layer* dengan fungsi aktivasi *softmax* yang mana berfungsi untuk mengeluarkan *output*

sebagai probabilitas (dengan jumlah 1). Perhitungan yang akan kita lakukan bisa dilihat dibawah.

$$z = x \cdot W + b$$

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^4 e^{z_j}}$$

$$x = [287.4, 124.8]$$

$$\text{Kita misalkan } W = \begin{bmatrix} 0.1 & 0.2 & -0.6 & 0.8 \\ -0.8 & 0.9 & 0.3 & 0.4 \end{bmatrix}$$

$$\text{Dan misalkan bias } b = [0.1 \quad -0.2 \quad 0.05 \quad 0.3]$$

Perhitungannya menjadi

$$z = [287.4, 124.8] \times \begin{bmatrix} 0.1 & 0.2 & -0.6 & 0.8 \\ -0.8 & 0.9 & 0.3 & 0.4 \end{bmatrix} + [0.1 \quad -0.2 \quad 0.05 \quad 0.3]$$

$$z_1 = (287.4 \times 0.1) + (124.8 \times -0.8) + 0.1 = 28.74 - 99.84 + 0.1 = -71$$

$$z_2 = (287.4 \times 0.2) + (124.8 \times 0.9) - 0.2 = 57.48 + 112.32 - 0.2 = 169.6$$

$$z_3 = (287.4 \times -0.6) + (124.8 \times 0.3) + 0.05 = -172.44 + 37.44 + 0.05 \\ = -134.95$$

$$z_4 = (287.4 \times 0.8) + (124.8 \times 0.4) + 0.3 = 229.92 + 49.92 + 0.3 = 280.14$$

$$z = [-71, 169.6, -134.95, 280.14]$$

Karena hasil terlalu besar kita akan membagi hasil dengan seratus sehingga lebih mudah dihitung. menjadi

$$z = [-0.71, 1.69, -1.34, 2.80]$$

Hitungan eksponensial tiap element adalah

$$e^{z_1} = e^{-0.71} \approx 0.491$$

$$e^{z_2} = e^{1.69} \approx 5.419$$

$$e^{z_3} = e^{-1.34} \approx 0.261$$

$$e^{z_4} = e^{2.80} \approx 16.444$$

$$\text{softmax}(z_1) = \frac{e^{z_1}}{\sum_{j=1}^4 e^{z_j}} = \frac{0.491}{0.491 + 5.419 + 0.261 + 16.444} = \frac{0.491}{22.615} = 0.021$$

$$\text{softmax}(z_2) = \frac{e^{z_2}}{\sum_{j=1}^4 e^{z_j}} = \frac{5.419}{0.491 + 5.419 + 0.261 + 16.444} = \frac{5.419}{22.615} = 0.236$$

$$\text{softmax}(z_3) = \frac{e^{z_3}}{\sum_{j=1}^4 e^{z_j}} = \frac{0.261}{0.491 + 5.419 + 0.261 + 16.444} = \frac{0.261}{22.615} = 0.012$$

$$\text{softmax}(z_4) = \frac{e^{z_4}}{\sum_{j=1}^4 e^{z_j}} = \frac{16.444}{0.491 + 5.419 + 0.261 + 16.444} = \frac{16.444}{22.615} = 0.727$$

Jadi hasil dari *input* yang kita masukkan adalah kemungkinan ada di kelas 4 yang mana memiliki probabilitas tertinggi dengan probabilitas 72.7%.

3.5 Object detection dengan YOLO

Setelah dilakukan pra-proses untuk dimasukkan ke dalam algoritma YOLO, gambar yang sudah di pra-proses dimasukkan ke dalam algoritma YOLO untuk di *training*, algoritma ini digunakan untuk menentukan kalau gambar yang dimasukkan oleh *user* merupakan gambar yang memiliki daun kelapa sawit didalamnya. Misalnya gambar yang dimasukkan adalah gambar daun kelapa sawit, model akan mengeluarkan *output* 1, jika tidak ada daun kelapa sawit terdeteksi pada gambar yang dimasukkan oleh *user*, maka model akan mengeluarkan *output* 0.

3.5 Evaluasi model

Model akan dievaluasi menggunakan *confusion matrix* yang akan menghasilkan nilai *accuracy*, *loss*, *val_accuracy*, *val_loss*. Yang mana dari nilai – nilai ini lah kita dapat melihat apakah model kita bekerja dengan baik atau tidak. Untuk menghitung akurasi kita menggunakan *confusion matrix* yang mana rumus dari akurasi adalah sebagai berikut

$$\text{akurasi} = \frac{\text{jumlah prediksi benar}}{\text{total prediksi semuanya}}$$

Contoh perhitungan akurasi, kita misalkan dari 100 uji coba, jumlah prediksi benar ada 32, jadi cara menghitungnya adalah seperti dibawah

$$akurasi = \frac{32}{100} = 0.32 = 32\%$$

Jadi hasil akurasi yang kita dapatkan adalah 32%. Penilaian pada akurasi ini akan dilakukan disetiap *epoch* nya, misalnya kita melatih model kita menggunakan 10 *epoch*, perhitungan akurasi akan dilakukan 10 kali, begitu juga misalnya kita menggunakan 20 *epoch*, 30 *epoch*, dan seterusnya.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Perangkat yang Digunakan

Perangkat yang digunakan untuk membuat model *CNN* dan *YOLO* serta mengaplikasikannya ke *Android* untuk bisa digunakan sebagai alat klasifikasi penyakit kelapa sawit adalah sebagai berikut:

4.1.1. Perangkat Keras

Detail perangkat keras yang digunakan untuk membuat sistem adalah sebagai berikut:

1. AMD Ryzen 7 4800HS with Radeon Graphic (16CPUs)
2. 16 GB RAM
3. 20GB disk space
4. NVIDIA GeForce GTX 1660 Ti
5. Android version 14

4.1.2 Perangkat Lunak

Detail Perangkat lunak yang digunakan untuk membuat sistem adalah sebagai berikut:

1. Google *Colaboratory*
2. Google *Drive*
3. Sistem Operasi *Windows 10 Enterprise 64-bit*
4. Android Studio

4.1.3 Library



Detail Library yang digunakan untuk membuat sistem adalah sebagai berikut:



