PENGEMBANGAN METODE GEOTAGGING TANAMAN KAKAO SECARA OTOMATIS MENGGUNAKAN SMARTPHONE

TUGAS AKHIR

DANIEL YOGATAMA MAYDIPUTRA

NIM : 311910006

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS MA CHUNG

MALANG

2023

LEMBAR PENGESAHAN

TUGAS AKHIR

PENGEMBANGAN METODE GEOTAGGING UNTUK MEREKAM DATA SPASIAL TANAMAN KAKAO SECARA AUTOMATIS MENGGUNAKAN SMARTPHONE

Oleh:

DANIEL YOGATAMA MAYDIPUTRA

NIM. 311910006

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS dan TEKNOLOGI

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana Komputer (S.Kom.)

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Kestrilia Rega Prilianti, M.Si.	**Hendry Setiawan, ST., M.Kom.**
NIP. 20120035	**NIP. 20100006**

Dekan Fakultas Sains dan Teknologi,

Dr. Kestrilia Rega Prilianti, M.Si.

NIP. 20120035

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan *"PENGEMBANGAN METODE GEOTAGGING TANAMAN KAKAO SECARA OTOMATIS MENGGUNAKAN SMARTPHONE"* adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 21 Juli 2023

Daniel Yogatama Maydiputra

311910006

PENGEMBANGAN METODE GEOTAGGING TANAMAN KAKAO SECARA OTOMATIS MENGGUNAKAN SMARTPHONE

Daniel Yogatama Maydiputra, Kestrilia Rega Prilianti, Hendry Setiawan

Universitas Ma Chung

Abstrak

Kakao merupakan salah satu komoditas perkebunan strategis dalam perekonomian Indonesia. Dengan adanya metode remote sensing, monitoring perkebunan menjadi jauh lebih mudah. Pada citra *orthophoto,* tanaman kakao akan tertutupi oleh kanopi tanaman penaung, sehingga tidak ada informasi mengenai tanaman kakao yang dapat dianalis. Sehingga pada penelitian ini akan dikembangkan sistem geotagging yang dapat memprediksi titik koordinat dan jumlah buah pada citra tanaman kakao. Untuk melakukan prediksi titik koordinat tanaman diperlukan titik koordinat citra, *heading* serta jarak tanaman terhadap kamera. Hal ini dapat dilakukan dengan menggunakan model *Convolutional Neural Network monocular depth estimation*. Model tersebut akan menghasilkan citra *heatmap* yang setiap pikselnya merepresentasikan nilai prediksi kedalaman. Lalu akan dikembangkan *Artificial Neural Network* untuk memprediksi nilai jarak dengan input nilai rgb. Akan diambil nilai rgb objek pada citra *heatmap* tersebut yang kemudian digunakan untuk memprediksi jarak kamera terhadap tanaman. Kemudian digunakan *Vincenty Formula* untuk mengkalkulasi titik koordinat tanaman berdasarkan titik koordinat kamera, *heading,* serta jarak kamera terhadap tanaman. Untuk melakukan deteksi buah kakao akan digunakan model CNN YOLOV8. Pengujian performa model dilakukan dengan mengevaluasi nilai loss setiap model. Untuk melakukan prediksi jarak, model ANN yang memiliki akurasi terbaik adalah model ANN yang dilatih menggunakan *optimizer adamax* dengan *batch size* 7 pada epoch 1000. Model ANN tersebut memiliki akurasi loss MAE sebesar 0.333776. Sedangkan model YOLO yang memiliki performa terbaik adalah model YOLOV8 nano dengan epoch 100. Model tersebut memiliki nilai *precision* 0.907 dan *recall* 0.958. Sistem yang dikembangkan pada penelitian ini merupakan prototipe sistem monitoring perkebunan yang praktis.

Kata Kunci: *Artificial Neural Network***,** Citra *orthophoto***,** *Convolutional Neural Network,* Kakao**,** *Remote sensing***,** YOLOV8

DEVELOPMENT OF AUTOMATIC GEOTAGGING METHOD FOR COCOA PLANTATION ON SMARTPHONE

Daniel Yogatama Maydiputra, Kestrilia Rega Prilianti, Hendry Setiawan

Universitas Ma Chung

Abstract

Cocoa is a crucial plantation commodity in Indonesia's economy. Remote sensing methods offer an easier way to monitor plantations. However, the presence of shade trees obstructs cocoa plant analysis in orthophoto images. To address this, a geotagging system was developed in this research to predict cocoa plant coordinates and fruit quantity. The system utilizes a Convolutional Neural Network (CNN) for monocular depth estimation, generating a heatmap representing depth predictions. An Artificial Neural Network (ANN) predicts the distance using input RGB values extracted from the heatmap image, which helps determine the camera-to-plant distance and heading. The Vincenty Formula is then applied to calculate plant coordinates based on the camera coordinates. For cocoa fruit detection, a YOLOV8 CNN model is employed. Model performance is evaluated by assessing loss values, with the ANN model achieving a best MAE loss accuracy of 0.333776 when trained with the adamax optimizer, a batch size of 7, and 1000 epochs. The top-performing YOLO model is YOLOV8 nano with 100 epochs, demonstrating a precision of 0.907 and recall of 0.958. This research presents a practical prototype of a plantation monitoring system.

Keywords: *Artificial Neural Network, Cocoa, Convolutional Neural Network, Orthophoto images, Remote sensing, YOLOV8*

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Kuasa atas segala rahmat dan karuniaNya penulis dapat menyelesaikan tugas akhir ini. Adapun judul dari tugas akhir ini adalah *"PENGEMBANGAN METODE GEOTAGGING TANAMAN KAKAO SECARA OTOMATIS MENGGUNAKAN SMARTPHONE."*

Pada kesempatan ini penulis mengucapkan terima kasih kepada dosen pembimbing yang telah mendampingi dan membimbing penulis selama pembuatan tugas akhir ini.

Diharapkan adanya kritik dan saran yang membangun dari pembaca. Penulis berharap makalah ini dapat berguna bagi penulis, pembaca serta masyarakat.

Malang, 19 Juli 2023

Daniel Yogatama Maydiputra

311910006

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Kakao merupakan salah satu komoditas perkebunan strategis dalam perekonomian Indonesia. Berdasarkan laporan yang dirilis oleh Direktorat Jenderal Perkebunan, pada tahun 2020 nilai ekspor kakao mencapai 1,24 milyar US dolar. Luasan area perkebunan kakao terus menurun sejak tahun 2016. Pada tahun 2016, perkebunan kakao di Indonesia seluas 1,7 juta Ha. Sedangkan pada 2020, perkebunan kakao di Indonesia seluas 1,5 juta Ha. Namun hal ini tidak menghambat produksi kakao. Meskipun produktivitas kakao mengalami penurunan hingga titik terendahnya yakni pada tahun 2019 menyentuh 721 kg/ha, produktivitas kakao kembali meningkat pada tahun 2020. Produktivitas kakao pada tahun 2020 menyentuh angka 723 kg/ha. (Ditjenbun, 2020) Tidak hanya sebagai pendukung ekonomi nasional, kakao juga menjadi sumber pendapatan utama 1,7 juta kepala keluarga petani kakao di Indonesia (Puslitkoka, 2021).

Untuk menghasilkan produksi kakao yang optimal, diperlukan pemeliharaan perkebunan kakao yang baik. Pemelliharaan perkebunan kakao meliputi pemangkasan daun, pengelolaan tanaman penaung, pemupukan, pengendalian hama, pengendalian penyakit dan pengendalian gulma. Pemangkasan dilakukan untuk mengatur jumlah dan sebaran daun. Pemangkasan juga bertujuan untuk mengatur iklim mikro yang tepat untuk pertumbuhan bunga dan buah. Keberadaan tanaman penaung diperlukan untuk mengatur penyinaran matahari, suhu, udara, kelembapan serta laju kehilangan lengas melalui transpirasi maupun evaporasi. Pemupukan dilakukan untuk menambah unsur-unsur hara yang tidak tersedia di dalam tanah. Pengendalian hama juga perlu dilakukan karena tanaman kakao merupakan tanaman yang cukup disukai oleh hama. Tepatnya ada 130 spesies dalam kelompok serangga yang merupakan hama dari tanaman kakao. Hal ini dilakukan dengan tujuan mengurangi kerusakan yang dapat mengurangi produksi kakao dan kerusakan lingkungan. Pengendalian penyakit dilakukan untuk mengurangi kegagalan dan menjaga kelestarian lingkungan. Sedangkan pengendalian gulma perlu dilakukan karena apabila dihiraukan dapat menyebabkan

terhambatnya pertumbuhan tanaman muda dan menunda masa tanaman menghasilkan, serta berpotensi untuk menjadi inang hama dan penyakit (Prawoto, 2009).

Sebelum adanya bantuan teknologi, monitoring perkebunan dilakukan secara manual / tradisional. Dengan adanya metode *remote sensing, monitoring* perkebunan menjadi jauh lebih mudah. Chatterjee (2018) menyatakan, *monitoring* perkebunan berbasis *remote sensing* terbukti menjadi metode yang paling efisien untuk melakukan estimasi dan prediksi hasil produksi dari waktu ke waktu. *Remote sensing* dapat memonitor pertumbuhan tanaman berdasarkan periode waktu tertentu dengan cepat. Hal ini dapat membantu pengelola perkebunan untuk melakukan mitigasi pada tanaman dengan cepat apabila terdapat tanaman yang terkena serangan hama dan penyakit. *Monitoring* tanaman dilakukan untuk menganalisis dan mempelajari kondisi tanaman saat ini, serta membantu pengamatan pertumbuhan tanaman.

Ditjenbun menjelaskan, tanaman kakao memerlukan tanaman penaung untuk meredam suhu maksimum dari paparan sinar matahari yang dapat merusak tanaman kakao. Tanaman penaung juga berfungsi sebagai pemecah angin karena daun tanaman kakao mudah rontok. Namun, keberadaan tanaman penaung menjadi faktor penghambat *remote sensing* pada perkebunan kakao. Metode *remote sensing* pada umumnya menggunakan citra *orthophoto* yang merupakan hasil dari satelit / penerbangan UAV sebagai sumber pengamatan. Pada citra *orthophoto,* tanaman kakao akan tertutupi oleh kanopi tanaman penaung, sehingga tidak ada informasi mengenai tanaman kakao yang dapat dianalisa (Ditjenbun, 2021).

Untuk mengatasi kendala tersebut, diperlukan teknologi geotagging otomatis dengan memprediksi titik koordinat tanaman kakao. Dengan menggunakan *monocular depth estimation* serta *triangulation* dapat dilakukan pemetaan otomatis. Pemetaan otomatis ini dilakukan untuk mengidentifikasi kemunculan beberapa objek yang mirip secara otomatis serta mendapatkan titik koordinat dari objek tersebut. (Vladimir A. Krylov, 2018).

Dikarenakan terbatasnya informasi yang didapatkan pada metode *monitoring* dari atas kanopi, maka penulis menggunakan metode *monitoring* di bawah kanopi pada penelitian ini. Penulis menawarkan alternatif yang murah,

cepat, serta praktis untuk melakukan *monitoring* tanaman kakao dibawah kanopi. Pada penelitian ini akan digunakan *smartphone* untuk melakukan pengambilan gambar. *Monitoring* dilakukan dengan mengambil gambar tanaman kakao menggunakan *smartphone* dari depan tanaman. Metode ini diharapkan dapat memberikan informasi yang lebih lengkap dan akurat mengenai tanaman kakao. *Monitoring* akan dilakukan dengan melakukan akuisisi citra setiap tanaman kakao. Teknologi ini akan melakukan prediksi titik koordinat tanaman kakao berdasarkan titik koordinat kamera pada saat pengambilan gambar dilakukan serta estimasi jarak tanaman kakao dari kamera.

1.2. Identifikasi Masalah

Analisa kondisi tanaman kakao menggunakan *orthophoto* memiliki beberapa keterbatasan. Data *orthophoto* hanya dapat menangkap informasi yang nampak dari atas tanaman kakao. Sedangkan, di Indonesia, kebanyakan tanaman kakao ditanam bersama dengan tanaman-tanaman lainnya sebagai tanaman penaung. Hal ini dilakukan untuk mengurangi cahaya yang mengenai tanaman kakao. Tanaman penaung menyebabkan teknik remote sensing menggunakan *orthophoto* menjadi solusi yang kurang sesuai. Apabila dilihat dari atas, tanaman kakao seringkali tertutup oleh tanaman penaungnya. Sehingga tidak dapat dilakukan analisa lebih lanjut terkait tanaman kakao.

1.3. Batasan Masalah

Berikut beberapa batasan masalah dalam penelitian ini:

a. Posisi kamera lurus terhadap tanaman kakao untuk membatasi variasi sudut pengambilan gambar
b. Tingkat akurasi model geotagging berdasarkan gps pada *smartphone*
c. Buah yang terhitung hanya buah yang terlihat jelas pada citra
d. Objek yang diamati : Tanaman Kakao
e. Akuisisi Citra dilakukan menggunakaan telepon genggam
f. Tanaman kakao yang diamati berada di perkebunan kakao di pasuruan.

g. Data Tanaman Kakao yang dihasilkan adalah jumlah buah pada tanaman kakao

1.4. Perumusan Masalah

Berdasarkan identifikasi masalah di atas, berikut rumusan masalah dalam penelitian ini.

a. Bagaimana pengembangan metode geotagging yang dapat memprediksi titik koordinat tanaman?
b. Bagaimana pengembangan metode untuk melakukan kuantifikasi otomatis yang dapat digunakan untuk menghitung jumlah buah kakao pada citra?

1.5. Tujuan Penelitian

a. Mengembangkan metode geotagging yang dapat memprediksi titik koordinat tanaman.
b. Mengembangkan metode untuk melakukan kuantifikasi otomatis yang dapat digunakan untuk menghitung jumlah buah kakao pada citra.

1.6. Luaran

Metode remote sensing baru yang dapat mengakuisisi lebih banyak informasi mendetail pada kakao serta publikasi ilmiah terkait metode tersebut.

1.7. Manfaat

- Bagi Peneliti : melakukan penerapan ilmu.
- Bagi Masyarakat : mempermudah monitoring tanaman kakao pada perkebunan kakao.
- Bagi Universitas : menambah kepustakaan.

1.8. Sistematika Penulisan

Sistematika dalam penulisan proposal Tugas Akhir ini akan dibagi menjadi lima bab seperti berikut.

Bab I: Pendahuluan

Pada bab pendahuluan, akan dijelaskan latar belakang, identifikasi masalah, batasan masalah, tujuan penelitian, manfaat penelitian, luaran tugas akhir, dan sistematika penulisan.

Bab II: Tinjauan Pustaka

Pada bab tinjauan pustaka, akan diuraikan secara sistematis literatur yang digunakan dalam penyusunan Tugas Akhir. Hal ini bertujuan untuk memperoleh landasan teori terkait dengan CNN, Geotagging dan Kakao.

Bab III: Metodologi Penelitian

Bab ini akan menjelaskan tahapan pengerjaan dan analisis perancangan awal sistem yang akan dibuat. Tahapan tersebut mencakup identifikasi masalah, studi pustaka, pengumpulan data, profiling, desain sistem, dan pengujian.

Bab IV: Hasil dan Pembahasan

Bab ini akan menjelaskan tahapan pengerjaan dan analisis perancangan awal sistem yang akan dibuat. Tahapan tersebut mencakup identifikasi masalah, studi pustaka, pengumpulan data, desain sistem, dan pengujian.

Bab V: Kesimpulan dan Saran

Bab ini akan berisi simpulan dari hasil penelitian Tugas Akhir yang telah dilakukan, serta saran yang mungkin dapat dilakukan untuk memperbaiki sistem aplikasi dalam penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1. Komoditas Kakao dan Permasalahannya

Komoditas kakao merupakan tanaman tropis yang menghasilkan biji kakao yang digunakan sebagai bahan baku dalam produksi cokelat. Kakao tumbuh terutama di daerah tropis, terutama di Afrika Barat, Asia Tenggara, dan Amerika Selatan. Kakao memiliki peran penting dalam perekonomian banyak negara produsen, memberikan mata pencaharian bagi petani dan pendapatan ekspor yang signifikan.

2.1.1. Komoditas Kakao

Kakao berasal dari hutan tropis di Amerika Tengah. Awalnya biji kakao diolah oleh suku Indian dengan cara dikeringkan di bawah sinar matahari, lalu disangrai dan dijadikan adonan. Suku Indian membuat minuman dari kakao, dengan cara mencampur adonan tersebut dengan vanili. Pada masa tersebut, kakao tidak hanya berfungsi sebagai minuman tetapi juga digunakan sebagai mata uang atau alat tukar-menukar antar individu. Bangsa spanyol juga mencoba untuk mengolah kakao dengan cara mereka sendiri yaitu dengan mengsangrai biji kakao, menumbuk lalu ditambahkan gula tebu. Metode tersebut lebih disukai oleh Bangsa Spanyol. Pada tahun 1560, Spanyol memperkenalkan kakao di Indonesia tepatnya di Sulawesi. Kemudian pada tahun 1825-1838, Indonesia melakukan ekspor kakao ke Manila sebanyak 92 ton. Namun, pada periode setelah itu ekspor kakao cenderung menerun karena banyak tanaman kakao yang terserang penyakit. Kakao juga ditanam di Ambon, pada 1859 terdapat 10.000-12.000 tanaman tanaman kakao dan telah menghasilkan 11,6 ton. Di pulau Jawa, kakao baru ditanam pada tahun 1880. (Wahyudi, 2008)

Dikutip dari Statistik Perkebunan Ditjenbun, produksi kakao di Indonesia mencapai puncak dengan nilai 837.918 ton pada tahun 2010. Sedangkan luasan lahan kakao di Indonesia sempat mencapai nilai maksimum pada tahun 2012 seluas 1.774.464 Ha. Luasan lahan perkebunan kakao terus menurun hingga tahun 2022, Indonesia hanya memiliki lahan kakao seluas 1.476.776 Ha. Namun terdapat

peningkatan jumlah produksi kakao dari tahun 2021 ke tahun 2022. Pada tahun 2021, Indonesia memproduksi 706.636 ton kakao. Sedangkan pada tahun 2022 Indonesia memproduksi 732.256 ton kakao. Dengan berkurangnya luasan lahan perkebunan kakao, namun Indonesia berhasil meningkatkan produksi kakao. Artinya Indonesia berhasil meningkatkan produktivitas kakao (Ditjenbun 2021).