1. *Tensorflow*
2. *Os*
3. *Numpy*
4. *Keras*

5. *Albumentations*

4.2 Penerapan Tahap Pengumpulan Dataset

Dataset yang digunakan untuk melatih model didapat dari perkebunan sawit di Kabupaten Mukomuko, dan *kaggle*. Dataset dikumpulkan dengan memfoto daun sawit yang penulis rasa cocok digunakan sebagai data untuk *training* model. Jumlah total data yang digunakan untuk training adalah 400 data. Data yang sudah diambil di *crop* dan *resize* menjadi 500 x 500 piksel agar seragam dan bisa digunakan untuk melatih model. Menggunakan library *albumentations* data ditambah menjadi 1600 data dengan cara 400 data awal di *rotate*, *transpose*, dan *flip* agar jumlah data menjadi lebih banyak. 400 data awal lalu dipilih kembali yang mana yang cocok dan diberi *bouding box* dan koordinat untuk digunakan sebagai data untuk model YOLO dan 1600 data digunakan sebagai data model CNN. Tabel 4.1 dibawah merupakan contoh gambar *dataset* yang digunakan.



Antraknosa	
Bintik Daun	

Garis Kuning	
Sehat	

Tabel 4.1 Contoh gambar *dataset*

4.3 Penerapan Tahap Pra-proses *dataset*

Untuk melakukan *cropping* dan *resizing*, penulis memotong dan me-*resize* sendiri gambar gambar original untuk bisa digunakan sebagai *dataset*. Tabel 4.2 dibawah adalah tabel yang berisi gambar *dataset* sebelum dan sesudah diproses.

Gambar Asli	Gambar yang sudah diproses
	



Tabel 4.2 contoh gambar sebelum dan sesudah *pra-proses*

4.4 Penerapan Tahap Pelatihan untuk model CNN

Setelah gambar di pra-proses, model CNN dibuat dengan *library* keras menggunakan google colaboratory, model CNN yang dibuat menggunakan

arsitekturs *LeNet* memiliki 5 *convolutional layer*, 4 *pooling layer*, sebuah *flatten layer*, dua *layer dense*, sebuah *dropout*, fungsi aktivasi *relu* dan *softmax*, dan *optimizer Adam* dengan *learning rate* yaitu pada 0.001. Pada *convolutional layer* jumlah filter yang penulis gunakan adalah 16, 32, 64, 128, dan 256 dengan ukuran filter 3 x 3. Fungsi aktivasi yang digunakan pada lapisan ini adalah ReLu yang mana dapat membantu model mempelajari hubungan antara fitur – fitur gambar dan dapat mengatasi masalah *exploding and vanishing gradient*, *pooling layer* yang digunakan menggunakan metode *max pooling*, dapat mengatasi *overfitting*, dan dapat mempertahankan informasi yang paling penting. Pooling layer menggunakan *filter* kernel 2 x 2. Setelah gambar melewati semua itu *convolutional layer* dan *pooling layer*, gambar akan direpresentasikan dengan *matriks multidimensi*, *flatten layer* akan mengubah *matriks multidimensi* menjadi *matriks* satu dimensi sehingga bisa dimasukkan ke dalam *fully connected layer* yaitu *layer dense*, output *fully connected layer* lalu masuk ke ke dalam fungsi aktivasi *softmax* untuk didistribusikan probabilitasnya untuk setiap kelas. Model kemudian dilatih menggunakan fungsi *fit()* dengan total *epoch* 30. Dibawah merupakan kode untuk model CNN.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128,activation = 'relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(4, activation='softmax') #Menentukan
jumlah Kelas
])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=
0.001),
              loss='categorical_crossentropy',
```

```
metrics=['accuracy'])

model.summary()
```

4.5 Penerapan Tahap Pelatihan untuk Model YOLO

Model YOLO digunakan untuk mendeteksi *object* pada gambar, untuk menentukan apakah gambar yang dimasukkan oleh *user* memiliki daun sawit atau tidak. Proses ini dilakukan dengan cara, pertama kita mem pra-proses gambar gambar dari data yang terpilih untuk digunakan pada model YOLO ini, gambar yang sudah dipilih kemudian dimasukkan ke dalam website *makesense.ai* yang mana pada website ini digunakan untuk membantu penulis memberikan *bounding box* dan koordinat dari *object* tertentu yang mana pada kasus ini adalah daun sawit, yang digunakan untuk pelatihan model YOLO ini. Setelah gambar di pra-proses , gambar dimasukkan kedalam kode untuk pelatihan YOLO yang mana kode ini diambil dari github *Ultralytics YOLOv5*. Gambar yang sudah di pra-proses dimasukkan kedalam kode untuk *training* dengan epoch 60. Dibawah adalah kode pelatihan yolo.

```
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml # install

import torch
import utils
display = utils.notebook_init() # checks

#edit
from google.colab import drive
drive.mount('/content/drive')

!unzip -q ../train_data2.zip -d ../

# Train YOLOv5s on custom data for 60 epochs
!python train.py --img 640 --batch 10 --epochs 60 --data
custom_data.yaml --weights yolov5x.pt --nosave --cache
```

4.6 Penerapan Tahap Evaluasi

Model CNN dan YOLO yang sudah dilatih lalu disimpan dan dilakukan evaluasi menggunakan *confusion matrix* hasil dari evaluasi untuk model CNN adalah sebagai berikut:

- Accuracy 0.9534
- Loss 0.0980
- Val_accuracy 0.9782
- Val_loss 0.1085

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

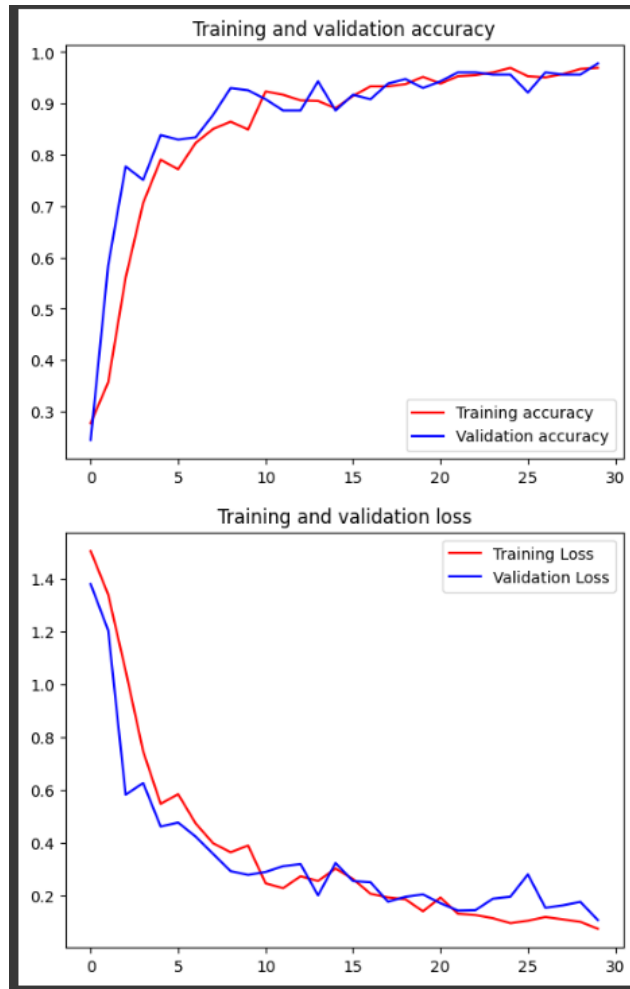
epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()
plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

Diatas adalah kode untuk menyimpan hasil akurasi setiap epoch



Gambar 4.1 Graph untuk setiap epoch

Kode dan Gambar 4.1 diatas adalah kode dan graph untuk melihat perubahan akurasi pada model. Pada model YOLO evaluasi dilakukan dengan melalui metrik deteksi objek untuk setiap epochnya yang mana berisi mAP (*Mean Average Precision*) adalah metrik utama yang menilai kerja model deteksi objek berdasarkan seberapa akurat model dalam mendeteksi dan mengklasifikasikan objek, *precision* mengukur seberapa akurat prediksi model pada objek yang benar-benar ada, dan *recall* mengukur seberapa banyak objek yang benar benar ada berhasil dideteksi oleh model. Berikut hasil evaluasi model YOLO 100 epoch:

- P 0.976
- R 0.99
- mAP50 0.992
- mAP50-95 0.846


```
!python train.py --img 640 --batch 100 --epochs 60 --data
custom_data3.yaml --weights yolov5x.pt --nosave --cache
```

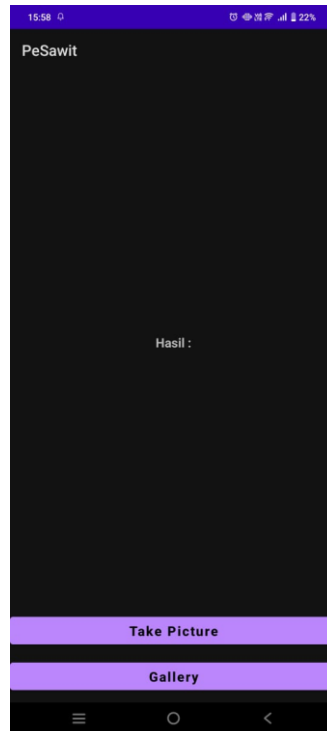
Adalah kode yang digunakan untuk mentraining sekaligus mengetahui evaluasi untuk model.

4.7 Penerapan ke Dalam Android

Setelah kedua model selesai dibuat, kedua model tersebut dirubah ke model *tflite* lalu kedua model tersebut dimasukkan kedalam android agar siap untuk digunakan.

Berikut adalah kode didalam *android studio* yang digunakan untuk mengambil gambar dari *user* untuk dimasukkan kedalam aplikasi untuk diproses.

```
// Mengambil gambar dari camera
camera.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Check camera permission
        if (checkSelfPermission(Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED) {
            Intent cameraIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent, 3);    //
        } else {
            requestPermissions(new
String[]{Manifest.permission.CAMERA}, 100);
        }
    }
});
// Mengambil gambar dari Galeri
gallery.setOnClickListener((new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent cameraIntent = new
Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(cameraIntent, 1);
    }
}));
}
```



Gambar 4.2 Tampak awal aplikasi untuk memilih gambar

Gambar 4.2 diatas adalah tampilan aplikasi sebelum *user* memilih gambar yang mau di *input*. Setelah gambar dipilih oleh user maka gambar akan dimasukkan kedalam model YOLO. Kode dibawah adalah kode untuk memproses gambar dari *user*, gambar akan di pra-proses agar bisa masuk ke dalam model masuk ke dalam model YOLO untuk di proses.

```
//yolo
public void classifyImage2(Bitmap image) {
    try {
        // Load the YOLOv5 FP16 model
        LastFp16 model =
        LastFp16.newInstance(getApplicationContext());
```

```

        // Resize and prepare the input image (640x640 for YOLOv5)
        TensorBuffer inputFeature0 =
TensorBuffer.createFixedSize(new int[]{1, 640, 640, 3},
DataType.FLOAT32);
        ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * 640
* 640 * 3);
        byteBuffer.order(ByteOrder.nativeOrder());

        // Convert Bitmap to ByteBuffer
        int[] intValues = new int[640 * 640];
        image.getPixels(intValues, 0, image.getWidth(), 0, 0,
image.getWidth(), image.getHeight());
        int pixel = 0;

        for (int i = 0; i < 640; i++) {
            for (int j = 0; j < 640; j++) {
                int val = intValues[pixel++]; // Get RGB pixel
value
                byteBuffer.putFloat(((val >> 16) & 0xFF) * (1.f /
255)); // Red
                byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f /
255)); // Green
                byteBuffer.putFloat((val & 0xFF) * (1.f / 255));
// Blue
            }
        }
        // Load image into input tensor
        inputFeature0.loadBuffer(byteBuffer);
        // Run the YOLOv5 model inference
        LastFp16.Outputs outputs = model.process(inputFeature0);
        TensorBuffer outputFeature0 =
outputs.getOutputFeature0AsTensorBuffer();
        // Post-process the YOLOv5 output
        float[] confidences = outputFeature0.getFloatArray(); //
This is where the detection results are located