2.1.2. Penyakit Vascular Streak Dieback

Tanaman kakao yang terkena serangan *vascular streak dieback* akan menunjukkan gejala adanya daun yang menguning dengan bercak-bercak berwarna hijau. Daun-daun tersebut akan gugur sehingga tampak gejala ranting ompong. Apabila bekas duduk daun disayat akan terlihat tiga buah noktah berwarna cokelat kehitam-hitaman. Pada bekas potongan daun, bekas duduk daun, bekas potongan ranting akan muncul benang-benang berwarna putih. Penyakit ini disebabkan oleh jamur O. theobromae (Wahyudi, 2008).

2.2. *Geographic Coordinate System (GCS)*

Geographic Coordinate System (GCS) adalah sistem berbasis koordinat yang digunakan untuk merepresentasikan posisi suatu lokasi di permukaan bumi. *GCS* menggunakan permukaan 3 dimensi berbentuk bola untuk mendefinisikan posisi. Terdapat 3 komponen *GCS* yakni *latitude, longitude* dan *altitude. Latitude* dan *longitude* mendifinisikan lokasi di permukaan sedangkan *altitude* mendefinisikan elevasi atatu ketinggian diatas atau kedalaman dibawah permukaan laut. *GCS* yang paling umum digunakan adalah WGS 84 *(World Geodetic System)*, yang digunakan untuk navigasi, pemetaan dan sistem penentuan posisi satelit (Longley, 2015).

2.3. *Global Positioning System (GPS)*

GPS adalah sistem navigasi berbasis satelit yang menyediakan informasi lokasi dan waktu pada berbagai kondisi cuaca, di semua area di bumi. Sistem GPS awalnya dikembangkan untuk kegunaan militer dan sekarang digunakan untuk berbagai navigasi dan layanan berbasis lokasi seperti pemetaan, *geotagging,*

pelacakan lokasi. *GPS receiver* mengkalkulasi posisi dengan catatan waktu presisi ketika sinyal dikirim oleh satelit *GPS*. Sinyal ini mengandung informasi mengenai lokasi satelit, waktu ketika sinyal dikirim, kondisi ionosferik dan atmosferik yang mempengaruhi pengiriman sinyal. *Receiver* menggunakan informasi ini untuk menentukan lokasi dan menyediakan koordinat *latitude, longitude dan altitude* (Zarchan, 1996).

2.4. *Geotagging*

Geotagging merupakan proses menambahkan informasi geografi seperti koordinat, serta nama lokasi ke media digital seperti foto, video dan lainnya. *Geotagging* mempermudah untuk melakukan pemetaan dan pencarian berbasis lokasi. Informasi geografi dapat ditambahkan secara manual ataupun otomatis dengan menggunakan perangkat *GPS* atau aplikasi kamera ponsel. *Geotagging* digunakan dalam berbagai bidang termausk pariwisata, jurnalistik, perencanaan tata kota, manajemen lingkungan, dan lainnya. Hal ini menyediakan informasi yang berharga untuk melakukan visualisasi, analisa pola atau tren, relasi ruang fisik dan virtual (Atunggal, 2018).

2.5. *Remote Sensing*

Remote Sensing adalah proses akuisisi data mengenai sebuah objek atau fenomena tanpa melakukan kontak fisik dengan objek. Biasanya dilakukan menggunakan perangkat berbasis satelit atau aerial, yang dapat menangkap informasi mengenai objek. Beberapa sensor digunakan untuk menangkap informasi mengenai objek, seperti sensor spectrometer, radiometer, hyperspectral radiometer, sounder, accelerometer. Data-data yang dihasilkan sensor-sensor tersebut dapat digunakan untuk melakukan *monitoring* informasi spasial berdasarkan waktu. Selama 5 dekade terakhir, teknologi *remote sensing* telah digunakan dalam berbagai riset area lahan basah seperti perubahan penggunaan lahan / pemetaan daerah lahan basah. Siklus karbon dan peringatan perubahan iklim, pelepasan karbon pada kebakaran lahan gambut, serta proses hidrologi pada lahan basah (Guo, 2017).

2.6. *Remote Sensing* pada kakao

Neswati (2019) menggunakan metode *remote sensing* untuk mendapatkan informasi penggunaan lahan dan kesesuaian lahan untuk digunakan sebagai perkebunan kakao. Hasil penelitian tersebut menyatakan bahwa hasil analisa area yang berpotensi untuk dijadikan lahan perkebunan kakao menyatakan 90% dari area yang di analisa sesuai untuk digunakan dengan index kesesuaian berkisar antara 35 hingga 60. Sedangkan 10% dari area yang ada dinyatakan tidak sesuai untuk ditanami kakao dengan *Land Suitability Index* kurang dari 25. Berdasarkan hasil estimasi, produktivitas perkebunan hanya berkisar antara 0,3 hingga 1 ton/ha. Angka ini dikategorikan sebagai produktivitas rendah hingga sedang (R Neswati, 2019).

2.7. Roboflow

Roboflow merupakan platform untuk melakukan pekerjaan dalam lingkup *computer vision.* Platform ini digunakan oleh lebih dari 250.000 *engineer* untuk membuat dataset, melatih model, dan melakukan deploy model pada server. Beberapa fitur dari Roboflow meliputi bounding boxes poligon, label assist, dan infrastruktur pelatihan model. Bagi para pengembang aplikasi yang ingin menggunakan Roboflow dalam bahasa Python, tersedia *library* Python untuk menggunakan Roboflow di PyPI. *Library* Python Roboflow adalah wrapper python yang menghubungkan aplikasi web Roboflow inti dan REST API. Selain itu, terdapat juga utilitas visi komputer sumber terbuka dan tutorial notebook dalam bahasa Python yang tersedia di GitHub. Untuk informasi lebih lanjut tentang cara menggunakan Roboflow, terdapat situs dokumentasi yang tersedia. Situs web tersebut memberikan pengantar tentang Roboflow dan bagaimana menggunakannya untuk membangun model visi komputer yang kuat. Roboflow mempermudah manajemen dataset dan penggunaan dataset (Roboflow, 2023).

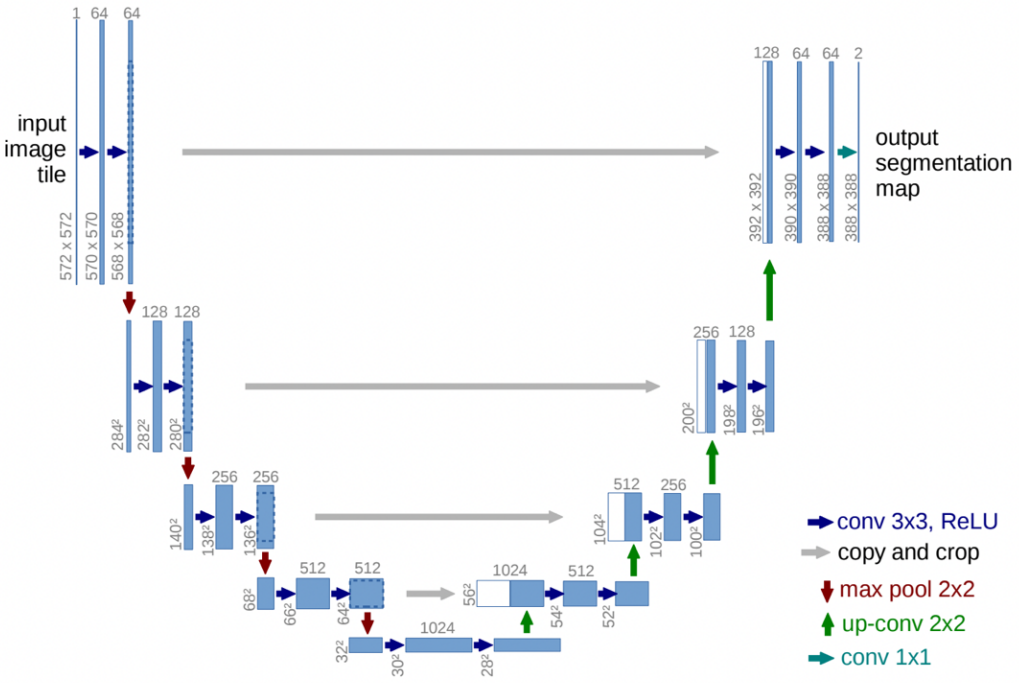
2.8. Sistem Estimasi Titik Koordinat

Dalam penelitian ini, akan dibuat sistem untuk mengkalkulasi estimasi titik koordinat suatu objek berdasarkan data yang tersedia. Sistem ini bertujuan untuk

memberikan perkiraan titik koordinat yang mungkin dari objek yang diamati, berdasarkan informasi yang ada. Metode yang digunakan dalam sistem ini mencakup analisis data spasial, pengolahan citra, dan teknik pemodelan matematis. Tujuan dari penelitian ini adalah meningkatkan akurasi dan ketepatan dalam menentukan estimasi titik koordinat, sehingga dapat digunakan dalam berbagai aplikasi seperti pemetaan, navigasi, dan pemantauan objek di lingkungan. Dalam pengembangan sistem, akan dilakukan pengujian dan evaluasi untuk memastikan kinerja dan keandalan sistem dalam memberikan estimasi titik koordinat yang akurat.

2.8.1. Metode estimasi kedalaman monokular

Estimasi kedalaman monocular merupakan sebuah teknik untuk memperkirakan kedalaman secara 3 dimensi dari citra 2 dimensi. Kedalaman 3 dimensi diprediksi dengan mengambil informasi dari gambar yang dihasilkan oleh satu kamera. Salah satu Teknik estimasi kedalaman monocular yang paling populer adalah menggunakan Convolutional Neural Network (CNN). Model CNN dapat dilatih untuk mempelajari pola-pola dalam data gambar yang menunjukkan kedalaman secara 3 dimensi. CNN mengambil gambar sebagai input dan mengeluarkan prediksi kedalaman 3 dimensi sebagai output (Khan, 2020). Godard (2019) mengembangkan metode estimasi kedalaman monokular berbasis arsitektur U-Net. Pada penelitian tersebut, beberapa model diintegrasikan untuk menghasilkan nilai estimasi kedalaman objek. Arsitektur CNN U-Net dapat dilihat pada gambar 2.1.



Gambar 2. 1 Arsitektur CNN U-Net

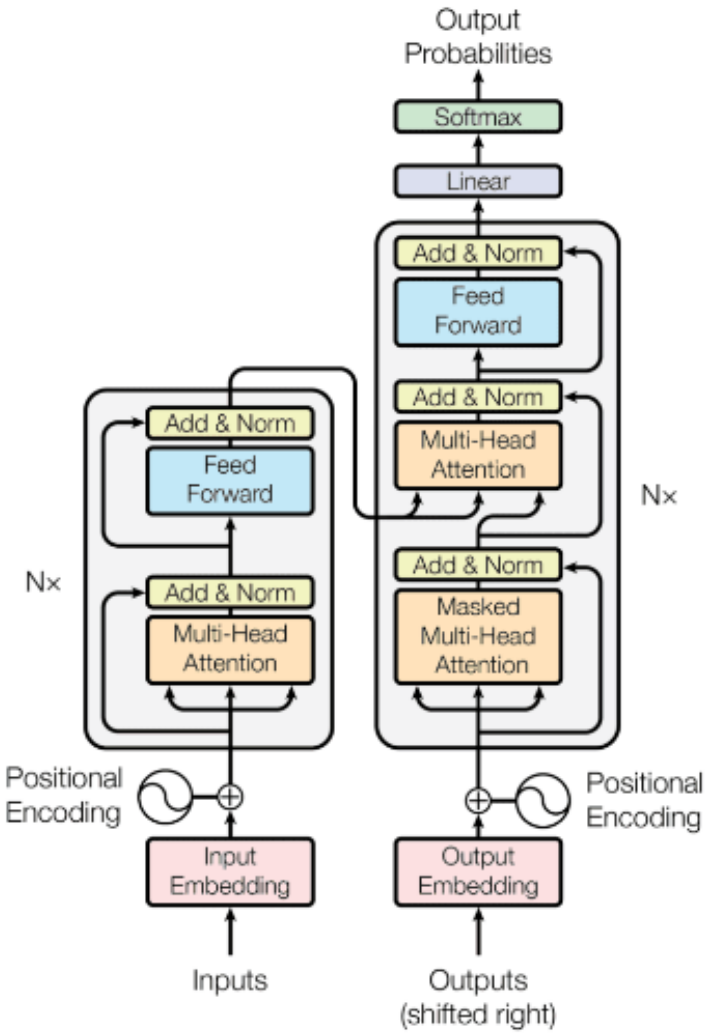
Arsitektur jaringan ini digambarkan dalam Gambar 2.1. Jaringan ini terdiri dari jalur kontraksi (sisi kiri) dan jalur ekspansi (sisi kanan). Jalur kontraksi mengikuti arsitektur jaringan konvolusi yang khas. Jalur ini terdiri dari pengulangan dua kali konvolusi 3x3 (konvolusi tanpa padding), masing-masing diikuti oleh unit linear ReLU dan operasi max pooling 2x2 dengan langkah 2 untuk melakukan downsampling. Pada setiap tahap downsampling, jumlah saluran fitur digandakan. Setiap tahap pada jalur ekspansi terdiri dari upsampling dari peta fitur, diikuti oleh konvolusi 2x2 ("up-convolution") yang mengurangi separuh jumlah saluran fitur, penyatuan dengan peta fitur yang sesuai dari jalur kontraksi, dan dua konvolusi 3x3, masing-masing diikuti oleh ReLU. Pemangkasan diperlukan karena adanya kehilangan piksel batas pada setiap konvolusi. Pada lapisan akhir, konvolusi 1x1 digunakan untuk memetakan setiap vektor fitur 64-komponen menjadi jumlah kelas yang diinginkan. Secara total, jaringan ini memiliki 23 lapisan konvolusi (Ronneberger, 2015).

2.8.2. Model CNN *Segment Anything*

Segment Anything Model (SAM) adalah model CNN yang dikembangkan oleh Meta AI Research yang mampu melakukan segmentasi terhadap objek apa pun

pada citra. SAM telah dilatih menggunakan dataset segmentasi yang sangat besar dengan lebih dari 1 miliar *mask* dan telah menunjukkan potensi luar biasa dalam berbagai aplikasi, termasuk segmentasi gambar, deteksi objek, dan ekstraksi otak. SAM juga telah dikombinasikan dengan model-model lain, seperti Grounding DINO, Stable Diffusion, dan ChatGPT, untuk menunjukkan keberagaman sebagai model dasar. Namun, SAM memiliki keterbatasan dalam mendeteksi objek transparan dan skenario yang menantang terkait kaca. Sebuah survei komprehensif tentang SAM telah dilakukan untuk memberikan wawasan mengenai aplikasi praktis, manfaat, dan keterbatasannya.

"*Segment Anything*" merupakan sebuah proyek dan model yang dikembangkan oleh Meta AI untuk segmentasi gambar. Tujuan dari proyek ini adalah menciptakan model yang dapat dengan akurat "memotong" atau melakukan segmentasi pada objek apa pun dalam sebuah gambar hanya dengan satu klik. Model *Segment Anything* (SAM) menggunakan berbagai input prompt, seperti titik atau kotak, untuk menghasilkan masker objek berkualitas tinggi untuk semua objek dalam gambar. Model ini telah dilatih menggunakan dataset berisi jutaan gambar dan miliaran masker, dan telah menunjukkan performa yang kuat pada berbagai tugas segmentasi. Model ini dapat digunakan untuk menghasilkan masker untuk objek tertentu atau untuk seluruh gambar. Proyek *Segment Anything* bertujuan untuk mendemokrasikan segmentasi gambar dan membuatnya lebih mudah diakses (Zhang, 2023). Model ini dikembangkan menggunakan arsitektur *transformer*.



Gambar 2. 2 Arsitektur Transformer pada model *Segment Anything*

2.8.3. Vincenty Formula

Rumus Vincenty adalah dua metode iteratif terkait yang digunakan dalam geodesi untuk menghitung jarak antara dua titik pada permukaan sebuah sferoid. Rumus ini dikembangkan oleh Thaddeus Vincenty pada tahun 1975 dan didasarkan pada asumsi bahwa bentuk Bumi adalah sebuah sferoid datar, sehingga membuatnya lebih akurat dibandingkan dengan metode yang menganggap Bumi sebagai bola, seperti jarak lingkaran besar. Rumus Vincenty digunakan untuk menghitung jarak antara dua titik pada permukaan sebuah sferoid, seperti Bumi. Rumus ini lebih akurat daripada metode yang menganggap Bumi sebagai bola. Rumus Vincenty adalah metode iteratif, yang berarti mereka menggunakan serangkaian pendekatan untuk mencapai jawaban akhir. Rumus ini dikembangkan

oleh Thaddeus Vincenty pada tahun 1975. Rumus Vincenty mengasumsikan bahwa Bumi adalah sferoid datar, yang merupakan representasi yang lebih akurat tentang bentuk Bumi daripada bola. Rumus ini sangat berguna untuk menghitung jarak dalam jarak yang panjang atau pada permukaan sferoid datar, seperti Bumi. Namun, perlu dicatat bahwa solusi inversi Vincenty dapat gagal pada titik yang hampir antipodal.

Menggunakan rumus Vincenty memungkinkan kita untuk menghitung latitude dan longitude tujuan dari suatu titik awal. Hal ini dimungkinkan karena rumus Vincenty didasarkan pada model matematika ellipsoid, yang merupakan representasi yang lebih akurat tentang bentuk Bumi daripada bola. Rumus ini memperhitungkan perataan Bumi di kutub dan pembengkokan di khatulistiwa, yang memengaruhi jarak antara dua titik di permukaan Bumi. Dengan menggunakan metode iteratif, rumus Vincenty dapat menghitung jarak antara dua titik di permukaan ellipsoid dengan akurasi tinggi. Ini memungkinkan penggunaan rumus Vincenty untuk menghitung latitude dan longitude tujuan dari suatu titik awal, dengan memberikan jarak yang ingin ditempuh (Kettle, 2017).