        // YOLOv5 model outputs detection boxes, classes, and
confidences
        int numDetections = confidences.length / 6; // 6 values
per detection: (x, y, w, h, confidence, class)
        boolean sawitDetected = false;

        for (int i = 0; i < numDetections; i++) {
            int baseIndex = i * 6;
            float confidence = confidences[baseIndex + 4]; // Get
confidence score
            int detectedClass = (int) confidences[baseIndex + 5];
// Get class index (0 for "Sawit")

            // Check if the detected class is "Sawit" and
confidence is above 0.5
            if (detectedClass == 0 && confidence > 0.5) {
                sawitDetected = true;
                break;
            }
        }

        // Set the result based on whether "Sawit" was detected
        if (sawitDetected) {
            result.setText("Sawit");

```

```

        sawittemp = true;
    } else {
        result.setText("Bukan Sawit");
        sawittemp = false;
    }

    // Close the model to free resources
    model.close();

} catch (IOException e) {
    e.printStackTrace();
    // Handle error
}
}

```

Setelah dari model YOLO jika hasil dari gambar yang di deteksi tidak memiliki objek “sawit” maka aplikasi akan langsung mengeluarkan *output* tulisan “Bukan Sawit” tapi jika hasil menunjukkan adanya objek sawit maka gambar yang diberikan oleh *user* akan masuk ke dalam model CNN untuk diklasifikasikan penyakitnya.

Berikut adalah kode untuk memasukkan gambar yang dimasukkan oleh *user* ke dalam model CNN untuk diklasifikasikan penyakitnya. Sama seperti untuk model YOLO, gambar di rubah dulu sehingga menjadi sesuatu yang bisa dimasukkan ke dalam model untuk proses model CNN.

```

// Klasifikasi image (Costum Model) CNN
public void classifyImage(Bitmap image){
    try {
        Model8 model =
        Model8.newInstance(getApplicationContext());

        // Creates inputs for reference.
        TensorBuffer inputFeature0 =
        TensorBuffer.createFixedSize(new int[]{1, 256, 256, 3},
        DataType.FLOAT32);
        ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 *
        imageSize * imageSize * 3);
        byteBuffer.order(ByteOrder.nativeOrder());

        int[] intValues = new int[imageSize * imageSize];
        image.getPixels(intValues, 0, image.getWidth(), 0,0,
        image.getWidth(), image.getHeight());
    }
}

```

```

-         int pixel = 0;

        //iterate over each pixel and extract R,G,B values. Add
        those values individually to the byte buffer
        for(int i = 0; i < imageSize; i++){
            for(int j = 0; j < imageSize; j++){
                int val = intValues[pixel++]; //RGB
                byteBuffer.putFloat(((val >> 16 ) & 0xFF) * (1.f /
255));
                byteBuffer.putFloat(((val >> 8 ) & 0xFF) * (1.f /
255));
                byteBuffer.putFloat(((val & 0xFF) * 1.f / 255));
            }
        }

        inputFeature0.loadBuffer(byteBuffer);

        // Runs model inference and gets result.
        Model8.Outputs outputs = model.process(inputFeature0);
        TensorBuffer outputFeature0 =
        outputs.getOutputFeature0AsTensorBuffer();

        float[] confidences = outputFeature0.getFloatArray();
        //Find the index of the class with the biggest confidence.
        int maxPos = 0;
        float maxConfidence = 0;
        for (int i = 0; i < confidences.length; i++) {
            if (confidences[i] > maxConfidence){
                maxConfidence = confidences[i];
                maxPos = i;
            }
        }
        String[] classes = {"Antraknosa", "Bintik Daun", "Garis
Kuning", "Sehat"};
        result.setText(classes[maxPos]);

        // Releases model resources if no longer used.
        model.close();
    } catch (IOException e) {
        // TODO Handle the exception
    }
}

```

Berikut adalah kode untuk mengeluarkan hasil setelah gambar yang dimasukkan oleh *user* melewati model YOLO dan model CNN. Hasil yang dikeluarkan adalah nama penyakit, sehat, atau bukan sawit.

```

// Mengambil hasil
@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    if(resultCode == RESULT_OK){
        //Mengambil hasil dari Camera
        if(requestCode == 3){
            Bitmap image = (Bitmap) data.getExtras().get("data");

```

```

        // Resize image menjadi Persegi
        int dimension = Math.min(image.getWidth(),
image.getHeight());
        image = ThumbnailUtils.extractThumbnail(image,
dimension, dimension);

        imageView.setImageBitmap(image);

        // Menyamakan size image dengan model
        Bitmap image2 = Bitmap.createScaledBitmap(image, 640,
640, false); //untuk YOLOv5

        image = Bitmap.createScaledBitmap(image, imageSize,
imageSize, false); // untuk CNN

        classifyImage2(image2); //YOLO klasifikasi
        if(sawittemp){
            classifyImage(image);
        }
        else{}
    }else{
        //Mengambil Hasil dari galeri
        Uri dat = data.getData();
        Bitmap image = null;
        try {
            image =
MediaStore.Images.Media.getBitmap(this.getContentResolver(), dat);
        } catch (IOException e) {
            e.printStackTrace();
        }
        imageView.setImageBitmap(image);
        // Resize image menjadi Persegi
        Bitmap image2 = Bitmap.createScaledBitmap(image, 640,
640, false); //untuk YOLOv5
        image = Bitmap.createScaledBitmap(image, imageSize,
imageSize, false); //untuk CNN

        classifyImage2(image2);
        if(sawittemp){
            classifyImage(image);
        }
        else{}
    }
}

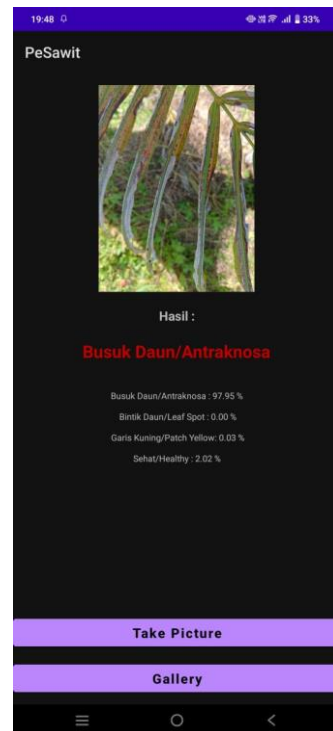
super.onActivityResult(requestCode, resultCode, data);
}

```

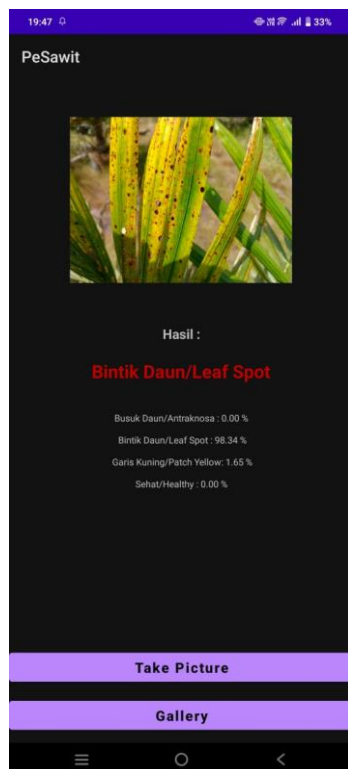
Gambar 4.3, gambar 4.4, gambar 4.5, gambar 4.6 dan gambar 4.7 Dibawah adalah gambar dari hasil yang bisa dihasilkan oleh aplikasi.



Gambar 4.3 Hasil Bukan Sawit



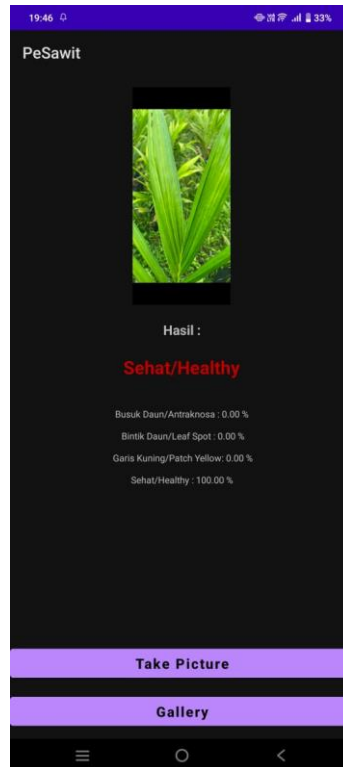
Gambar 4.4 Hasil Antraknosa



Gambar 4.5 Hasil Bintik Daun




Gambar 4.6 Hasil Garis Kuning

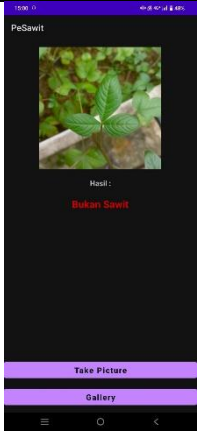





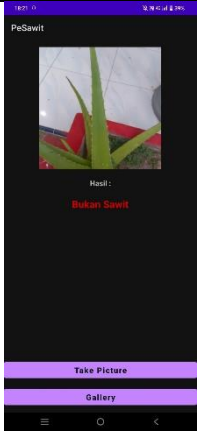


Gambar 4.7 Hasil Sehat

4.8 Pengujian

Pada halaman ini penulis akan menunjukkan beberapa hasil pengujian dengan berbagai macam tanaman. Pengujian bisa dilihat di tabel 4.3 dibawah.

Gambar	Nama tanaman - Hasil
	<p>Sawit - Sehat</p>

	<p>Maman Lanang – Bukan Sawit</p>
	<p>Sawit – Kuning Daun</p>
	<p>Pinang – Bukan Sawit</p>
	<p>Sawit – Bintik Daun</p>

	<p>Lidah Buaya – Bukan Sawit</p>
	<p>Sawit - Antraknosa</p>
	<p>Pandanus veitchii – Bukan Sawit</p>

Tabel 4.3 Pengujian Tanaman

Seperti yang bisa dilihat dari tabel diatas, Algoritma YOLO mampu mendeteksi tanaman yang bukan sawit untuk diklasifikasikan menjadi bukan sawit dan algoritma CNN mampu mengklasifikasikan penyakit kelapa sawit dengan tepat dengan gambar daun kelapa sawit yang diberikan.

BAB V

KESIMPULAN DAN SARAN

Bab V disini membicarakan kesimpulan dan saran dari sistem klasifikasi penyakit sawit menggunakan *Convolutional Neural Networks* dan *You Only Look Once*.

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari bab 4 adalah sebagai berikut:

1. Sistem yang dibuat menggunakan model CNN dan YOLO mampu mengklasifikasikan penyakit kelapa sawit melalui gambar daun kelapa sawit yang diberikan oleh *user* dengan akurasi yang cukup tinggi.
2. Model CNN yang sudah dilatih dengan 1120 data latih mendapatkan evaluasi *score* sebanyak *accuracy* 0.9534, *Loss* 0.0980, *Val_accuracy* 0.9782, dan *Val_loss* 0.1085. Validasi dari model ini menggunakan 480 data validasi, yang mana membuat keseluruhan data dibagi menjadi 70% data *training* dan 30% data validasi.
3. Model YOLO menggunakan pembagian gambar yang sama yaitu 70% data *training* dan 30% data validasi. Evaluasi dari hal ini mendapatkan *score* *precision* 0.976, *Recall* 0.99, mAP50 0.992, dan mAP50-95 0.846.

5.2 Saran

Saran yang bisa diberikan untuk membantu pengembangan sistem selanjutnya adalah sebagai berikut:

1. Pada penelitian selanjutnya diharapkan menggunakan model baru yang lebih mampu, lebih ringan, dan lebih mudah untuk dipersiapkan.
2. Diharapkan pada penelitian selanjutnya dataset ditambah menjadi lebih banyak agar model bisa mempelajari data – data yang beragam untuk segala kondisi.

DAFTAR PUSTAKA

- Asrianda, Aidilof, H., A., & Pangetsu, Y. (2021). Machine Learning for Detection of Palm Oil Leaf Disease Visually using Convolutional Neural Network Algorithm. *JITE (Journal Of Informatics And Telecommunication Engineering)*. 4(2) 286-293.
- Yuliany, S., Aradea, & Rachman, A., N. (2022). Implementasi Deep Learning Pada Sistem Klasifikasi Hama Tanaman Padi Menggunakan Metode Convolutional Neural Network (CNN). *Jurnal Buana Informatika*, 13(1), 54-65.
- Rasywir, S., Sinaga, R., & Pratama, Y. (2020). Analisis dan Implementasi Diagnosis Penyakit Sawit dengan Metode Convolutional Neural Network (CNN). *Jurnal Informatika dan Komputer*. 22(2), 117 – 123.