Dengan menggunakan titik awal (Φ1, L1) dan azimut awal, α1, serta jarak, s, sepanjang garis lintang, masalahnya adalah untuk mencari titik tujuan (Φ2, L2) dan azimut, α2. Mulai dengan melakukan perhitungan berikut.

$$U_1 = \arctan\left[(1-f)tan\phi_1\right. \tag{2-1}$$

$$\sigma_1 = \arctan2(tanU_1, \cos\alpha - f)tan \tag{2-2}$$

$$sin\ \alpha = cosU_1 sin\ \alpha_1 \tag{2-3}$$

$$u^2 = cos^2\ \alpha\left(\frac{a^2-b^2}{b^2}\right) = (1-sin^2\alpha)\left(\frac{a^2-b^2}{b^2}\right) \tag{2-4}$$

$$A = 1 + \frac{u^2}{16384}(4096 + u^2[-768 + u^2(320 - 175u^2)]) \tag{2-5}$$

$$B = \frac{u^2}{1024}(256 + u^2[-128 + u^2(74 - 47u^2)]) \tag{2-6}$$

Kemudian, dengan menggunakan nilai awal $\sigma = \frac{s}{bA}$, lakukan iterasi pada persamaan-persamaan berikut hingga tidak ada perubahan signifikan pada σ:

$$2\sigma_m = 2\sigma_1 + \sigma \tag{2-7}$$

$$\Delta\sigma = B\ sin\ \sigma\left\{cos(2\sigma_m)\right. \tag{2-8}$$

$$+\frac{1}{4}B\left(cos\sigma[-1+2cos^2(2\sigma_m)]\right.$$

$$\left.-\frac{B}{6}[2\sigma_m][-3+4sin^2\sigma][-3+4cos^2(2\sigma_m)]\right)\Big\}$$

$$\Delta\sigma = \frac{s}{bA} + \Delta\sigma \tag{2-9}$$

Setelah σ diperoleh dengan akurasi yang memadai, evaluasilah:

$$\phi_2 = arctan2\left(sinU_1 cos\sigma\right.$$

$$+ cos\ U_1 \sin\ \sigma\ cos\ \alpha_1, (1 \tag{2-10}$$

$$\left.- f)\sqrt{sin^2\alpha + (\sin\ U_1\ sin\ \sigma - \cos U_1 \cos\sigma\ cos\ \alpha_1)^2}\ \right)$$

$$\lambda = arctan2(\sin\sigma\ \ sin\ \alpha_1, \cos U_1 \cos\sigma - \sin U_1 \sin\sigma\ \ cos\ \alpha_1) \tag{2-11}$$

$$C = \frac{f}{16}cos^2\alpha[4 + f(4 - 3\ cos^2\alpha)] \tag{2-12}$$

$$L = \lambda - (1 - C)f sin\ \alpha\{\sigma \tag{2-13}$$

$$+ C\ sin\ \sigma(cos[2\sigma_m] + C\cos\sigma\ [= 1 + 2\ cos^2(2\sigma_m)])\}$$

$$L_2 = L + L_1$$

$$\alpha = arctan2(\sin\alpha, -sin\ U_1 \sin\sigma + cos\ U_1 \cos\sigma \cos\alpha_1) \tag{2-14}$$

Jika titik awal berada di Kutub Utara atau Kutub Selatan, maka persamaan pertama tidak dapat ditentukan. Jika azimut awal adalah Timur atau Barat, maka persamaan kedua tidak dapat ditentukan.

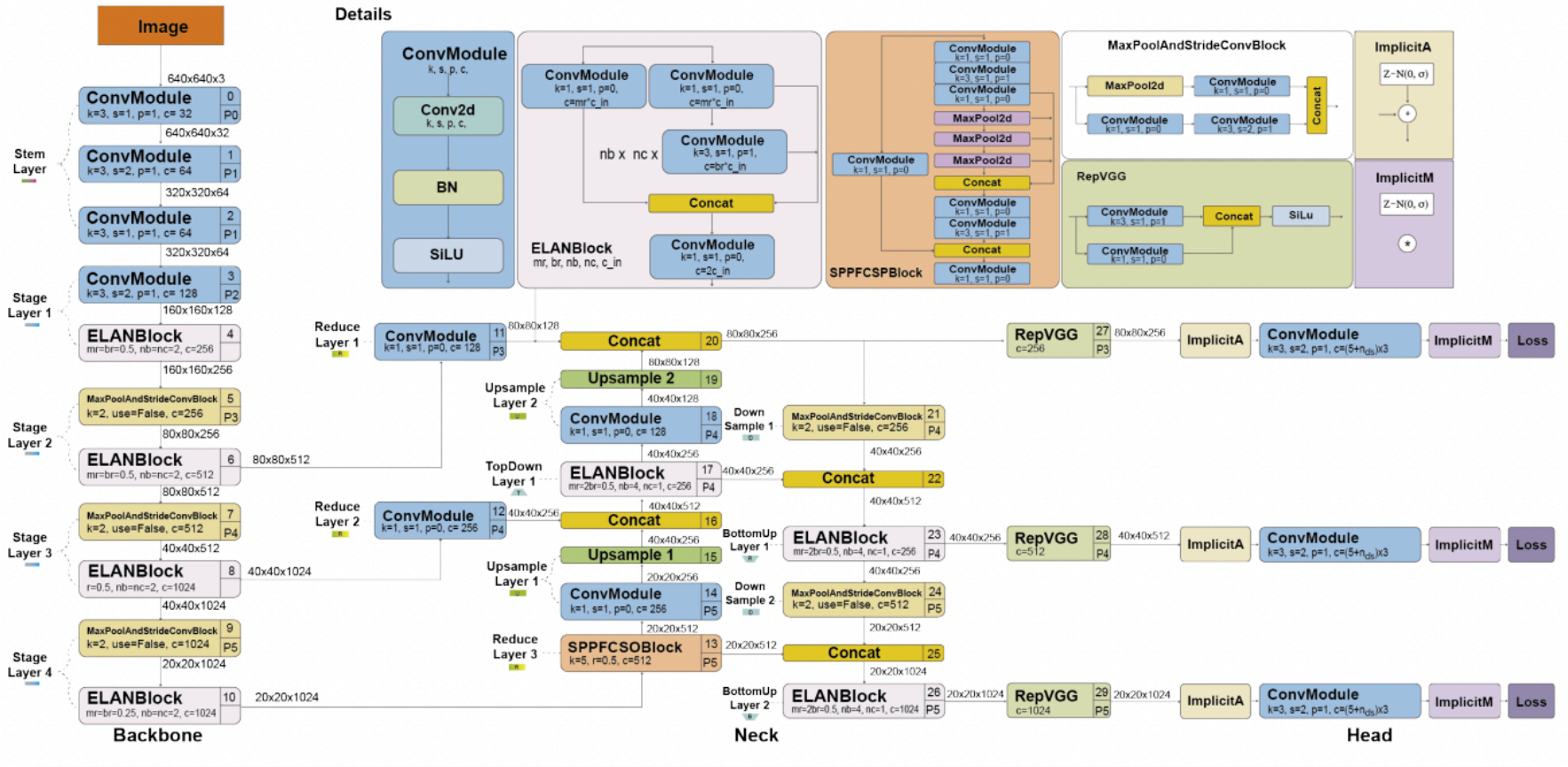
2.9. Sistem Deteksi Objek Buah Kakao menggunakan CNN YOLO

Pada penelitian ini akan dikembangkan model CNN untuk melakukan deteksi buah kakao pada tanaman kakao. Arsitektur model CNN yang digunakan pada penelitian ini adalah arsitektur YOLOV8. Arsitektur ini digunakan karena model yang dihasilkan memiliki akurasi deteksi yang baik dan dapat melakukan deteksi secara cepat.

3.9.1. Arsitektur YOLOV8

YOLOv8, yang dikembangkan oleh Ultralytics pada bulan Januari 2023 sebagai pengembangan dari YOLOv5, memperkenalkan beberapa versi dengan skala yang berbeda, mulai dari YOLOv8n (nano) hingga YOLOv8x (extra large). Model yang diperbarui ini mendukung beberapa tugas visi komputer seperti deteksi objek, segmentasi, estimasi pose, pelacakan, dan klasifikasi. Arsitektur YOLOv8 didasarkan pada YOLOv5 namun dengan beberapa modifikasi, terutama pada CSPLayer yang sekarang disebut sebagai modul C2f. Modul C2f menggabungkan fitur tingkat tinggi dengan informasi kontekstual untuk meningkatkan akurasi deteksi YOLOv8 menggunakan model tanpa anchor dengan kepala yang terpisah, memungkinkan pemrosesan independen untuk tugas objek, klasifikasi, dan regresi. Desain ini memungkinkan setiap cabang fokus pada tugasnya masing-masing dan meningkatkan akurasi keseluruhan model. Pada lapisan output, YOLOv8 menggunakan fungsi sigmoid untuk skor objek, yang mewakili probabilitas terdapatnya objek dalam sebuah bounding box. Fungsi softmax digunakan untuk probabilitas kelas, menunjukkan kemungkinan objek termasuk dalam kelas-kelas yang berbeda (Terven, 2023).

Fungsi *loss* pada YOLOv8 menggabungkan VFL Loss untuk klasifikasi dan DFL Loss+CIOU Loss untuk regresi. VFL Loss merupakan varian dari fungsi Focal Loss yang memberikan bobot lebih kepada contoh-contoh sulit dan mengurangi pengaruh contoh-contoh mudah. DFL Loss merupakan fungsi *loss* berbasis distribusi yang memodelkan distribusi koordinat bounding box dan memprediksi rata-rata dan variansinya. CIOU Loss merupakan varian dari fungsi Intersection over Union (IOU) yang memperhitungkan rasio aspek dan ukuran bounding box. Untuk *bounding box* dan area segmentasi, fungsi loss mempertimbangkan nilai *confidence bounding box* yang diprediksi dan IOU. Perhitungan loss digunakan untuk mengestimasi jumlah kesalahan, yang kemudian digunakan oleh optimizer untuk menyesuaikan bobot model. YOLOv8 menggunakan fungsi loss CIoU dan DFL untuk loss bounding box dan binary cross-entropy untuk loss klasifikasi. Fungsi-fungsi ini telah meningkatkan kinerja deteksi objek, terutama dalam menghadapi objek-objek kecil (Terven, 2023). Arsitektur model CNN YOLOv8 dapat dilihat pada gambar 2.3.



Gambar 2.3 Arsitektur YOLOV8

3.9.2. Arstitektur YOLOV8n

YOLOv8 nano adalah model deteksi objek yang ringan dan efisien yang dikembangkan oleh Ultralytics. Model ini didasarkan pada arsitektur *You Only Look Once* (YOLO) dan secara khusus dirancang untuk lingkungan dengan sumber daya terbatas dan daya komputasi terbatas, seperti perangkat edge dan sistem embedded. YOLOv8 nano mencapai keseimbangan yang baik antara akurasi dan kecepatan dengan menggunakan ukuran model yang lebih kecil dan mengoptimalkan arsitektur jaringan, sehingga mampu mendeteksi objek secara real-time. Meskipun ukurannya kompak, YOLOv8 nano tetap mempertahankan akurasi tinggi dengan memanfaatkan teknik canggih seperti deteksi bebas anchor dan feature pyramid networks. Hal ini menjadikannya cocok untuk berbagai aplikasi, termasuk robotika, sistem pengawasan, dan perangkat IoT. Arsitektur YOLOv8 nano dapat dilihat pada tabel 2.1.

Tabel 2. 1 Arsitektur YOLOV8n (nano)

layer num	from	n	params	modules	arguments
0	-1	1	464	ultralytics.nn.modules.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.SPPF	[256, 256, 5]

Tabel 2. 2 Arsitektur YOLOV8n (nano) (lanjutan)

layer num	from	n	params	modules	arguments
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.Concat	[1]
12	-1	1	148224	ultralytics.nn.modules.C2f	[384, 128, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.Concat	[1]
15	-1	1	37248	ultralytics.nn.modules.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.C2f	[192, 128, 1]
19	-1	1	147712	ultralytics.nn.modules.Conv	[128, 128, 3, 2]
20	[-1, 9]	1	0	ultralytics.nn.modules.Concat	[1]
21	-1	1	493056	ultralytics.nn.modules.C2f	[384, 256, 1]
22	[15, 18, 21]	1	751702	ultralytics.nn.modules.Detect	[2, [64, 128, 256]]

3.9.3. Arstitektur YOLOV8m

YOLOv8 Medium adalah model deteksi objek yang dikembangkan oleh Ultralytics, sebuah organisasi yang fokus pada visi komputer dan deep learning. Ini merupakan perluasan dari keluarga model populer You Only Look Once (YOLO). YOLOv8 Medium dirancang untuk mendeteksi dan lokaliser objek dengan efisien dan akurat dalam gambar atau frame video, sehingga cocok untuk aplikasi waktu nyata. Hal ini dicapai dengan membagi gambar masukan menjadi sebuah grid dan

memprediksi bounding box serta probabilitas kelas untuk objek dalam setiap sel grid. YOLOv8 Medium menggunakan arsitektur berukuran medium, menemukan keseimbangan antara kompleksitas model dan kecepatan, sehingga cocok untuk berbagai tugas visi komputer. Arsitektur model CNN YOLOv8 medium dapat dilihat pada tabel berikut.

Tabel 2. 3 Arsitektur YOLOV8m (medium)

layer	from	n	params	module	arguments
0	-1	1	1392	ultralytics.nn.modules.Conv	[3, 48, 3, 2]
1	-1	1	41664	ultralytics.nn.modules.Conv	[48, 96, 3, 2]
2	-1	2	111360	ultralytics.nn.modules.C2f	[96, 96, 2, True]
3	-1	1	166272	ultralytics.nn.modules.Conv	[96, 192, 3, 2]
4	-1	4	813312	ultralytics.nn.modules.C2f	[192, 192, 4, True]
5	-1	1	664320	ultralytics.nn.modules.Conv	[192, 384, 3, 2]
6	-1	4	3248640	ultralytics.nn.modules.C2f	[384, 384, 4, True]
7	-1	1	1991808	ultralytics.nn.modules.Conv	[384, 576, 3, 2]
8	-1	2	3985920	ultralytics.nn.modules.C2f	[576, 576, 2, True]
9	-1	1	831168	ultralytics.nn.modules.SPPF	[576, 576, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.Concat	[1]
12	-1	2	1993728	ultralytics.nn.modules.C2f	[960, 384, 2]

Tabel 2. 4 Arsitektur YOLOV8m (medium) (lanjutan)

layer	from	n	params	module	arguments	layer
13		-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1,	4]	1	0	ultralytics.nn.modules.Concat	[1]
15		-1	2	517632	ultralytics.nn.modules.C2f	[576, 192, 2]
16		-1	1	332160	ultralytics.nn.modules.Conv	[192, 192, 3, 2]
17	[-1, 1	2]	1	0	ultralytics.nn.modules.Concat	[1]
18		-1	2	1846272	ultralytics.nn.modules.C2f	[576, 384, 2]
19		-1	1	1327872	ultralytics.nn.modules.Conv	[384, 384, 3, 2]
20	[-1,	9]	1	0	ultralytics.nn.modules.Concat	[1]
21		-1	2	4207104	ultralytics.nn.modules.C2f	[960, 576, 2]
22	[15, 18, 2	1]	1	3776854	ultralytics.nn.modules.Detect	[2, [192, 384, 576]]

3.9.4. Metode Deteksi Objek Buah menggunakan YOLO

Fu (2022) mememperkenalkan metode baru untuk mendeteksi dan menghitung jumlah polong kedelai secara otomatis dan akurat di lapangan. Metode ini dapat mengatasi masalah efisiensi rendah, ketidakakuratan, dan ukuran sampel yang kecil pada pengumpulan fenotipe kedelai secara manual di lapangan. Metode

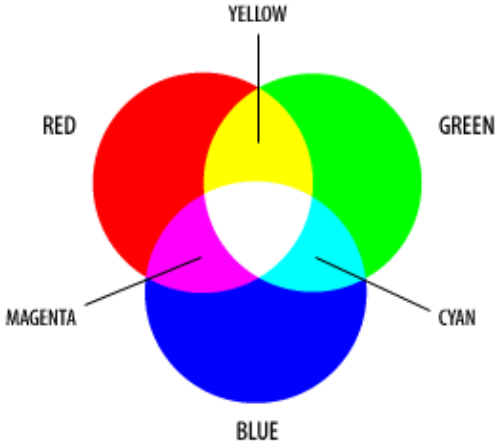
ini menggunakan kendaraan pemindaian tiga dimensi yang dikembangkan sendiri untuk memperoleh gambar warna RGB dan kedalaman polong kedelai di lapangan. Kemudian, gambar RGB dan kedalaman disesuaikan menggunakan metrik titik fitur tepi untuk mengidentifikasi dengan akurat polong kedelai di latar belakang lingkungan yang kompleks.

Model jaringan yang dilatih menggunakan dataset gabungan RGB dan kedalaman memberikan hasil yang lebih baik dibandingkan dengan model yang hanya dilatih dengan dataset RGB. Tingkat ketepatan (precision) model jaringan yang ditingkatkan YOLO-v5 juga meningkat sekitar 6%, dengan tingkat ketepatan mencapai 88.14% dalam mendeteksi jumlah polong di populasi kedelai dengan 200 tanaman. Setelah dilakukan kompensasi model, kesalahan relatif antara jumlah polong yang diprediksi dan yang sebenarnya hanya berkisar antara 2% hingga 3% untuk dua varietas kedelai yang diuji. Meskipun masih terdapat beberapa faktor lingkungan yang mempengaruhi deteksi dan kuantifikasi polong kedelai, metode ini merupakan langkah awal yang signifikan untuk memperoleh data fenotipe kedelai secara otomatis dan akurat di lapangan (Fu, 2022).

2.10. Red, Green, Blue (RGB)

Kusumanto *et al.* (2011) menjelaskan, RGB *(Red, Green, Blue)* adalah citra warna yang masing-masing memiliki warna tertentu yaitu merah, hijau dan biru. Masing-masing warna memiliki rentang intensitas 0 sampai dengan 255. Sehingga dari kombinasi 3 warna tersebut menghasilkan 256^3 kombinasi warna (16.777.216). Gupta *et al.* (2014) menyatakan citra RGB dapat digunakan untuk melakukan analisa pada tanaman. Dengan menggunakan citra RGB, dapat dilakukan metode yang bersifat non-destruktif untuk menganalisa/mengevaluasi kondisi tanaman. Weinstein *et al.* (2019) menjelaskan, dengan menggunakan citra lanskap alam berbasis RGB membuka banyak peluang baru dalam ekologi, perhutanan, serta pengelolaan lahan. Model CNN deteksi tanaman saat ini masih dapat diperluas kegunaannya. Tidak hanya untuk mendeteksi titik tanaman, apabila dikembangkan maka model CNN juga dapat mendeteksi kondisi kesehatan tanaman. Maraknya penggunaan UAV/*drone* dalam lingkungan *remote sensing,* juga membuka peluang untuk mengkombinasikan data-data yang diambil secara mandiri / data lokal

dengan data pada skala informasi yang lebih luas. Pada umumnya, *drone* konvensional hanya dapat menangkap reflektansi cahaya dalam format RGB. Data tersebut dapat dikombinasikan dengan data-data dari satelit yang menangkap informasi menggunakan sensor-sensor tertentu. Sensor-sensor tersebut dapat menghasilkan citra multi spektral. Selain spektrum cahaya warna RGB, sensor tersebut menangkap gelombang *Nir-Infrared*, penguapan air, serta gelombang *infrared* pendek.