- Alamsyah, Derry & Pratama, Dicky. (2020). Implementasi Convolutional Neural Networks (CNN) Untuk Klasifikasi Ekspresi Citra Wajah pada Fer-2013 Dataset. *Jurnal Teknologi Informasi*. 4(2), 350-355.
- Luchman, I., Wati, T., & Prasvita, D., S. (2022). Klasifikasi Pohon Kelapa Sawit Menggunakan Citra Lidar dengan Convolutional Neural Network. *Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya*. 290 – 301.
- Irfansyah, D., Mustikasari, M., & Suroso, A. (2021). Arsitektur Convolutional Neural Network (CNN) Alexnet Untuk Klasifikasi Hama pada Citra Daun Tanaman Kopi. *Jurnal Informatika : Jurnal Pengembangan IT (JPIT)*. 6(2), 87-92.
- Diantika, A., S. & Firmanto, Yuki. (2020). Implementasi Machine Learning Pada Aplikasi Penjualan Produk Digital (Studi Pada Grabkios). *Jurnal Ilmiah Mahasiswa FEB*. 9(1).
- Satia, G., A., W., Firmansyah, E., & Umami, A. (2022). Perancangan Sistem Identifikasi Penyakit pada Daun Kelapa Sawit (*Elaeis Gueineensis* Jacq.) dengan Algoritma Deep Learning Convolutional Neural Networks. *Jurnal Ilmiah Pertanian*. 19(1), 1-10.

- Muhammad, S. & Wibowo, A., T. (2021) Klasifikasi Tanaman Aglaonema Berdasarkan Citra Daun Menggunakan Metode Convolutional Neural Network (CNN). *E-Proceeding of Engineering*. 8(5), 10621-10636.
- Irawan, F., A., Sudarma, M., & Khrisne, D., C. (2021). Rancang Bangun Aplikasi Identifikasi Penyakit Tanaman Pepaya California Berbasis Android Menggunakan Metode CNN Model Arsitektur Squeezenet. *Jurnal Spektrum*. 8(2), 18-27.
- Sabilla, I., A. (2020). Arsitektur Convolutional Neural Network (CNN) untuk Klasifikasi Jenis dan Kesegaran Buah Pada Neraca Buah. (*Tesis, Institut Teknologi Sepuluh Nopember*).
- Alwanda, M., R., Ramadhan, R., P., K., & Alamsyah, D. (2020). Implementasi Metode Convolutional Neural Network Menggunakan Arsitektur LeNet-5 untuk Pengenalan Doodle. *Jurnal Algoritme*. 1(1), 45-56.
- Felix, Wijaya, J., Sutra, S., P., Kosasih, P., W., & Sirait, P. (2020). Implementasi Convolutional Neural Network Untuk Identifikasi Jenis Tanaman Melalui Daun. *Jurnal SIFO Mikroskil*. 21(1), 1-10.
- Kumar, G., B. & Rani, N., A. (2020). Traffic Sign Detection Using Convolutional Neural Network. *International Journal of Creative Research Thoughts (IJCRT)*. 8(5), 2635-2641.
- Priwiratama, H., Eris, D. D., Pradana, M. G., & Rozziansha, T. A. P. (2023). STATUS TERKINI PENYAKIT BERCAK DAUN KELAPA SAWIT DI SUMATERA DAN KALIMANTAN. *WARTA Pusat Penelitian Kelapa Sawit*, 28(1), 27-38.
- Kusunah, H., Zahran, M. S., Rifqi, K. N., Putri, D. A., & Hapsari, E. M. W. (2023). Deep Learning Pada Detektor Jerawat: Model YOLOv5. *Journal Sensi*, 9(1), 24 – 35.
- Dwiyanto, R., Widodo, D. W., & Kasih, P. (2022). Implementasi Metode You Only Look Once (YOLOv5) untuk Klasifikasi Kendaraan Pada CCTV Kabupaten Tulungagung, *Seminar Nasional Inovasi Teknologi*, 102 - 104.

Surianti, & Banyal, N., A. (2021). SISTEM PAKAR DIAGNOSA PENYAKIT TANAMAN KELAPA SAWIT BERBASIS ANDROID. *Jurnal Ilmiah MATRIK*. 23(1), 28-33.

Isnaini, Siregar, I., K., & Ihsan, M. (2022). APPLICATION OF THE CERTAINTY FACTOR METHOD FOR DIAGNOSE PALM OIL DISEASE WEB-BASED. *Jurnal Teknik Informatika (JUTIF)*. 3(2), 581-590.

LAMPIRAN

Lampiran 1 Kodingan Model CNN