Gambar 2. 4 Ruang Warna RGB (Nixsensor, 2022)

2.11. *Mean Absolute Error* (MAE)

Mean Absolute Error (MAE) merupakan parameter yang digunakan untuk mengevaluasi akurasi nilai yang di prediksi oleh sebuah model prediksi. MAE menunjukkan rata-rata kesalahan nilai aktual dengan nilai prediksi.

$$MAE = \sum \frac{|Y' - Y|}{n} \qquad (2\text{-}15)$$

Keterangan :

Y' : Nilai Prediksi

Y : Nilai Sebenarnya

n : Jumlah Data

2.12. *Mean Squared Error* (MSE)

Mean Square Error (MSE) adalah sebuah parameter yang mengukur kesalahan pada sebuah prediksi dengan menghitung rata-rata kesalahan kuadrat antara nilai sebenarnya dan nilai yang diprediksi. Dengan menggunakan metode ini, kita dapat memperkirakan seberapa besar kesalahan pada prediksi tersebut.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \qquad (2\text{-}16)$$

Keterangan :

Y' : Nilai Prediksi

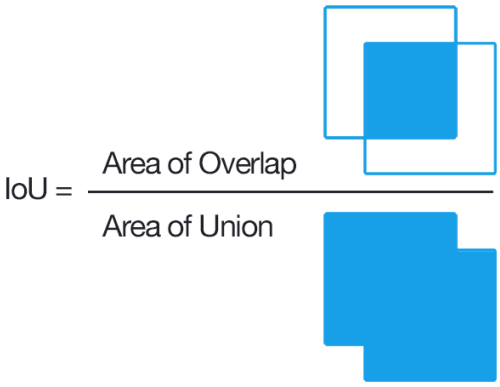
Y : Nilai Sebenarnya

n : Jumlah Data

2.13. *Intersection over Union* (IoU)

Intersection over Union (IoU) adalah metrik yang umum digunakan untuk mengevaluasi performa deteksi objek, segmentasi, dan tugas-tugas computer vision lainnya. IoU mengukur seberapa banyak area yang tumpang tindih antara bounding box atau masker prediksi dengan bounding box atau masker acuan. Untuk menghitung IoU, pertama-tama kita menghitung luas area yang tumpang tindih antara bounding box atau masker prediksi dan bounding box atau masker acuan. Kemudian, kita menghitung luas keseluruhan dari kedua bounding box atau masker tersebut. Akhirnya, kita membagi luas area yang tumpang tindih dengan luas area keseluruhan untuk mendapatkan skor IoU.

Skor IoU berkisar dari 0 hingga 1, dengan skor 1 menunjukkan tumpang tindih yang sempurna antara bounding box atau masker prediksi dan bounding box atau masker acuan, dan skor 0 menunjukkan tidak ada tumpang tindih. IoU sering digunakan sebagai metrik evaluasi dalam deteksi objek dan segmentasi karena memberikan ukuran kuantitatif seberapa baik model dapat melokalisasi dan mengsegmentasi objek dalam gambar. Skor IoU yang lebih tinggi menunjukkan bahwa model lebih baik dalam memprediksi lokasi dan ukuran yang benar dari objek pada gambar.



Gambar 2. 5 Penjelasan Intersection of Union

2.14. Perangkat Lunak

Perangkat lunak digunakan dalam pengembangan sistem atau model untuk memberikan alat yang diperlukan dalam proses desain, pengkodean, dan pengujian. Perangkat lunak seperti lingkungan pengembangan terintegrasi (IDE), framework, dan library menyediakan alat dan sumber daya yang diperlukan untuk mengimplementasikan sistem atau model secara efisien. Mereka juga membantu dalam mengelola data, mengoptimalkan kinerja, dan mempermudah proses pengembangan secara keseluruhan.

2.15.1 Python

Bahasa pemrograman python menggunakan perintah dalam bahasa inggris dan sintaks yang mudah dimengerti. Python menawarkan alternatif *open-source* untuk teknik-teknik tradisional dan aplikasi (Sahoo *et al, 2019)*. Python memiliki pilihan *library standard* yang besar. *Library-library* tersebut berfokus pada *general programming,* serta memuat modul-modul untuk berinteraksi dengan sistem operasi, jaringan, basis data, pengolahan citra digital serta keperluan spesifik lainnya (Ozgur *et al,* 2017). Python menyediakan banyak pilihan struktur data tingkat tinggi. Beberapa contohnya yaitu *list* untuk melakukan numerasi pada sebuah koleksi objek, *dictionary* untuk membangun *hash tables* dan lainnya. Bagaimanapun, struktur data diatas tidak sepenuhnya ideal untuk melakukan komputasi numerikal dengan performa tinggi (Walt *et al,* 2011).

Gambar 2. 6 Logo Bahasa Pemrograman Python

(Python, 2022)

2.15.2 Google Colaboratory

Google Colaboratory atau yang dikenal secara umum Google Colab merupakan layanan *open source* yang disediakan oleh google kepada semua pengguna layanan akun gmail. Google Colab menyediakan GPU (Unit Pemrosesan Grafis) untuk melakukan riset. Layanan ini ditujukan bagi orang-orang yang tidak memiliki sumber daya GPU untuk melakukan komputasi tingkat tinggi. Layanan Google Colab menyediakan RAM sebesar 12,72 GB dan ruang penyimpanan *hard disk* sebesar 358,27 GB dalam 1 *runtime*. Setiap *runtime* berlangsung selama 12 jam, setelah itu *runtime* akan ter-*reset* dan pengguna perlu melakukan koneksi ulang. Hal ini diberlakukan untuk memastikan bahwa layanan GPU tidak digunakan untuk melakukan penambangan mata uang kripto dan tujuan illegal lainnya. Setelah pengguna membuka Google Colab, pengguna perlu memilih jenis *runtime*. Terdapat 3 pilihan *runtime*, yaitu n*one*, GPU, TPU. *None* artinya *runtime* hanya akan menggunakan CPU pada komputer pengguna. GPU artinya *runtime* akan menggunakan GPU di dalam server Google. TPU, digunakan untuk melakukan proses tensor (Kanani *et al,* 2019).

Gambar 2. 7 Logo Google Colaboratory

(Google Colab, 2017)

2.15.3 Numpy

Numpy adalah sebuah *library* numerikal *Python* yang secara efisien memanipulasi *array* besar (Drude *et al,* 2018). Pada pertengahan 90an, sebuah tim

internasional yang terdiri dari relawan-relawan memulai pengembangan sebuah struktur data untuk melakukan komputasi *array* dengan efisien. Struktur ini berkembang menjadi apa yang saat ini kita kenali sebagai *N-dimensional Numpy array. Library Numpy* yang terdiri dari berbagai gabungan fungsi matematis. *Library* tersebut telah dimanfaatkan pada berbagai bidang seperti akademis, laboratorium nasional, serta berbagai implementasi di industri yang tersebar mulai dari industri *gaming* hingga eksplorasi antariksa (Walt, 2011). *Array NumPy* merupakan sebuah koleksi elemen serupa dalam multi dimensi. Sebuah *Array* digambarkan oleh tipe elemen didalamnya serta oleh bentuknya. Sebagai contoh, sebuah matriks dapat direpresentasikan sebagai sebuah array yang berbentuk (M x N) yang mengandung angka-angka, nilai desimal atau bilangan kompleks. Namun, tidak seperti matriks, *array Numpy* dapat memiliki berbagai dimensi. Lebih jauh lagi, *array* tersebut dapat memuat berbagai jenis elemen lainnya (bahkan kombinasi beberapa elemen) seperti boolean atau tanggal. *Array NumPy* merupakan metode yang cukup mudah untuk mendekripsikan satu atau lebih blok memori komputer sehingga angka-angka yang direpresentasikan dapat dengan mudah dimanipulasi.

Gambar 2. 8 Logo Library Numpy

(Numpy, 2022)

2.15.4 OpenCV

OpenCV merupakan sebuah *library* penglihatan komputer. Pengembangan OpenCV dimulai sebagai sebuah projek riset di Intel pada 1998. OpenCV sudah dapat digunakan pada tahun 2000 dibawah lisensi *open source* BSD. OpenCV bertujuan untuk menyediakan perangkat lunak yang diperlukan untuk menyelesaikan permasalahan penglihatan *computer*. Didalam *library* OpenCV terdapat gabungan dari fungsi pemrosesan gambar tingkat rendah dan algoritma tingkat tinggi seperti deteksi wajah, deteksi pejalan kaki, pencocokan fitur dan

pelacakan. Library tersebut telah diunduh sebanyak lebih dari 3 juta kali. Pada 2010 sebuah modul baru yang menyediakan akselerasi GPU ditambahkan ke OpenCV. Modul GPU tersebut mencakup bagian signifikan dari fungsionalitas library dan masih aktif dalam pengembangan. Modul tersebut mengimplementasikan penggunaan CUDA (Pulli *et al,* 2012).

Gambar 2. 9 Logo Library OpenCV

(OpenCV, 2022)

2.15.5 Pandas

Library Pandas, telah dikembangkan semenjak 2008. *Library Pandas* bertujuan untuk menjembatani banyaknya perangkat lunak analisis data dalam *Python*. Pandas tidak hanya bertujuan untuk menyediakan fungsionalitas sebagai pembaca data, namun juga menyediakan banyak fitur seperti penyelarasan data otomatis dan pengindeksan hierarkis. Dimana fitur-fitur tersebut tidak terintegrasi dalam *library* lainnya ataupun lingkungan komputasi lainnya. Selagi dikembangkan untuk analisis data finansial, pengembang berharap *Pandas* dapat akan memungkinkan *Python* saintifik menjadi lebih atraktif serta menjadi lingkungan komputasi statistik yang praktis bagi praktisi akademis dan industri. Nama *Pandas* berasal dari panel data, sebuah istilah umum untuk dataset multidimensi dalam *statistic dan ekonometriks* (Walt, 2011).

Pandas, 2022

Gambar 2. 10 Logo Library Pandas

2.15.6 Matplotlib & Seaborn

Matplotlib merupakan salah satu *library* Python untuk melakukan visualisasi data yang cukup popular. *Library* ini dibangun oleh John Hunter bersama beberapa kontributor. Matplotlib merupakan *library* grafis untuk melakukan visualisasi data dalam Python. Matplotlib dapat digunakan dengan beberapa *library* yang umum digunakan dalam pengolahan data pada Python seperti Numpy, Pandas dan *library* lainnya (Sial *et al,* 2021). Stančin *et al,* (2019) mendefinisikan Matplotlib sebagai *library* Python yang mengimplementasikan grafik-grafik yang ada didalam MATLAB. Matplotlib menawarkan banyak variasi dan penyesuaian sesuai kebutuhan pengguna. Sintaks Matplotlib cukup membingungkan bagi pemula, namun setelah memahami konsep utamanya akan jadi mudah untuk membuat berbagai jenis grafik.

Seaborn merupakan *library* yang dikembangkan di atas *library* Matplotlib dan lebih mudah untuk digunakan dan dipelajari bagi pemula ketimbang Matplotlib. Meskipun lebih mudah untuk digunakan, pada kasus yang memerlukan penyesuaian dan keperluan yang lebih kompleks Seaborn akan menjadi pilihan yang kurang tepat (Stančin *et al,* 2019).

Gambar 2. 11 Logo Library Matplotlib

(matplotlib, 2022)

Gambar 2. 12 Logo Library Seaborn

(Seaborn, 2022)

2.15.7 Visual Studio Code

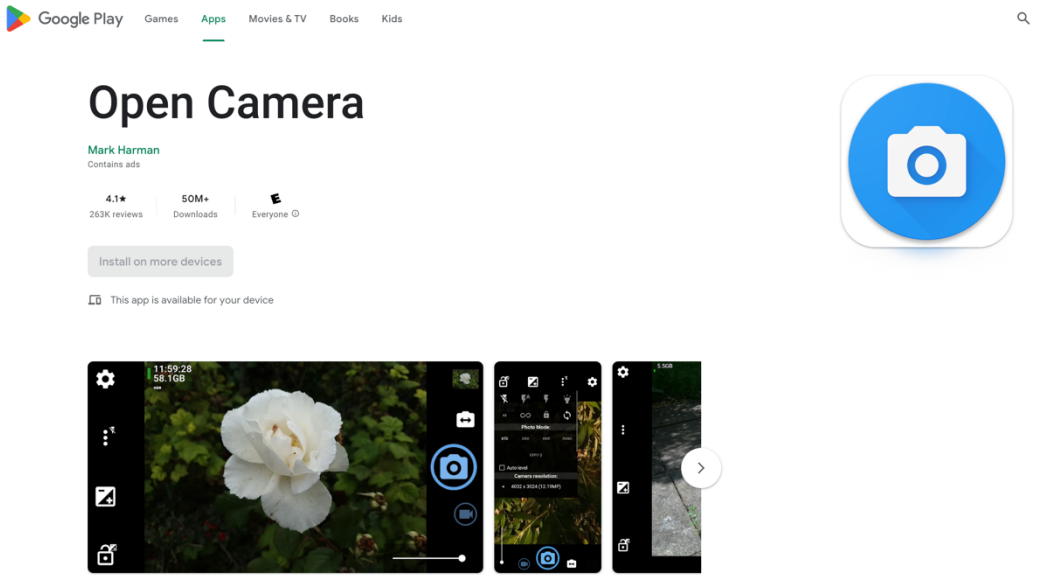
Visual Studio Code (VS Code) adalah sebuah teks editor ringan dan handal yang dibuat oleh Microsoft yang bersifat multiplatform. Artinya VS Code dapat berjalan pada sistem operasi Linux, Mac dan Windows. Teks editor ini secara langsung mendukung Bahasa pemrograman JavaScript, Typescript, dan Node.js, serta Bahasa lainnya (seperti C++, C#, Python, Go, Java) dengan bantuan *plugin* yang dapat dipasang via *marketplace Visual Studio Code.* Terdapat banyak fitur-fitur yang disediakan oleh *Visual Studio Code*, diantaranya Intellisense, Git Integration, Debugging, dan fitur ekstensi yang menambah kemampuan teks editor. Fitur-fitur tersebut akan terus bertambah seiring dengan bertambahnya versi *Visual Studio Code.* VS Code bersifat *open source*, artinya sumber kodenya dapat dilihat dan dikembangkan oleh semua orang. Hal ini merupakan daya tarik tersendiri bagi pengembang aplikasi karena dapat ikut serta dalam pengembangan VS Code (Salamah 2021).

Gambar 2. 13 Logo dan Tampilan Perangkat Lunak VS Code
(Microsoft, 2022)

2.15.8 Open Camera

Open Camera adalah aplikasi kamera open source untuk ponsel dan tablet android yang memiliki berbagai fitur. Dalam aplikasi ini, terdapat opsi untuk menjaga foto tetap seimbang dan tidak miring, serta menambahkan berbagai mode pengambilan gambar, efek warna, keseimbangan warna, ISO, kunci eksposur, selfie dengan "flash layar", dan video HD. Selain timer dengan suara penghitung mundur, pengulangan otomatis dengan penundaan yang dapat dikonfigurasi, dan opsi untuk mengambil foto dari jauh dengan suara. Open Camera juga menyediakan fitur untuk

menambahkan bingkai grid dan panduan potong, serta opsi penandaan lokasi GPS (geotagging) pada foto dan video. Aplikasi ini juga memiliki dukungan untuk HDR. Selain itu, terdapat opsi untuk menghapus metadata exif perangkat dari foto, panorama, reduksi kebisingan, dan mode optimasi rentang dinamis. Open Camera adalah aplikasi yang gratis dan tanpa iklan di dalamnya serta sumber terbuka (OpenCamera, 2021).



Gambar 2. 14 Halaman Aplikasi *Open Camera* di *Google Play* (OpenCamera, 2021)

2.15.9 Pytorch

PyTorch adalah sebuah *library machine learning* yang dirancang untuk mendukung gaya pemrograman yang imperatif. PyTorch memudahkan pemodelan kode, memudahkan proses debugging, dan konsisten dengan *library* komputasi ilmiah populer lainnya. Selain itu, PyTorch tetap efisien dan mendukung akselerator perangkat keras seperti GPU. PyTorch merupakan *framework deep learning* yang menyediakan operasi kustom, lapisan, model, dan alat untuk penelitian, pengembangan, dan evaluasi codec kompresi gambar dan video end-to-end. PyTorch dibangun di atas beberapa proyek, terutama Lua Torch, Chainer, dan HIPS Autograd. PyTorch menyediakan lingkungan berkinerja tinggi dengan akses mudah ke diferensiasi otomatis pada model yang dieksekusi di perangkat yang berbeda, seperti CPU dan GPU. Selain itu, PyTorch juga digunakan dalam bidang-bidang

lain seperti serangan dan pertahanan adversarial, rekonstruksi jejak partikel menggunakan pembelajaran mendalam, dan mempercepat penelitian pembelajaran mendalam 3D (Paszke, 2019).

2.15.10 Torchvision

Torchvision adalah sebuah *libray* PyTorch yang menyediakan berbagai alat dan model terkait visi komputer untuk para peneliti dan praktisi di bidang tersebut. *Library* ini mencakup dataset, data loader, transformasi, model, dan utilitas untuk tugas-tugas umum dalam visi komputer seperti klasifikasi gambar, deteksi objek, segmentasi semantik, dan lainnya. Torchvision menyediakan model-model yang telah dilatih sebelumnya untuk tugas klasifikasi gambar dan deteksi objek, seperti AlexNet, VGG, ResNet, dan Faster R-CNN, di antara lainnya. Selain itu, Torchvision juga menyediakan berbagai teknik augmentasi data, seperti pemotongan acak, pembalikan, dan variasi warna, untuk membantu meningkatkan generalisasi model (Jatavallabhula, 2019).

2.15.11 Glob

Modul glob dalam Python digunakan untuk mencari semua nama alamat direktori yang cocok dengan pola yang ditentukan sesuai dengan aturan yang digunakan oleh shell Unix. Modul ini mengembalikan daftar nama alamat direktori yang cocok dengan pola yang ditentukan, yang kemudian dapat digunakan untuk melakukan berbagai operasi pada file-file tersebut. Modul glob mendukung berbagai pola yang dapat digunakan untuk mencocokkan nama file, seperti * untuk mencocokkan string karakter apa pun, ? untuk mencocokkan satu karakter saja, dan [] untuk mencocokkan karakter apa pun dalam set yang ditentukan. Perlu diperhatikan bahwa glob hanya mengembalikan nama jalur file yang ada dan dapat dibaca (Virtanen, 2019).

2.15.12 PIL

Python Imaging Library (PIL) adalah sebuah library untuk bekerja dengan gambar dalam bahasa Python. Library ini menyediakan berbagai fungsi pemrosesan gambar, seperti pengubah ukuran, pemotongan, rotasi, dan penyaringan, serta

dukungan untuk berbagai format file gambar. PIL adalah library populer untuk pemrosesan gambar dalam Python dan telah digunakan dalam berbagai aplikasi, termasuk visi komputer, citra ilmiah, dan pengembangan web. Namun, PIL tidak lagi aktif dipelihara dan telah digantikan oleh library Pillow, yang merupakan cabang dari PIL yang menyediakan fitur tambahan dan perbaikan bug. Pillow dirancang sebagai pengganti PIL yang kompatibel dan menyediakan API yang serupa, sehingga mudah beralih dari PIL ke Pillow (Guan, 2019).

2.15.13 Onnxruntime

ONNX Runtime adalah mesin inferensi sumber terbuka yang dirancang untuk menjalankan model pembelajaran mesin yang sesuai dengan format *Open Neural Network Exchange* (ONNX). ONNX Runtime dirancang untuk memberikan eksekusi yang efisien dan portabel dari model pembelajaran mesin pada berbagai platform perangkat keras, termasuk CPU, GPU, dan akselerator khusus. ONNX Runtime mendukung berbagai bahasa pemrograman, termasuk Python, C++, dan C#, dan dapat diintegrasikan dengan framework pembelajaran mesin populer seperti PyTorch dan TensorFlow. ONNX Runtime menyediakan serangkaian API yang memungkinkan pengembang untuk memuat, menjalankan, dan mengelola model pembelajaran mesin, serta alat-alat untuk mengoptimalkan dan memproses kinerja model. ONNX Runtime dioptimalkan secara khusus untuk inferensi dengan latensi rendah dan mendukung berbagai backend dan metode optimasi (Ashfaq, 2022).

2.15.14 Ipython.display

Modul IPython.display dalam IPython menyediakan sejumlah fungsi yang sangat berguna untuk menampilkan berbagai jenis konten di Jupyter Notebook. Fungsi "display" digunakan untuk menampilkan objek dengan representasi terbaik yang tersedia di Jupyter Notebook. Misalnya, jika kita ingin menampilkan gambar, kita dapat menggunakan fungsi "Image" untuk menampilkan gambar di dalam notebook. Fungsi "Video" memungkinkan kita untuk menampilkan video di dalam notebook, sementara fungsi "Audio" digunakan untuk menampilkan pemutar audio yang memungkinkan kita untuk memainkan file audio langsung di dalam notebook.

Selain itu, modul IPython.display juga menyediakan fungsi "HTML" yang memungkinkan kita untuk menampilkan kode HTML di dalam notebook, dan fungsi "Markdown" yang memungkinkan kita untuk menampilkan kode Markdown yang akan ditafsirkan dan ditampilkan sebagai teks yang diformat dengan baik di dalam notebook. Semua fungsi ini membantu dalam membuat tampilan yang menarik dan interaktif di Jupyter Notebook (Ipython, 2019).

2.15.15 Time

Package `time` dalam bahasa pemrograman Python adalah sebuah *package* yang menyediakan fungsionalitas untuk mengakses waktu sistem dan melakukan operasi terkait waktu. *Package* ini memungkinkan pengembang untuk mengukur waktu eksekusi program, mengatur jeda atau penundaan dalam eksekusi program, dan melakukan operasi lainnya terkait waktu seperti mengubah format waktu, menghitung selisih waktu, dan mengatur waktu sistem. *Package* `time` sangat berguna dalam pengembangan aplikasi yang memerlukan pemantauan waktu, pengukuran kinerja, sinkronisasi tugas, atau manipulasi waktu secara umum (Python, 2023).

2.15.16 Ultralytics

Ultralytics adalah sebuah perusahaan teknologi yang mengkhususkan diri dalam pengembangan perangkat lunak komputer visi komputer berbasis Deep Learning dan deteksi objek real-time. Perusahaan ini terkenal karena library perangkat lunak YOLO (You Only Look Once) yang mereka kembangkan. YOLO merupakan salah satu pendekatan populer dalam deteksi objek yang memungkinkan pengguna untuk melakukan deteksi objek secara cepat dan akurat dalam aplikasi real-time. Ultralytics menyediakan perangkat lunak dan sumber daya yang membantu pengembang dan peneliti dalam mengimplementasikan deteksi objek menggunakan YOLO, serta terus mengembangkan dan meningkatkan kinerja model tersebut. (Ultralytics, 2021).

Gambar 2. 15 Logo ultralytics

(Ultralytics, 2021)

2.15.17 Sys

Package `sys` dalam bahasa pemrograman Python adalah sebuah paket yang menyediakan akses ke fungsi dan variabel yang terkait dengan interpreter Python dan lingkungan sistem. Dengan *package* `sys`, pengembang dapat mengakses argumen baris perintah, mengelola jalur modul, mengontrol perilaku program, dan mendapatkan informasi tentang sistem operasi yang digunakan (Python, 2023).

Gambar 2. 16 Logo modul sys python

(Python, 2023)

2.15.18 Scipy

Scipy adalah sebuah library perangkat lunak open-source untuk bahasa pemrograman Python yang digunakan untuk komputasi ilmiah dan analisis data. Library ini menyediakan berbagai algoritma dan fungsi matematika yang kuat, termasuk optimisasi, integrasi numerik, transformasi Fourier, aljabar linear, statistik, pemrosesan sinyal, dan banyak lagi. Scipy memperluas fungsionalitas Python standar dengan menambahkan kemampuan komputasi numerik yang canggih, yang sangat berguna dalam penelitian ilmiah, analisis data, dan

pemodelan. Scipy digunakan secara luas dalam berbagai disiplin ilmu, seperti fisika, biologi, ekonomi, ilmu komputer, dan lain-lain (Jones, 2001).

Gambar 2. 17 Logo Library SciPy

2.15.19 Csv

CSV dalam bahasa pemrograman Python adalah sebuah paket yang menyediakan fungsi-fungsi untuk membaca dan menulis file dalam format Comma-Separated Values (CSV). CSV merupakan format yang umum digunakan untuk menyimpan data tabular, di mana nilai-nilai dalam setiap baris dipisahkan oleh tanda koma. *Package* CSV memudahkan pengembang dalam memanipulasi file CSV dengan menyediakan metode untuk membaca data dari file CSV ke dalam struktur data Python, serta menulis data dari struktur data Python ke dalam file CSV. Dengan menggunakan *package* CSV, pengembang dapat dengan mudah melakukan operasi seperti membaca, mengubah, atau menyimpan data dalam format CSV dengan cepat dan efisien (A. Junaidi,2017).

2.15.20 Os

Package `os` dalam bahasa pemrograman Python adalah sebuah paket yang menyediakan fungsionalitas untuk berinteraksi dengan sistem operasi yang digunakan oleh komputer. Paket ini memungkinkan pengembang untuk melakukan berbagai operasi terkait sistem operasi, termasuk mengakses file dan direktori, mengatur variabel lingkungan, menjalankan perintah shell, dan banyak lagi. Dengan *package* `os`, pengembang dapat dengan mudah mengelola file, melakukan manipulasi direktori, dan mengatur variabel lingkungan melalui bahasa pemrograman Python. *Package* `os` merupakan alat yang penting dalam pengembangan aplikasi yang melibatkan operasi system (Sridianti, 2022).

Gambar 2. 18 Logo Modul OS Python

2.15.21 Warnings

Package `warnings` dalam bahasa pemrograman Python adalah sebuah paket yang digunakan untuk mengelola dan mengontrol peringatan (warnings) yang muncul selama eksekusi program. Ketika suatu potensi masalah atau situasi yang tidak diharapkan terjadi selama proses eksekusi program, *package* `warnings` memungkinkan pengembang untuk memberikan peringatan kepada pengguna atau pengembang lain tentang situasi tersebut. Dengan *package* `warnings`, pengembang dapat mengatur tindakan yang diambil ketika peringatan muncul, seperti menampilkan pesan peringatan, mengabaikan peringatan, atau mengubahnya menjadi pengecualian (exception). *Package* `warnings` sangat berguna dalam pemeliharaan dan debug program, membantu pengembang untuk memperbaiki potensi masalah dan meningkatkan kualitas dan keandalan aplikasi (Python, 2023).

2.15.22 Keras

Package `keras` dalam bahasa pemrograman Python adalah sebuah paket yang populer dan kuat untuk membangun dan melatih model jaringan saraf (neural network). Keras menyediakan antarmuka tingkat tinggi yang user-friendly untuk merancang dan mengimplementasikan berbagai jenis arsitektur jaringan saraf seperti jaringan saraf konvolusional (CNN), jaringan saraf rekurens (RNN), dan jaringan saraf yang dikombinasikan. Keras menyediakan beragam lapisan (layer), fungsi aktivasi, algoritma optimisasi, dan metrik evaluasi yang dapat digunakan dengan mudah untuk mengonstruksi model yang kompleks. Selain itu, Keras juga

menyediakan kemampuan untuk melatih dan menguji model dengan kumpulan data yang diberikan. Dengan pendekatan yang modular dan fleksibel, Keras mempermudah para pengembang dalam melakukan eksperimen, penyesuaian, dan peningkatan model jaringan saraf. (Chollet, 2015).

2.15.23 Datetime

Package `datetime` dalam bahasa pemrograman Python adalah sebuah paket yang menyediakan fungsionalitas untuk mengelola, memanipulasi, dan bekerja dengan tanggal (date) dan waktu (time). *Package* ini menyediakan kelas-kelas seperti `datetime`, `date`, `time`, `timedelta`, dan `tzinfo` yang memungkinkan pengembang untuk melakukan operasi seperti membuat objek tanggal dan waktu, mengekstrak komponen tanggal dan waktu (seperti tahun, bulan, hari, jam, menit, dan detik), melakukan operasi aritmatika pada tanggal dan waktu, memformat dan memparse tanggal dan waktu dalam berbagai format, serta mengubah zona waktu. *Package* `datetime` sangat berguna dalam pengembangan aplikasi yang memerlukan manipulasi dan pengaturan tanggal dan waktu dengan presisi dan keakuratan (Rosihan, 2018).

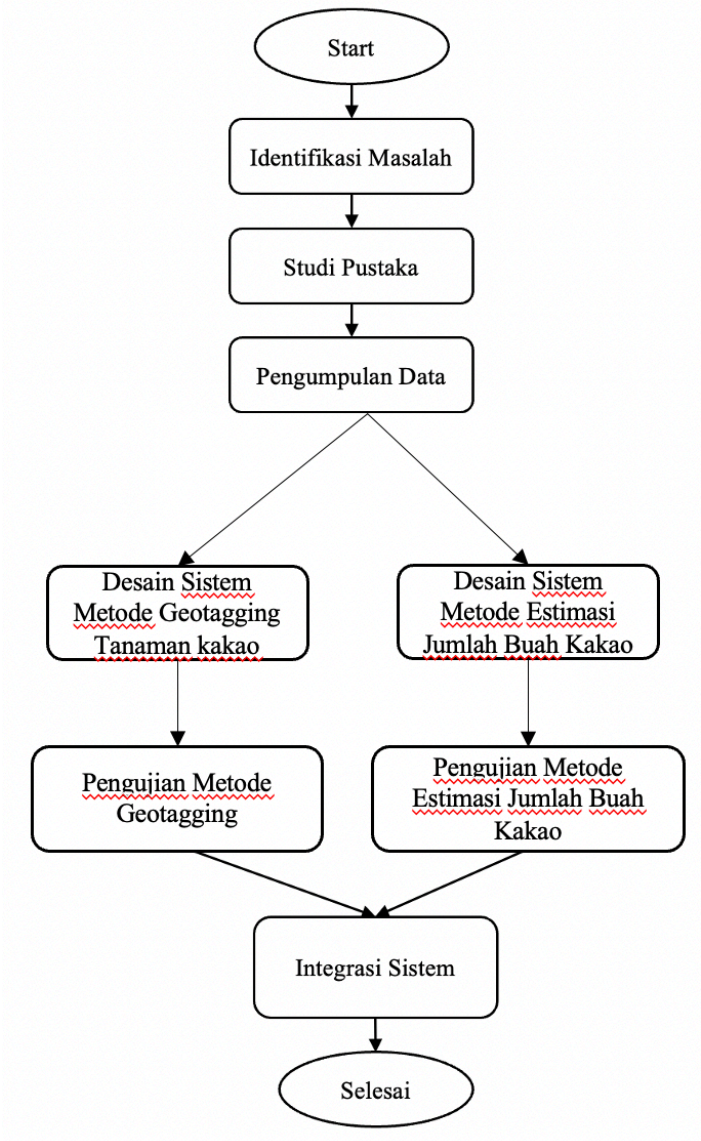
2.15.24 Math

Package `math` dalam bahasa pemrograman Python adalah sebuah paket yang menyediakan berbagai fungsi matematika yang umum digunakan. *Package* ini memberikan akses ke berbagai fungsi matematika dasar seperti trigonometri, logaritma, eksponensial, akar kuadrat, pembulatan, dan banyak lagi. Dengan *package* `math`, pengembang dapat melakukan operasi matematika kompleks dengan mudah, *package* ini menjadi alat yang penting dalam pengembangan aplikasi yang memerlukan manipulasi dan perhitungan matematika, seperti fisika, statistik, grafika, atau pemodelan matematika (Python, 2023).

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1. Alur Penelitian

Proyek Tugas Akhir ini bertujuan untuk mengembangkan metode *Geo-tagging* menggunakan *Artificial Neural Network* dengan memprediksi koordinat tanaman kakao pada citra serta mendeteksi buah kakao untuk melakukan penghitungan otomatis buah pada tanaman kakao. Alur penelitian dari proyek ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

Penelitian akan dimulai dengan mengidentifikasi masalah, dengan tujuan agar peneliti dapat mengetahui permasalahan yang dibahas pada penelitian ini. Selanjutnya, akan dilakukan studi pustaka untuk mempelajari penelitian beberapa tahun terakhir terkait topik penelitian ini. Beberapa dasar-dasar teori yang berhubungan juga akan dipelajari untuk mengetahui metode yang tepat untuk menyelesaikan permasalahan. Pada tahap selanjutnya, dilakukan pengumpulan data-data citra tanaman kakao yang akan digunakan untuk menguji akurasi model estimasi jarak tanaman kakao dan membangun model estimasi koordinat tanaman kakao berbasis *Artificial Neural Network* (ANN) untuk metode Geotagging. Kemudian citra-citra tersebut digunakan pula untuk melakukan pelatihan model CNN untuk mendeteksi buah kakao yang ada pada tanaman, untuk dilakukan penghitungan buah pada setiap tanaman kakao.

3.2. Analisis Kebutuhan

Pengembangan metode geotagging yang dikerjakan membutuhkan analisis agar penelitian berjalan dengan baik dan lancar. Analisis yang dilakukan untuk perancangan dan pengembangan model geotagging tanaman kakao, kebutuhan perangkat keras dan perangkat lunak pada sisi pengguna dan peneliti.

3.2.1. Kebutuhan Pengguna

Analisis kebutuhan diperoleh berdasarkan tujuan penelitian yaitu pengembangan metode geotagging pada tanaman kakao. Pengembangan model geotagging untuk melakukan estimasi koordinat tanaman kakao dapat digunakan pada aplikasi mini berbasis web. Model Geotagging akan menerima input citra yang memiliki informasi koordinat pengambilan gambar. Lalu model akan melakukan estimasi jarak tanaman kakao pada citra. Setelah itu, arsitektur ANN akan melakukan estimasi koordinat tanaman kakao berdasarkan koordinat pengambilan citra dan jarak tanaman kakao. Pengguna dapat menggunakan *smartphone* untuk mengambil citra tanaman kakao. Lalu hasil estimasi koordinat tanaman kakao akan ditampilkan. Pada penelitian ini akan dikembangkan pula metode kuantifikasi otomatis untuk mendapatkan jumlah buah pada tanaman kakao. Teknologi yang

akan digunakan yaitu CNN YOLO untuk mendeteksi buah pada tanaman kakao. Sehingga buah-buah yang terdeteksi akan dihitung, lalu akan ditampilkan jumlah buah pada tanaman kakao.

3.2.2. Kebutuhan Peneliti

Berikut terdapat beberapa perangkat keras dan perangkat lunak yang digunakan oleh peneliti dalam melakukan penelitian ini.

1. Perangkat Keras
    a. Laptop Mac Book Air M1 2020
        i. Prosesor : Apple M1
        ii. RAM : 8 GB
        iii. SSD : 512 GB
        iv. Sistem Operasi : macOS Monterey
    b. Smartphone Samsung Galaxy A6 2018
        i. Prosesor : Exynos 7870 Octa
        ii. RAM : 3 GB
        iii. *Internal Storage* : 32 GB
        iv. Lensa Kamera : 16 MP, f/1.7, 26mm (wide)
        v. Sistem Operasi : Android 8.0 (Oreo)
2. Perangkat Lunak
    a. Python 3
    b. Google Colaboratory
    c. Visual Studio Code

3.3. Identifikasi Masalah

Masalah utama sistem monitoring perkebunan kakao saat ini adalah, dengan menggunakan foto udara (aerial) tidak banyak informasi terkait tanaman kakao yang dapat diekstrak. Karena, apabila dilihat dari atas kebanyakan tanaman kakao tertutup oleh tanaman penaungnya. Masalah utama yang saat ini dihadapi, belum ada teknologi yang murah untuk melakukan koleksi data spasial terkait tanaman kakao di perkebunan untuk sistem monitoring perkebunan.