```
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import pathlib
import pandas as pd
import os

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from sklearn.model_selection import train_test_split

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

from google.colab import drive
drive.mount('/content/drive')
import os

train_dir = "/content/drive/MyDrive/oil-palms-desease/training"

# List the class directories
Labels = os.listdir(train_dir)

# Sort the class names alphabetically
# sorted_labels = sorted(Labels)

# Print sorted class names with their corresponding numeric
labels
for i, label in enumerate(Labels):
    print(f"Class Name: {label}, Numeric Label: {i}")

train_dir="/content/drive/MyDrive/oil-palms-desease/training"
val_dir="/content/drive/MyDrive/oil-palms-desease/test"
Labels=[]

for file in os.listdir(train_dir):
    Labels+= [file]
```

```

Label=[]

for i in range(len(Labels)):
    Label+= [i]

mapping = dict(zip(Labels, Label))
reverse_mapping = dict(zip(Label, Labels))

def mapper(value):
    return reverse_mapping[value]

dataset=[]
valset=[]
count=0
for file in os.listdir(train_dir):
    path=os.path.join(train_dir,file)
    t=0
    for im in os.listdir(path):
        image=load_img(os.path.join(path,im), color_mode='rgb',
target_size=(256,256))
        image=img_to_array(image)
        image=image/255.0
        dataset+= [[image,count]]
        t+=1
    count=count+1
count=0
for file in os.listdir(val_dir):
    path=os.path.join(val_dir,file)
    t=0
    for im in os.listdir(path):
        image=load_img(os.path.join(path,im), color_mode='rgb',
target_size=(256,256))
        image=img_to_array(image)
        image=image/255.0
        valset+= [[image,count]]
        t+=1
    count=count+1

data,labels0=zip(*dataset)
test,testlabels0=zip(*valset)
labels1=to_categorical(labels0)
labels=np.array(labels1)
data=np.array(data)
test=np.array(test)
trainx,testx,trainy,testy=train_test_split(data,labels,test_size=0.2,random_state=44)

```



```

print(trainx.shape)
print(testx.shape)
print(trainy.shape)
print(testy.shape)

datagen =
ImageDataGenerator(horizontal_flip=True,vertical_flip=True,rotation_range=20,zoom_range=0.1,

width_shift_range=0.2,height_shift_range=0.2,shear_range=0.2,fill_mode="nearest")

#Arsitektur leNet
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128,activation = 'relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(4, activation='softmax') #Menentukan
jumlah Kelas
])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=
0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

image=load_img("/content/drive/MyDrive/oil-palms-
desease/test/Bintik Daun/12.png", target_size=(256,256))

image=img_to_array(image)
image=image/255.0
prediction_image=np.array(image)

```

```

prediction_image= np.expand_dims(image, axis=0)
prediction=model.predict(prediction_image)
value=np.argmax(prediction)
move_name=mapper(value)
print("Prediction is {}".format(move_name))

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()
plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()

```

Lampiran 2 Kodingan Model YOLO

```
#YOLO TRAINING YANG INI
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
!pip install -qr requirements.txt comet_ml # install

import torch
import utils
display = utils.notebook_init() # checks

#edit
from google.colab import drive
drive.mount('/content/drive')
!python train.py --img 640 --batch 10 --epochs 60 --data
custom_data3.yaml --weights yolov5x.pt --nosave -cache

!python detect.py --weights runs/train/exp2/weights/last.pt --
img 640 --conf 0.25 --source ../11.jpg
```

Lampiran 3 Kodingan MainActivity Pembuatan Android

```
package com.example.sawit;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.media.Image;
import android.media.ThumbnailUtils;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.sawit.ml.LastFp16;
import com.example.sawit.ml.LastFp32;
import com.example.sawit.ml.Model7;
import com.example.sawit.ml.Model8;

import org.tensorflow.lite.DataType;
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;

public class MainActivity extends AppCompatActivity {

    Button camera, gallery;
    ImageView imageView;
    TextView result;
    TextView prob_Antraknosa, prob_Bintik, prob_Kuning, prob_Sehat;
    int imageSize = 256; //yolo = 640 // cnn = 256
    boolean sawittemp = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        camera = findViewById(R.id.Picture);
        gallery = findViewById(R.id.Gallery);

        result = findViewById(R.id.result);
        prob_Antraknosa = findViewById(R.id.prob_antraknosa);
        prob_Bintik = findViewById(R.id.prob_bintik);
        prob_Kuning = findViewById(R.id.prob_kuning);
        prob_Sehat = findViewById(R.id.prob_sehat);

        imageView = findViewById(R.id.imageView);
```

```

        // Mengambil gambar dari camera
        camera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Check camera permission
                if(checkSelfPermission(Manifest.permission.CAMERA)
== PackageManager.PERMISSION_GRANTED){
                    Intent cameraIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                    startActivityForResult(cameraIntent, 3);    //
                }else{
                    requestPermissions(new
String[]{Manifest.permission.CAMERA}, 100);
                }
            }
        });

        // Mengambil gambar dari Galeri
        gallery.setOnClickListener((new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent cameraIntent = new
Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                startActivityForResult(cameraIntent, 1);
            }
        }));

        //yolo
        public void classifyImage2(Bitmap image) {
            try {
                // Load the YOLOv5 FP32 model
                LastFp32 model =
LastFp32.newInstance(getApplicationContext());

                // Resize and prepare the input image (640x640 for
YOLOv5)
                TensorBuffer inputFeature0 =
TensorBuffer.createFixedSize(new int[]{1, 640, 640, 3},
DataType.FLOAT32);
                ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 *
640 * 640 * 3);
                byteBuffer.order(ByteOrder.nativeOrder());

                // Convert Bitmap to ByteBuffer
                int[] intValues = new int[640 * 640];
                image.getPixels(intValues, 0, image.getWidth(), 0, 0,
image.getWidth(), image.getHeight());
                int pixel = 0;

                for (int i = 0; i < 640; i++) {
                    for (int j = 0; j < 640; j++) {
                        int val = intValues[pixel++]; // Get RGB pixel
value
                        byteBuffer.putFloat(((val >> 16) & 0xFF) *
(1.f / 255)); // Red
                        byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f
/ 255)); // Green

```

```

        byteBuffer.putFloat((val & 0xFF) * (1.f /
255));        // Blue
    }
}

// Load image into input tensor
inputFeature0.loadBuffer(byteBuffer);

// Run the YOLOv5 model inference
LastFp32.Outputs outputs =
model.process(inputFeature0);
TensorBuffer outputFeature0 =
outputs.getOutputFeature0AsTensorBuffer();

// Post-process the YOLOv5 output
float[] confidences = outputFeature0.getFloatArray();
// This is where the detection results are located