3.4. Studi Pustaka

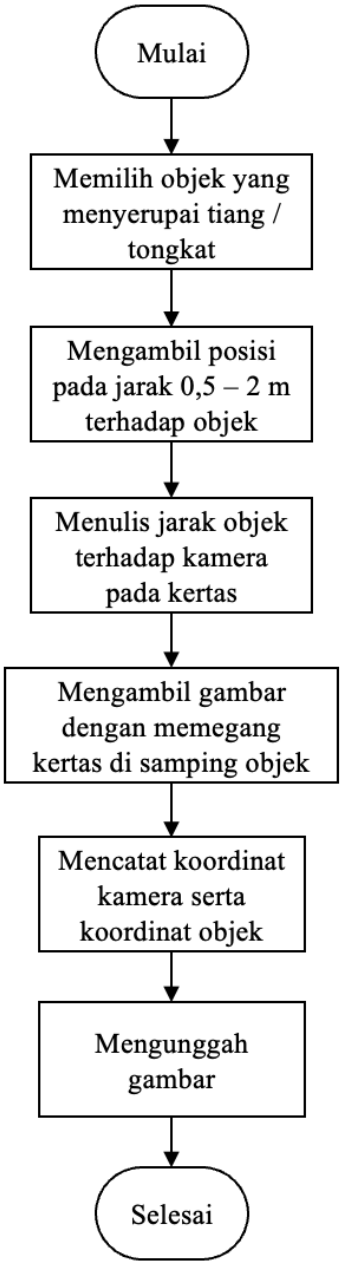
Dalam tahap ini, peneliti akan melakukan studi pustaka terkait dengan geotagging otomatis dan penghitungan otomatis buah kakao. Tujuan dari studi pustaka ini adalah untuk menemukan metode dan langkah-langkah yang dapat digunakan dalam penelitian ini. Peneliti akan mencari dan mengumpulkan informasi dari penelitian terdahulu yang telah dilakukan dan relevan dengan topik penelitian ini. Ringkasan hasil studi pustaka akan ditampilkan pada tabel 3.1 yang berisi informasi singkat mengenai penelitian terdahulu yang digunakan sebagai acuan dalam penelitian ini.

Tabel 3. 1 Studi Pustaka

No	Topik	Pengetahuan	Temuan
1	Akurasi GPS	Perangkat Telepon Genggam memiliki akurasi yang cukup baik	Berdasarkan studi yang dilakukan, perangkat telepon genggam (iPhone 6) memiliki rata-rata error pada rentang 7-13 m. Sedangkan geotagging yang akan dilakukan memerlukan akurasi pada tingkat sentimeter. (Merry, 2019)
2	Estimasi Kedalaman	Model CNN untuk melakukan Estimasi Kedalaman dapat digunakan untuk memprediksi jarak objek pada Citra	Model CNN berbasis segmentasi (dengan arsitektur U-Net) dapat digunakan untuk melakukan estimasi kedalaman pada citra. (Godard, 2019)
3	Deteksi Buah Kakao Otomatis	Model CNN YOLO dapat digunakan untuk mendeteksi Buah Kakao	Model CNN dengan arsitektur YOLO-v5 dapat melakukan deteksi fenotip polong kedelai, termasuk klasifikasi dan kuantifikasi jumlah polong berdasarkan klasifikasinya. (Fu, 2022).

3.5. Pengumpulan Data

Berikut gambar 3.2 yang menjelaskan alur pengumpulan data yang akan dilakukan pada penelitian ini.



Gambar 3.2 Alur Pengumpulan Data

Pengumpulan data dalam penelitian ini dimulai dengan pemilihan objek yang memiliki bentuk yang menyerupai tiang atau tongkat. Pemilihan objek juga mempertimbangkan lingkungan sekitar objek. Khususnya lingkungan yang menjadi

latar belakang objek. Latar belakang yang memiliki warna mirip dengan objek akan mempersulit proses segmentasinya. Sehingga perlu dicari objek yang berada pada lingkungan dengan latar belakang yang bersih. Artinya, tidak banyak objek yang mirip dengan objek tiang serta terdapat perbedaan warna yang jelas antara objek dan lingkungan sekitarnya.

Objek yang menyerupai tiang dipilih karena objek dengan bentuk tersebut mudah dikenali oleh model CNN dan mampu memberikan hasil estimasi kedalaman yang optimal. Selanjutnya, jarak antara objek dan kamera diatur pada rentang 0,5-2 meter. Rentang jarak ini dipilih karena jarak yang terlalu dekat atau jauh dapat mempengaruhi kualitas hasil estimasi kedalaman yang diperoleh.

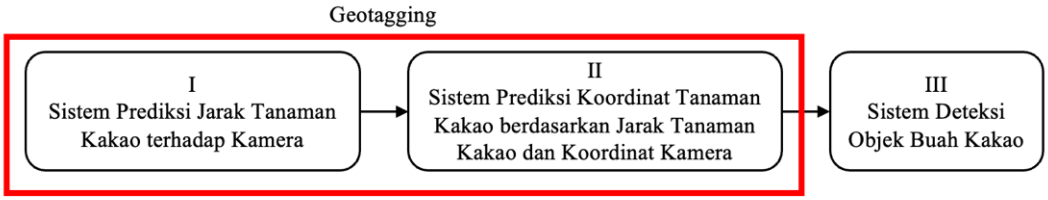
Setelah jarak antara objek dan kamera diatur, dilakukan pencatatan jarak antara objek dan kamera pada kertas kecil. Pencatatan ini dilakukan untuk memastikan bahwa jarak antara objek dan kamera telah sesuai dengan rentang yang ditentukan sebelumnya. Selanjutnya, dilakukan pengambilan gambar bersama dengan objek yang telah dipilih sebelumnya. Pada tahap ini, dilakukan pencatatan titik koordinat kamera saat pengambilan gambar dan titik koordinat objek.

Tahapan terakhir dalam pengumpulan data adalah mengunggah gambar yang telah diambil bersama dengan data koordinat kamera dan objek ke dalam sistem yang digunakan untuk analisis data lebih lanjut. Proses pengunggahan ini dilakukan agar data dapat diakses dan dianalisis dengan lebih mudah dan efektif. Dengan demikian, pengumpulan data yang dilakukan dengan metode ini dapat menghasilkan data yang akurat dan dapat digunakan untuk analisis lebih lanjut dalam berbagai aplikasi yang memerlukan estimasi kedalaman.

3.6. Desain Sistem

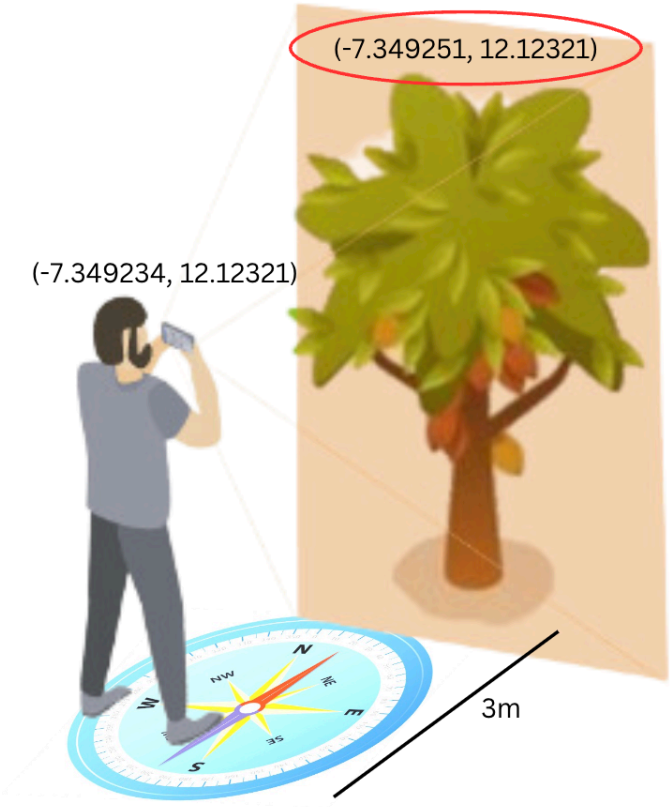
Pada tahapan ini, penulis memaparkan desain sistem yang akan dilakukan pada penelitian ini. Gambar 3.6 menjelaskan desain sistem terkait sistem prediksi jarak objek pada citra. Gambar 3.7 menjelaskan desain sistem mengenai sistem prediksi koordinat tanaman kakao pada citra. Sedangkan pada gambar 3.8

dijelaskan sistem estimasi jumlah buah kakao dengan melakukan deteksi buah kakao lalu menghitung jumlah buah yang terdeteksi.



Gambar 3.3 Desain Sistem Keseluruhan

Gambar 3.3 menjelaskan terkait Desain Sistem secara utuh. Tahap pertama yang dilakukan sistem yaitu melakukan prediksi jarak tanaman kakao dari kamera. Nilai jarak ini kemudian digunakan bersama titik koordinat kamera untuk memprediksi titik koordinat dari objek tanaman kakao. Lalu pada tahap terakhir dilakukan deteksi objek buah kakao untuk melakukan penghitungan buah kakao secara otomatis. Ilustrasi keseluruhan desain sistem dapat dilihat pada gambar 3.4 dan 3.5.

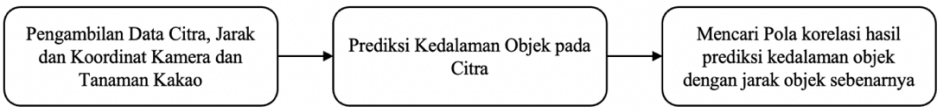


Gambar 3.4 Visualisasi Desain Sistem

Gambar 3.5 Visualisasi Alur Sistem Keseluruhan

3.6.1. Desain Sistem Prediksi Jarak Objek pada Citra

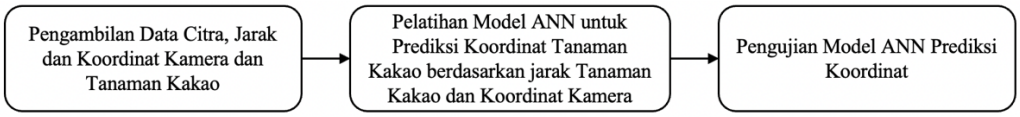
Pada tahap ini akan dijelaskan desain sistem prediksi jarak objek yang akan digunakan pada penelitian ini (dapat dilihat pada gambar 3.6). Model CNN *monocular depth estimation* digunakan pada penelitian ini, model tersebut dikembangkan oleh Godard (2019). Model CNN tersebut merupakan model yang dikembangkan menggunakan arsitektur U-Net untuk melakukan segmentasi objek kemudian memprediksi kedalaman dari setiap objek.



Gambar 3.6 Desain Sistem Metode Prediksi Jarak Objek pada Citra

3.6.2. Desain Sistem Prediksi Koordinat Tanaman Kakao pada Citra

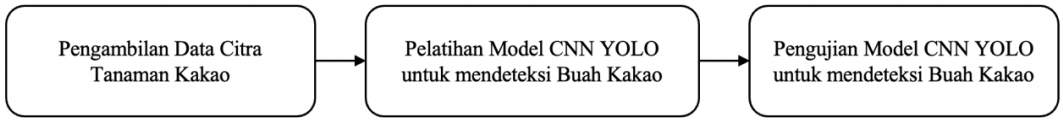
Pada tahap ini, akan dipaparkan sistem prediksi koordinat tanaman kakao pada citra (dapat dilihat pada gambar 3.7). Dalam proses ini, data citra yang telah diambil akan dimanfaatkan untuk melakukan proses ekstraksi fitur. Teknik pengolahan citra seperti segmentasi, ekstraksi tekstur, dan ekstraksi bentuk digunakan untuk mengidentifikasi dan memperoleh fitur-fitur yang relevan dari citra tersebut. Setelah fitur-fitur ini diperoleh, langkah selanjutnya adalah menerapkan algoritma prediksi yang telah dirancang. Algoritma ini dapat melibatkan pendekatan machine learning seperti pengklasifikasi atau regresi, yang memungkinkan sistem untuk memprediksi koordinat tanaman kakao dengan tingkat akurasi yang tinggi. Dengan menggunakan sistem prediksi ini, pemantauan pertumbuhan dan perkembangan tanaman kakao dapat dilakukan secara efisien dan otomatis, membantu para petani dan peneliti dalam mengoptimalkan produksi dan perawatan tanaman kakao.



Gambar 3.7 Desain Sistem Metode Prediksi Koordinat Tanaman Kakao

3.6.3. Desain Sistem Estimasi Jumlah Buah Kakao

Tahap ini merupakan paparan mengenai sistem estimasi jumlah buah kakao. Diagram alur desain sistem estimasi jumlah buah kakao terdapat pada Gambar 3.8. Untuk mengembangkan model CNN, diperlukan pelatihan menggunakan data set mengenai objek terkait. Pada penelitian ini, objek yang akan dideteksi adalah buah kakao. Sehingga tahap pertama yang diperlukan yaitu mengumpulkan dan menandai buah kakao pada citra tanaman kakao. Dataset buah kakao memiliki 2 kelas yaitu kelas buah kakao yang sudah matang dan kelas buah kakalo yang belum matang. Setelah itu, tanda / anotasi buah kakao beserta citra tanaman kakao akan digunakan pada tahap pelatihan model CNN YOLO. Setelah pelatihan selesai dilakukan, maka dihasilkan sebuah model CNN YOLO yang dapat mendeteksi buah kakao serta prediksi status kematangan buah tersebut. Buah-buah yang berhasil dideteksi oleh model CNN YOLO akan dihitung.



Gambar 3.8 Desain Sistem Metode Estimasi Jumlah Buah Kakao

3.7. Pengujian Arsitektur

Pada tahap pengujian arsitektur akan dilakukan beberapa percobaan untuk menguji akurasi dari sistem prediksi jarak, sistem prediksi koordinat serta sistem estimasi jumlah buah kakao.

3.7.1. Pengujian Sistem Prediksi Jarak Objek pada Citra

Untuk mengetahui akurasi nilai *output* dari sistem prediksi jarak objek pada citra maka pada penelitian ini digunakan parameter *Mean Absolute Error* (MAE) dan *Mean Squared Error* (MSE).

3.7.2. Pengujian Sistem Prediksi Koordinat Tanaman Kakao pada Citra

Untuk mengetahui akurasi nilai *output* dari sistem prediksi koordinat berupa longituda dan latitude tanaman kakao pada citra maka pada penelitian ini digunakan parameter *Mean Absolute Error* (MAE) dan *Mean Squared Error* (MSE).

3.7.3. Pengujian Sistem Estimasi Jumlah Buah Kakao

Untuk mengetahui akurasi hasil deteksi objek pada sistem estimasi jumlah buah kakao akan digunakan metrik *Intersection over Union* (IoU). IoU mengukur seberapa banyak area yang tumpang tindih antara bounding box atau masker prediksi dengan bounding box atau masker acuan.

BAB IV HASIL DAN PEMBAHASAN

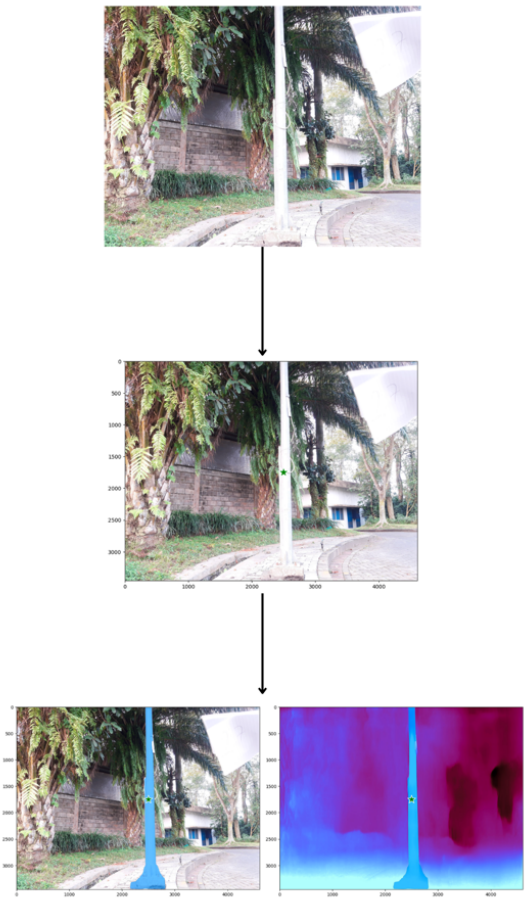
4.1. Dataset

Pada penelitian ini, dilakukan pengumpulan data untuk membangun dataset yang akan digunakan dalam proses pelatihan model *Artificial Neural Network* (ANN) untuk memprediksi jarak serta model *Convolutional Neural Network* (CNN) YOLO untuk mendeteksi buah kakao. Pengumpulan data dilakukan dengan cara mengambil sampel gambar-gambar buah kakao yang sudah matang dan yang belum matang. Selama proses pengumpulan data, gambar-gambar tersebut diolah dan diberikan anotasi. Anotasi ini mencakup informasi mengenai koordinat atau bounding box yang menandai letak buah kakao pada gambar, serta label yang menunjukkan jenis atau kelas buah kakao yang terdeteksi. Setelah pengumpulan data selesai, dataset yang terdiri dari gambar-gambar dan anotasinya akan digunakan dalam proses pelatihan model. Model ANN akan dilatih menggunakan data jarak antara kamera dan buah kakao sebagai input, serta data yang sesuai dengan jarak tersebut sebagai output yang diharapkan. Sementara itu, model CNN YOLO akan dilatih menggunakan gambar-gambar buah kakao beserta anotasinya. Proses pelatihan ini bertujuan untuk mengajarkan model untuk mendeteksi dan mengidentifikasi buah kakao dalam gambar.

4.1.1. Pembuatan Dataset Nilai Rgb dan Jarak Objek

Pembuatan dataset dimulai dengan mencari objek yang menyerupai tiang untuk menggantikan batang tanaman kakao. Setelah objek yang sesuai ditemukan, jarak antara kamera dan objek ditentukan. Setelah jarak kamera dengan objek ditentukan, citra diambil menggunakan kamera atau perangkat lainnya. Pada langkah ini, citra yang telah diambil akan diproses menggunakan model CNN *monodepth estimation*. Model ini akan memperkirakan kedalaman objek berdasarkan nilai RGB pada citra. Setelah estimasi kedalaman dilakukan, ditentukan titik koordinat piksel (x, y) pada area objek yang diestimasi

kedalamannya. Titik koordinat ini akan digunakan dalam proses segmentasi selanjutnya. Pada langkah ini, model CNN *segment anything* digunakan untuk melakukan segmentasi pada citra hasil estimasi kedalaman. Model CNN *segment anything* melakukan proses segmentasi berdasarkan titik koordinat yang telah ditentukan. Titik tersebut menjadi dasar penunjuk objek yang akan disegmentasi areanya. Segmentasi akan mengidentifikasi bagian objek yang menyerupai tiang dan memisahkannya dari latar belakang. Setelah proses segmentasi, nilai RGB pada bagian objek yang diidentifikasi akan diperoleh. Nilai RGB ini merupakan warna piksel pada citra hasil estimasi kedalaman. Setelah mendapatkan kumpulan nilai RGB, nilai median dari kumpulan tersebut diambil. Nilai median memberikan representasi warna tengah dari kumpulan tersebut, yang akan digunakan sebagai representasi nilai RGB untuk jarak objek yang diestimasi. Setelah semua langkah di atas dilakukan untuk setiap objek yang dipilih, data yang terkumpul ditambahkan ke dataset. Setiap data terdiri dari pasangan nilai RGB (nilai median) dan jarak kamera terhadap objek yang sesuai.



Gambar 4.1 Alur Pembuatan Dataset

4.1.2. Profil Dataset Nilai RGB dan Jarak Objek

Pada penelitian ini, terdapat dataset yang berisi pasangan nilai Red,Green,Blue (RGB) terhadap jarak objek dalam satuan meter. Namun, perlu dicatat bahwa nilai RGB yang ada dalam dataset tersebut bukanlah nilai RGB dari citra asli, melainkan nilai RGB yang berasal dari proses estimasi kedalaman menggunakan model CNN *monodepth estimation*. Nilai RGB akan digunakan sebagai input untuk memperkirakan jarak objek. Proses estimasi kedalaman menggunakan model CNN *monodepth estimation*. Dataset ini terdiri dari 112 data, dimana setiap data terdiri dari nilai RGB dan jarak objek yang sesuai.

Tabel 4.1 Dataset RGB dan Jarak Objek

id	r	g	b	distance
0	251.0	138.0	99.0	2.90
1	141.0	41.0	128.0	4.22
2	220.0	72.0	107.0	1.00
3	80.0	18.0	123.0	3.00
4	225.0	76.0	103.0	3.75
...	...	...	...	...
110	222.0	75.0	124.0	2.70
111	252.0	158.0	112.0	1.20
112	156.0	46.0	126.0	2.40

4.1.3. Dataset Buah Kakao

Dataset Buah Kakao dilakukan dengan melakukan pengambilan citra di perkebunan kakao yang terletak di Puslitkoka, Jember. Dilakukan pengambilan citra pada tanaman kakao yang sedang berbuah baik yang buahnya sudah matang maupun yang belum matang. Pengambilan citra juga divariasikan jaraknya yaitu pada rentang jarak sebesar 0,8 hingga 2,8 m sesuai dengan variasi yang terdapat pada dataset nilai rgb dan jarak objek (Bab 4.1.2). Dataset Buah Kakao terdiri dari

total 67 gambar buah kakao dengaan 446 anotasi. Untuk melakukan proses *training* digunakan 46 gambar (sekitar 69% dari total dataset). Selanjutnya, untuk memvalidasi performa model, 13 gambar (sekitar 19% dari total dataset) akan digunakan sebagai *validation set*. Terakhir, 8 gambar (sekitar 12% dari total dataset) akan dijadikan sebagai *test set* untuk menguji akurasi model yang telah dilatih. Dalam setiap gambar, terdapat rata-rata 6 anotasi (label) untuk buah kakao. Dataset ini memiliki dua kelas, yaitu buah kakao matang dan belum matang. Dengan pembagian ini, diharapkan model dapat mempelajari dan mengklasifikasikan gambar-gambar buah kakao dengan akurasi yang tinggi.

Gambar 4.2 Semua Dataset

Secara keseluruhan, jumlah anotasi pada kedua kelas, yaitu kelas cocoa matang *(ripe cocoa)* dan kelas cocoa belum matang *(unripe cocoa)*, tergolong cukup seimbang. Terdapat 227 anotasi pada kelas *cocoa* matang dan 217 anotasi pada kelas *cocoa* belum matang. Perbedaan jumlah anotasi antara kedua kelas tersebut tidak terlalu signifikan.

Gambar 4.3 Pembagian Dataset untuk Train

Data pelatihan ini memiliki jumlah sampel yang cukup representatif untuk kedua kelas, yaitu 171 sampel pada kelas *cocoa* matang *(ripe cocoa)* dan 130 sampel pada kelas *cocoa* belum matang *(unripe cocoa)*. Dengan jumlah yang seimbang antara kedua kelas, model pembelajaran mesin dapat mempelajari pola dan informasi yang relevan dari masing-masing kelas. Dengan menggunakan data pelatihan yang representatif ini, model yang dihasilkan memiliki kemampuan yang lebih baik dalam mengklasifikasikan *cocoa* berdasarkan tingkat kematangannya.

Gambar 4.4 Pembagian Dataset untuk Valid

Dataset validasi yang mencakup kedua kelas, yaitu *cocoa* matang *(ripe cocoa)* dengan 24 sampel dan *cocoa* belum matang *(unripe cocoa)* dengan 47 sampel, memungkinkan pengujian kehandalan model dalam mengklasifikasikan tingkat kematangan *cocoa*. Evaluasi menggunakan dataset validasi ini memberikan gambaran yang lebih akurat tentang kemampuan model dalam dunia nyata.

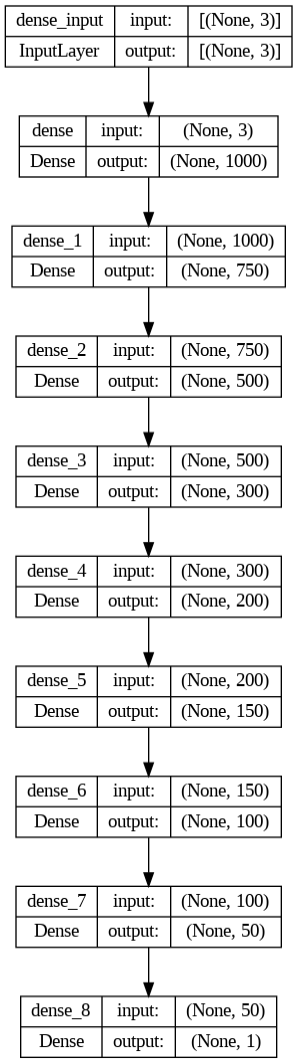
Gambar 4.5 Pembagian Dataset untuk Test

Pada dataset uji *(test set)*, terdapat 32 sampel anotasi pada kelas *cocoa* matang *(ripe cocoa)* dan 40 sampel anotasi pada kelas *cocoa* belum matang *(unripe cocoa)*. Dataset uji ini digunakan untuk menguji performa model yang telah dilatih pada data pelatihan dan divalidasi pada data validasi. Dengan menggunakan dataset uji yang mencakup kedua kelas, model dapat dievaluasi lebih lanjut dalam kemampuannya mengklasifikasikan *cocoa* berdasarkan tingkat kematangannya.

4.2. Eksperimen Model ANN Prediksi Jarak Objek pada Citra

Pada penelitian ini, dilakukan beberapa percobaan pelatihan *Artificial Neural Network* (ANN) untuk mengetahui korelasi antara variabel RGB piksel dengan jarak aktualnya. Percobaan ini menggunakan model CNN *Monodepth Estimation* yang menghasilkan citra RGB dengan nilai yang sangat bervariasi. Percobaan dilakukan dengan menggunakan dua jenis optimizer, yaitu Adamax dan SGD. Optimizer digunakan untuk mengatur proses pembelajaran ANN dengan menyesuaikan bobot dan bias agar mencapai hasil yang optimal. Model akan dilatih dengan sebanyak 112 data.

Selain itu, juga dilakukan variasi pada 3 jumlah epoch (1000, 3000, 5000) dan 3 variasi *batch size* (1, 7, 15). Epoch merupakan iterasi yang dilakukan saat melatih model, sedangkan *batch size* menentukan jumlah sampel yang digunakan dalam satu iterasi. Dengan menggabungkan variasi optimizer, epoch, dan *batch size*, percobaan ini bertujuan untuk mencari kombinasi yang paling baik dalam menghasilkan model ANN yang dapat memprediksi jarak aktual berdasarkan nilai RGB piksel. Sebagai tambahan informasi, pada pengembangan model ANN ini digunakan *Mean Absolute Error* (MAE) sebagai *loss metric* untuk mengetahui akurasi model yang telah dilatih. Sebagai acuan, pada penelitian ini telah disepakati untuk mengembangkan model ANN yang memiliki nilai MAE dibawah 0.3. Adapun arsitektur ANN yang akan digunakan dapat dilihat pada gambar 4.6.

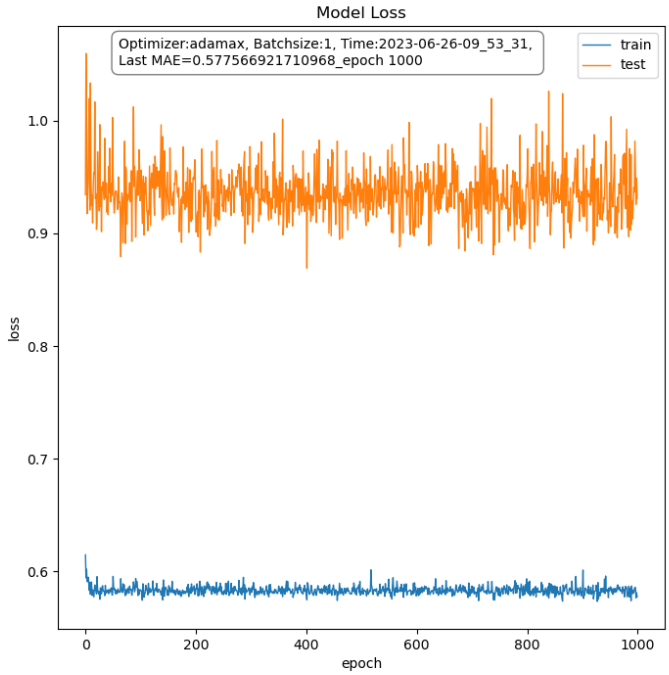


Gambar 4.6 Arsitektur Model ANN

Dari kombinasi epoch, *batch size* serta *optimizer* dihasilkan 18 model ANN berikut.

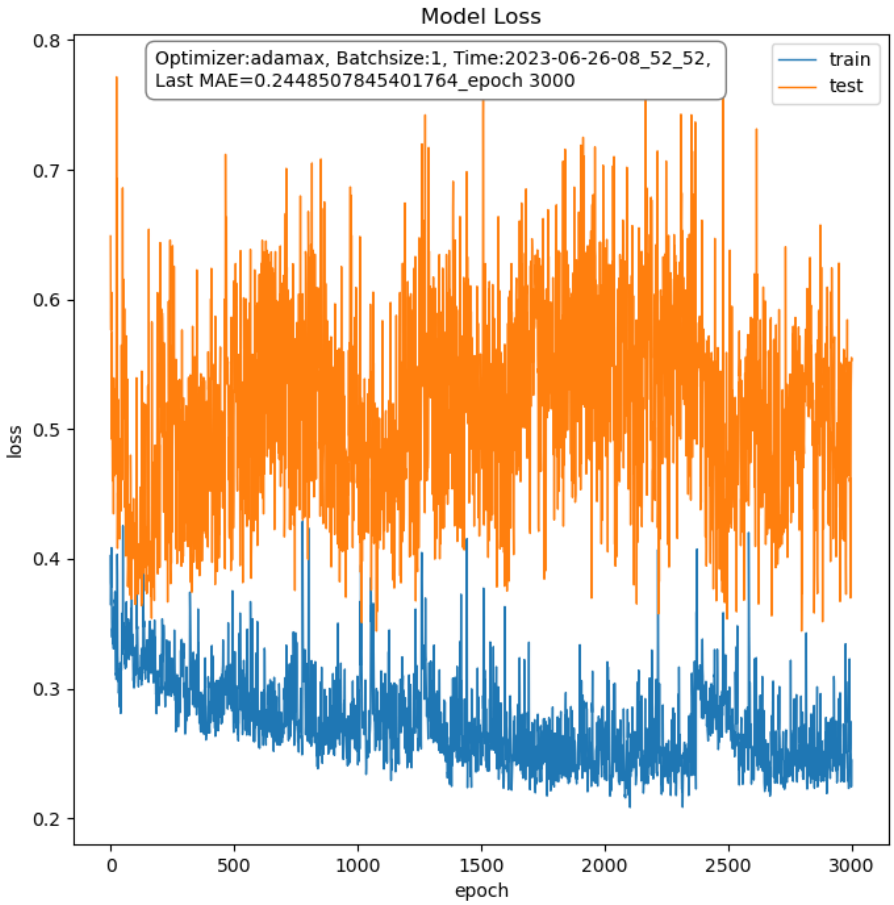
4.2.1. Model ANN dengan Optimizer Adamax dan *Batch Size* 1

Pada model ANN ini, digunakan optimizer Adamax dengan parameter *batch size* sebesar 1. Dilakukan training sampai dengan epoch ke 1000. Setelah melalui pelatihan awal yang berlangsung hingga mencapai 1000 epoch, ditemukan bahwa nilai *Mean Absolute Error* (MAE) model sebesar 0.577566921710968. Data histori pelatihan model dapat dilihat pada gambar 4.7.



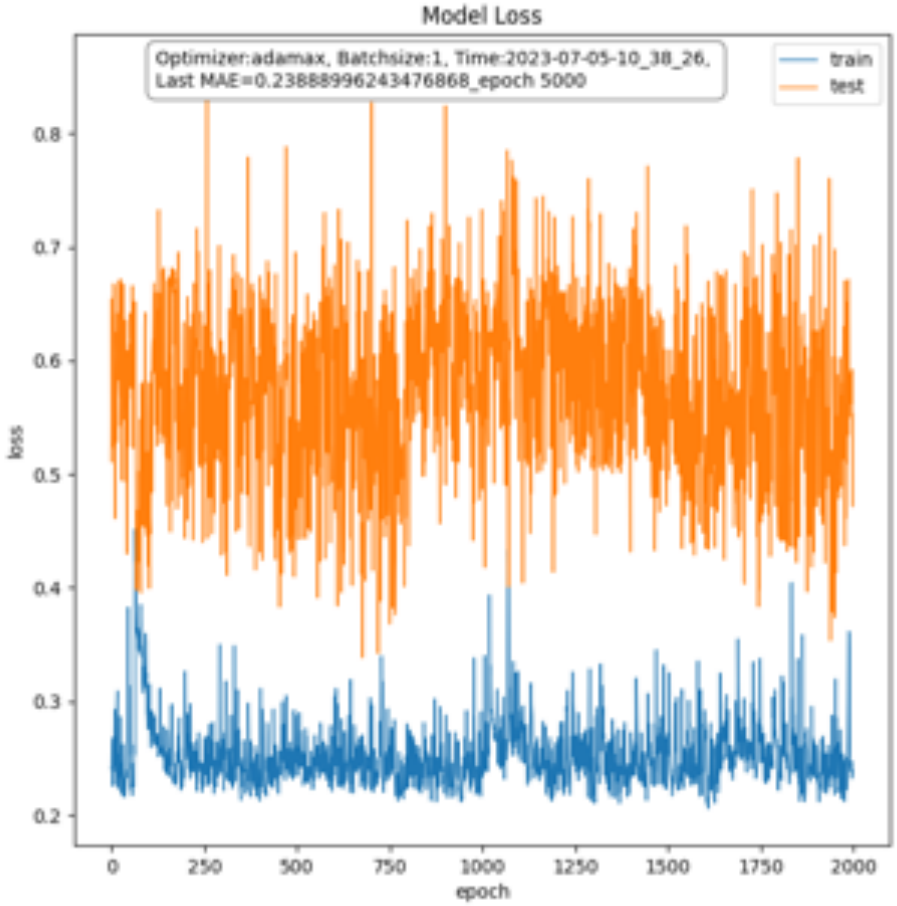
Gambar 4.7 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 1 pada epoch 1000

Meskipun demikian, belum terlihat perbaikan yang signifikan pada nilai loss baik pada data latih maupun data uji. Nilai loss *test* tidak menurun secara signifikan dan terus berubah pada kisaran 1 hingga 0,9. Sedangkan nilai loss train mengalami penurunan sedikit dari 0,6 menjadi 0,577. Meskipun penurunannya tidak signifikan, terdapat perbaikan nilai yang terjadi. Namun nilai MAE 0,577 masih terlalu besar dari acuan awal. Proses training dilanjutkan pada epoch 1000 hingga 3000. Pada gambar 4.8 akan disajikan hasil perbaikan nilai loss train dan *test* pada epoch 3000.



Gambar 4.8 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 1 pada epoch 3000

Setelah melatih model ini hingga epoch 3000, terlihat indikasi bahwa model mengalami overfitting karena terdapat perbedaan yang signifikan antara nilai loss pada data uji *(test)* dan data latih *(train)*. Namun, meskipun demikian, model ini berhasil mencapai *Mean Absolute Error* (MAE) yang cukup kecil dan sesuai dengan target yang diinginkan, yaitu 0,24485. Selanjutnya pelatihan model dilanjutkan pada epoch 3000 hingga 5000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.9.

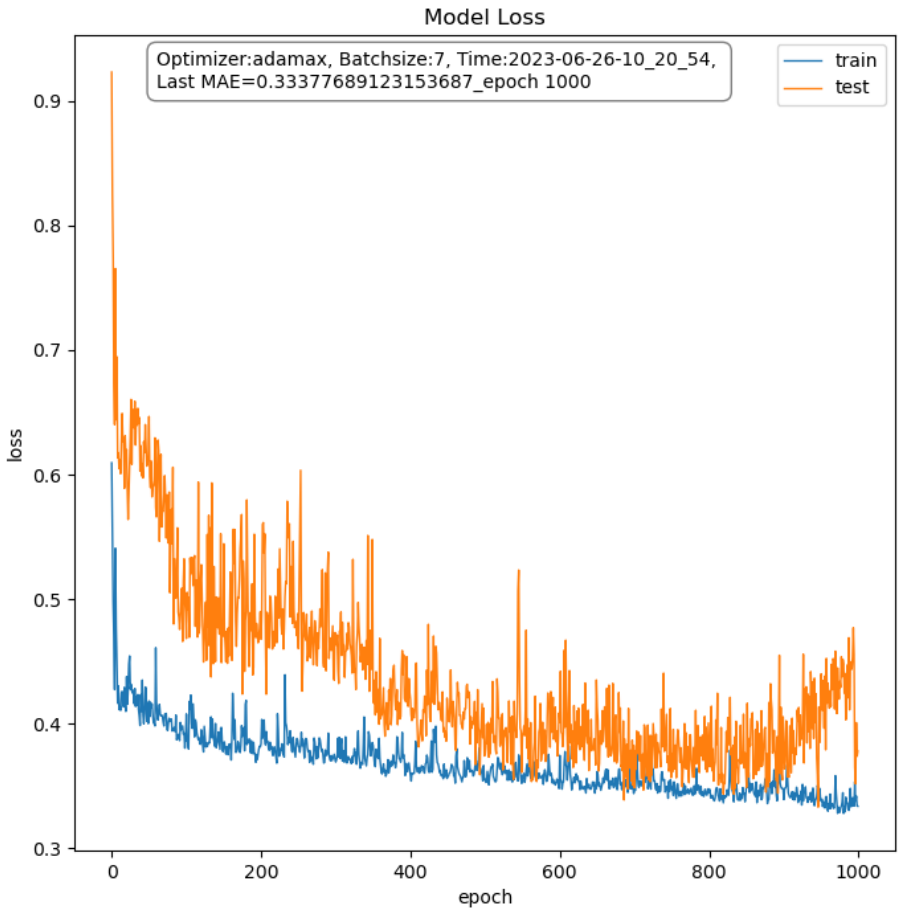


Gambar 4.9 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 1 pada epoch 5000

Setelah dilakukan pelatihan ulang hingga epoch 5000, model ini terindikasi mengalami *overfitting* karena terdapat perbedaan yang signifikan antara nilai loss pada data *test* dan data train. Tidak terlihat adanya perbaikan yang signifikan pada nilai loss juga. Nilai loss pada data *test* bervariasi antara 0,4 hingga 0,8, sedangkan nilai loss pada data train berkisar antara 0,2 hingga 0,4. Meskipun demikian, model ini berhasil mencapai *Mean Absolute Error* (MAE) yang cukup kecil dan sesuai dengan target yang diinginkan, yaitu sebesar 0,2388.