// YOLOv5 model outputs detection boxes, classes, and
confidences
int numDetections = confidences.length / 6; // 6
values per detection: (x, y, w, h, confidence, class)
boolean sawitDetected = false;

for (int i = 0; i < numDetections; i++) {
    int baseIndex = i * 6;
    float confidence = confidences[baseIndex + 4]; //
Get confidence score
    int detectedClass = (int) confidences[baseIndex +
5]; // Get class index (0 for "Sawit")

    // Check if the detected class is "Sawit" and
confidence is above 0.5
    if (detectedClass == 0 && confidence > 0.8 ) {
        sawitDetected = true;
        break;
    }
}

// Set the result based on whether "Sawit" was
detected
if (sawitDetected) {
    result.setText("Sawit");
    sawittemp = true;
} else {
    result.setText("Bukan Sawit/Not Palm Leaf");
    prob_Antraknosa.setText("");
    prob_Kuning.setText("");
    prob_Bintik.setText("");
    prob_Sehat.setText("");
    sawittemp = false;
}

// Close the model to free resources
model.close();

} catch (IOException e) {
    e.printStackTrace();
    // Handle error

```

```

    }
}

// Klasifikasi image (Costum Model) CNN
public void classifyImage(Bitmap image){
    try {
        Model8 model =
Model8.newInstance(getApplicationContext());

        // Creates inputs for reference.
        TensorBuffer inputFeature0 =
TensorBuffer.createFixedSize(new int[]{1, 256, 256, 3},
DataType.FLOAT32);
        ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 *
imageSize * imageSize * 3);
        byteBuffer.order(ByteOrder.nativeOrder());

        int[] intValues = new int[imageSize * imageSize];
        image.getPixels(intValues, 0, image.getWidth(), 0,0,
image.getWidth(), image.getHeight());
        int pixel = 0;

        //iterate over each pixel and extract R,G,B values.
Add those values individuallymodel to the byte buffer
        for(int i = 0; i < imageSize; i++){
            for(int j = 0; j < imageSize; j++){
                int val = intValues[pixel++]; //RGB
                byteBuffer.putFloat(((val >> 16 ) & 0xFF) *
(1.f / 255));
                byteBuffer.putFloat(((val >> 8 ) & 0xFF) *
(1.f / 255));
                byteBuffer.putFloat(((val & 0xFF) * 1.f /
255));
            }
        }

        inputFeature0.loadBuffer(byteBuffer);

        // Runs model inference and gets result.
        Model8.Outputs outputs = model.process(inputFeature0);
        TensorBuffer outputFeature0 =
outputs.getOutputFeature0AsTensorBuffer();

        float[] confidences = outputFeature0.getFloatArray();
        //Find the index of the class with the biggest
confidence.
        int maxPos = 0;
        float maxConfidence = 0;
        for (int i = 0; i < confidences.length; i++) {
            if (confidences[i] > maxConfidence){
                maxConfidence = confidences[i];
                maxPos = i;
            }
        }
        String[] classes = {"Busuk Daun/Antraknosa", "Bintik
Daun/Leaf Spot", "Garis Kuning/Patch Yellow", "Sehat/Healthy"};
        result.setText(classes[maxPos]);
    }
}

```

```

        prob_Antraknosa.setText("Busuk Daun/Antraknosa : " +
String.format("%.2f", confidences[0] * 100) + " %");
        prob_Bintik.setText("Bintik Daun/Leaf Spot : " +
String.format("%.2f", confidences[1] * 100) + " %");
        prob_Kuning.setText("Garis Kuning/Patch Yellow: " +
String.format("%.2f", confidences[2] * 100) + " %");
        prob_Sehat.setText("Sehat/Healthy : " +
String.format("%.2f", confidences[3] * 100) + " %");

        // Releases model resources if no longer used.
        model.close();
    } catch (IOException e) {
        // TODO Handle the exception
    }
}

// Mengambil hasil
@Override
protected void onActivityResult(int requestCode, int
resultCode, @Nullable Intent data) {
    if(resultCode == RESULT_OK){
        //Mengambil hasil dari Camera
        if(requestCode == 3){
            Bitmap image = (Bitmap)
data.getExtras().get("data");
            // Resize image menjadi Persegi
            int dimension = Math.min(image.getWidth(),
image.getHeight());
            image = ThumbnailUtils.extractThumbnail(image,
dimension, dimension);

            imageView.setImageBitmap(image);

            // Menyamakan size image dengan model
            Bitmap image2 = Bitmap.createScaledBitmap(image,
640, 640, false); //untuk YOLOv5

            image = Bitmap.createScaledBitmap(image,
imageSize, imageSize, false); // untuk CNN

            classifyImage2(image2); //YOLO klasifikasi
            if(sawittemp){
                classifyImage(image);
            }
            else{}

        }else{
            //Mengambil Hasil dari galeri
            Uri dat = data.getData();
            Bitmap image = null;
            try {
                image =
MediaStore.Images.Media.getBitmap(this.getContentResolver(), dat);
            } catch (IOException e) {
                e.printStackTrace();
            }
            imageView.setImageBitmap(image);
            // Resize image menjadi Persegi

```



```

        Bitmap image2 = Bitmap.createScaledBitmap(image,
640, 640, false); //untuk YOLOv5
        image = Bitmap.createScaledBitmap(image,
imageSize, imageSize, false); //untuk CNN

        classifyImage2(image2);
        if(sawittemp){
            classifyImage(image);
        }
        else{}

    }
}
super.onActivityResult(requestCode, resultCode, data);
}
}

```

BIODATA



Nama : Prisko Banoza
NIM : 191401072
Tempat/ Tanggal Lahir : Bengkulu, 20 Mei 2002
Agama : Islam
Jenis Kelamin : Laki – Laki
Kewarganegaraan : Indonesia
No. Telpn : 082237464460
E – Mail : prisko3rd@gmail.com
Pendidikan Formal : - TK Teratai Indah (2007 – 2008)
- SD N 03 Kota Mukomuko (2008 – 2014)
- SMP N 03 Kota Mukomuko (2014 – 2016)
- SMA N 01 Kota Mukomuko (2016 – 2019)
- Universitas Sumatera Utara (2019 – Sekarang)
Pendidikan Non Formal : - Bangkit Academy (2022 – 2023)