4.2.2. Model ANN dengan Optimizer Adamax dan *Batch Size* 7

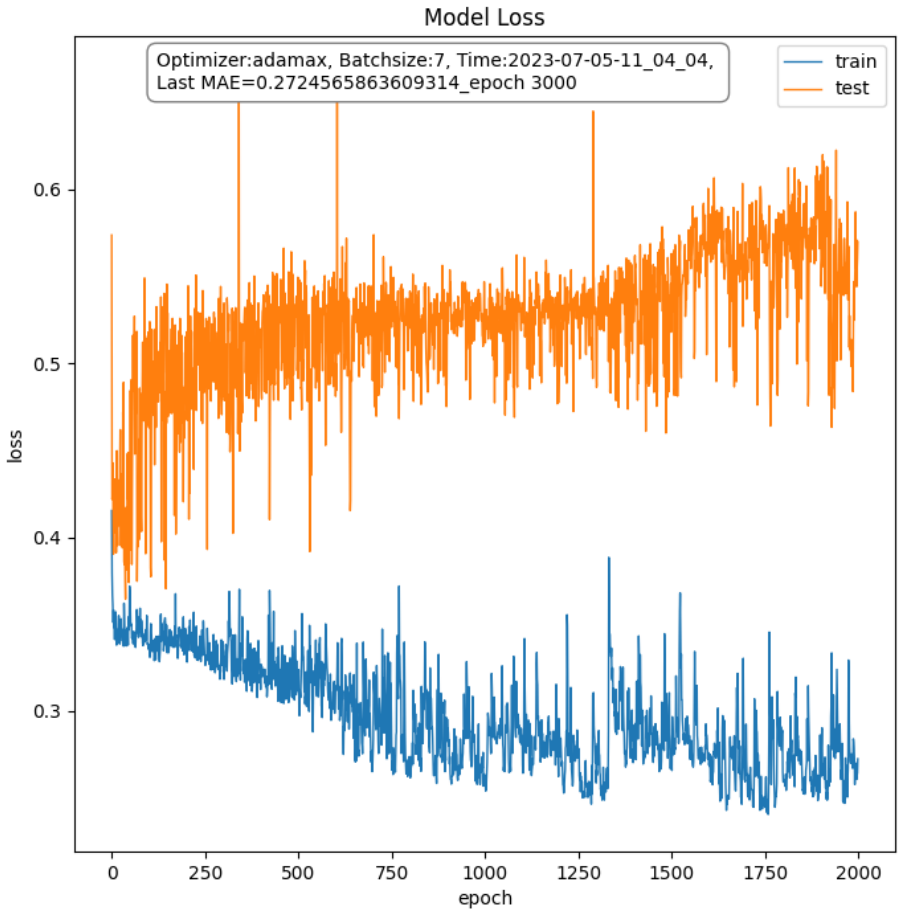
Eksperimen dilanjutkan dengan melakukan pelatihan model menggunakan optimizer Adamax dan menggunakan *batch size* sebesar 7. Pelatihan model dilakukan hingga epoch 1000. Hasil dari pelatihan tersebut dapat dilihat pada gambar 4.10.



Gambar 4.10 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 7 pada epoch 1000

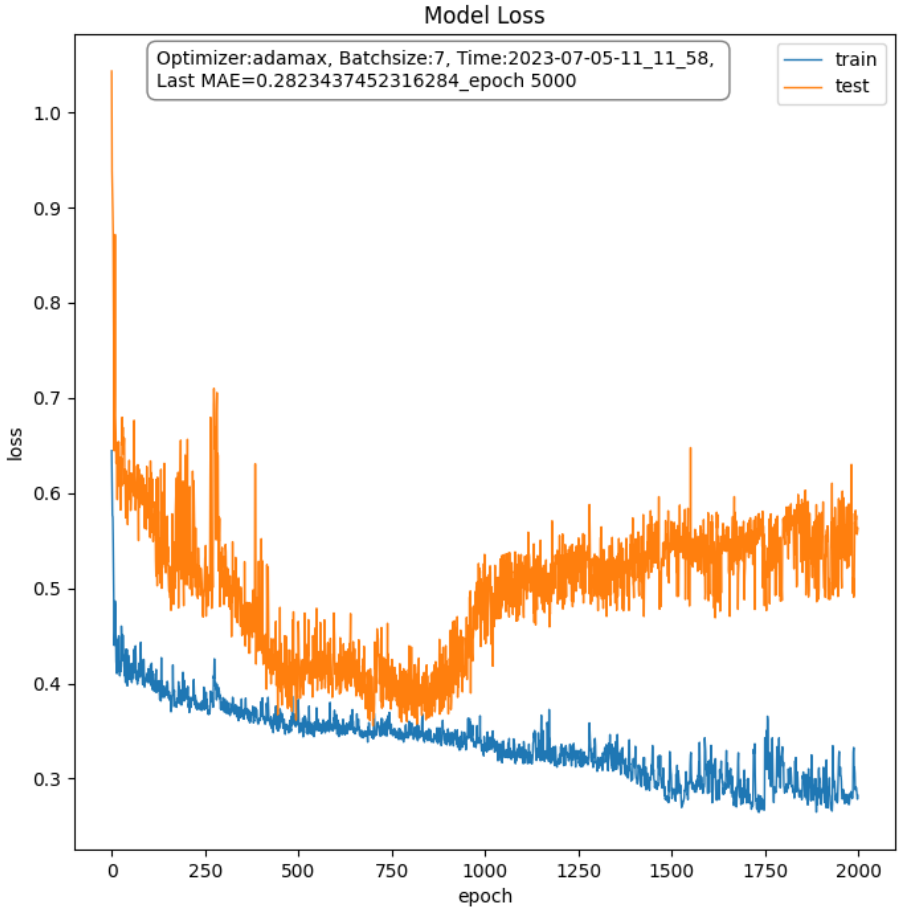
Pada grafik terlihat terjadi penurunan nilai loss yang signifikan pada proses train dan *test*. Hal ini mengindikasikan adanya perbaikan akurasi model. Nilai loss *test* awalnya berkisar pada 0,7, namun mengalami penurunan yang signifikan hingga mencapai kisaran nilai 0,35 pada epoch 800. Sementara itu, nilai loss train juga mengalami penurunan yang signifikan, dimulai dari kisaran 0,4 dan mencapai kisaran 0,35 pada epoch 1000. Meskipun nilai tersebut masih lebih tinggi dibandingkan dengan acuan yang ingin dicapai, pada proses pelatihan ini model

berhasil memperbaiki loss dengan baik. Pelatihan pada model ini dilanjutkan hingga epoch 3000. Hasil dari pelatihan tersebut akan disajikan pada gambar 4.11.



Gambar 4.11 Grafik Loss Model ANN dengan optimizer adamax dan batch size 7 pada epoch 3000

Pada grafik terlihat terjadi perbedaan antara tren grafik *train* dan *test.* Grafik *test* cenderung meningkat sedangkan grafik train cenderung menurun. Nilai *test* naik dari kisaran 0,4 hingga 0,55, sementara nilai *train* menurun dari kisaran 0,35 hingga 0,2. Hal ini merupakan indikasi yang kuat terjadinya *overfitting*. Namun, model ini berhasil mencatatkan nilai MAE yang cukup baik, yaitu sebesar 0,27245. Nilai MAE tersebut lebih kecil dibandingkan dengan nilai MAE acuan yang ingin dicapai. Pelatihan kembali dilanjutkan hingga epoch 5000. Hasil pelatihan tersebut akan disajikan pada gambar 4.12.

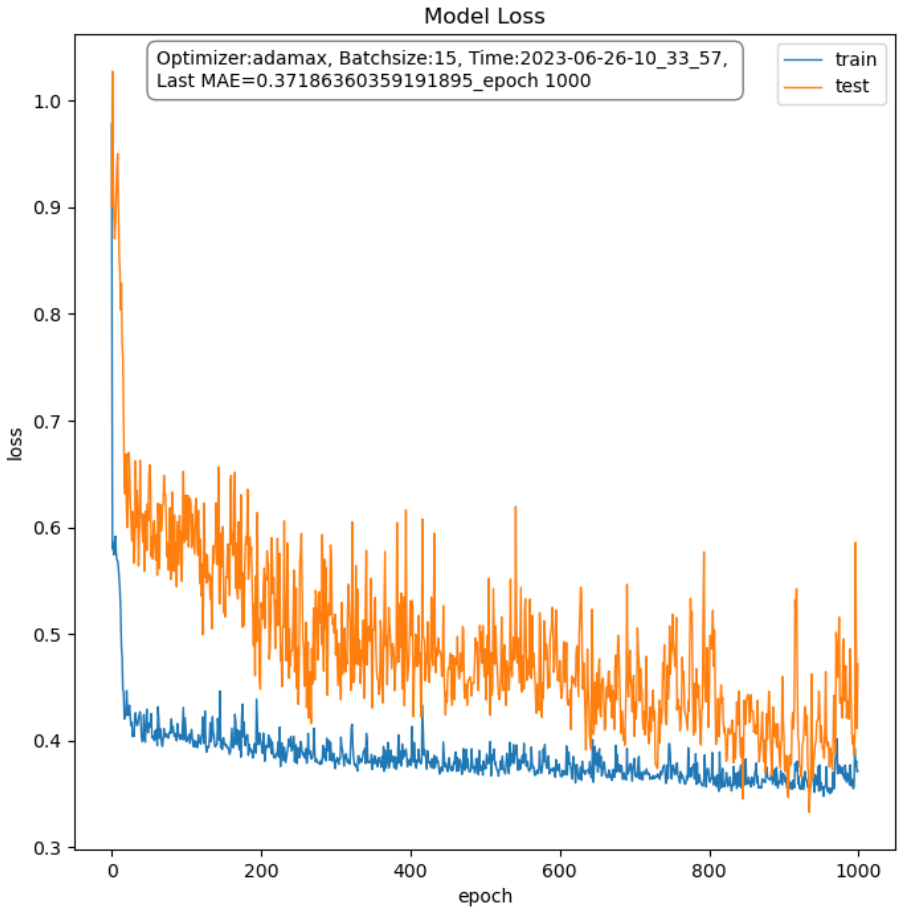


Gambar 4.12 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 7 pada epoch 5000

Dapat dilihat pada grafik di atas, terjadi perbaikan model yang signifikan antara epoch 3000 hingga 3800. Nilai *test* dan train secara bersamaan mengalami penurunan. Namun, setelah epoch 3800, nilai train terus menurun sementara nilai *test* kembali meningkat dan berkisar pada 0,5. Model ini berhasil mencapai MAE yang sesuai dengan acuan awal, yaitu sebesar 0,28234.

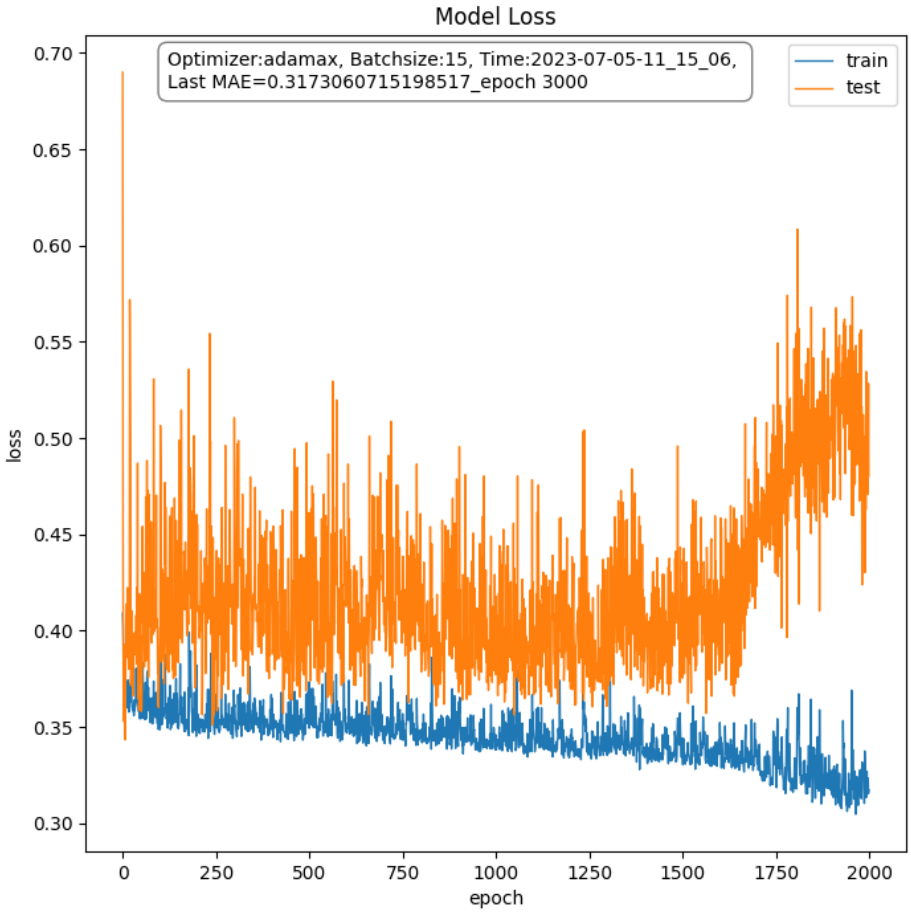
4.2.3. Model ANN dengan Optimizer Adamax dan *Batch Size* 15

Pada model *Artificial Neural Network* (ANN) ini, digunakan optimizer Adamax dengan parameter *batch size* sebesar 15. Dilakukan pelatihan model hingga mencapai epoch ke-1000. Dalam proses tersebut, dilakukan optimisasi menggunakan optimizer Adamax. Pelatihan dimulai hingga epoch 1000, hasil pelatihan tersebut akan disajikan pada gambar 4.13.



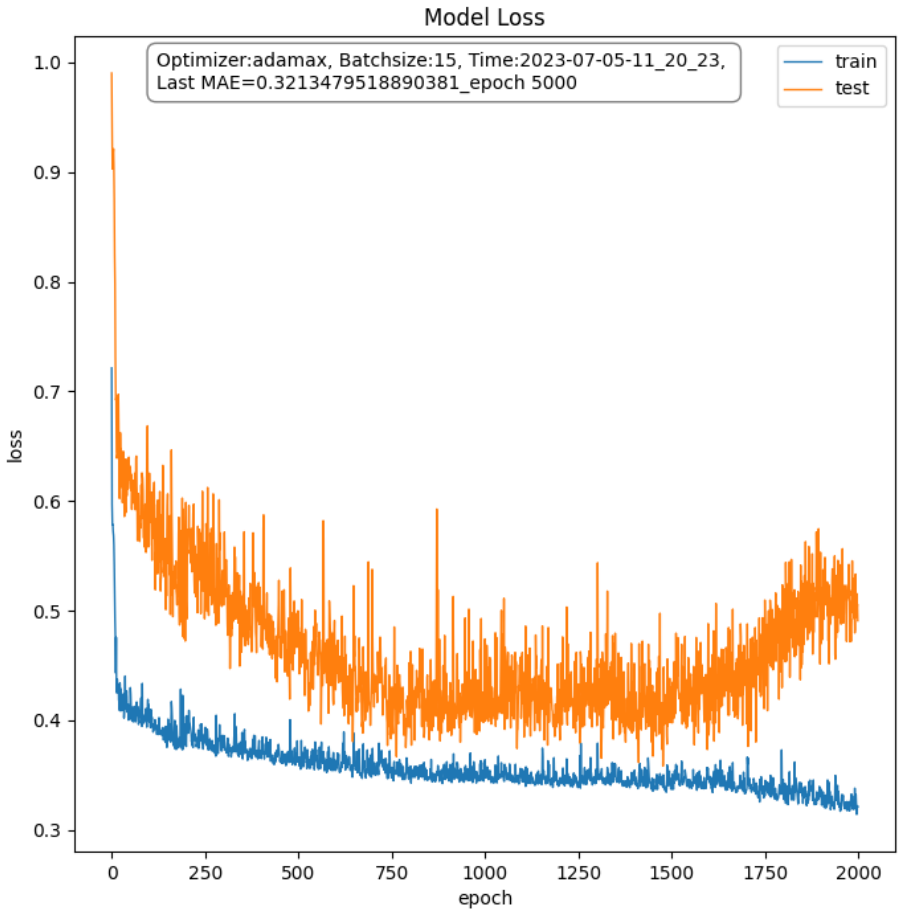
Gambar 4.13 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 15 pada epoch 1000

Pelatihan awal ini berhasil memperbaiki model dengan signifikan, terlihat dari penurunan nilai loss baik pada data train maupun data *test*. Pada epoch 1000, model ini mencapai nilai MAE sebesar 0,37186. Meskipun nilai MAE tersebut masih lebih besar dari acuan awal yang ditetapkan, tidak terlihat indikasi overfitting maupun underfitting pada model tersebut. Hal ini menunjukkan bahwa model tersebut dapat secara baik menyesuaikan diri dengan data pelatihan tanpa kehilangan kemampuan umum untuk memprediksi data baru. Pelatihan model ini dilanjutkan hingga epoch 3000. Hasil pelatihan tersebut akan disajikan pada gambar 4.14.



Gambar 4.14 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 15 pada epoch 3000

Pada pelatihan ini, terdapat indikasi bahwa model mengalami overfitting. Terlihat adanya penyimpangan pada sekitar epoch 1600, di mana nilai *test* cenderung meningkat sedangkan nilai train cenderung menurun. Model ini mencatatkan nilai MAE sebesar 0,317, yang masih lebih besar dari acuan awal yang ditetapkan. Selanjutnya, model akan dilatih kembali hingga epoch 5000, dan hasil pelatihan tersebut akan ditampilkan pada gambar 4.15.



Gambar 4.15 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 15 pada epoch 5000

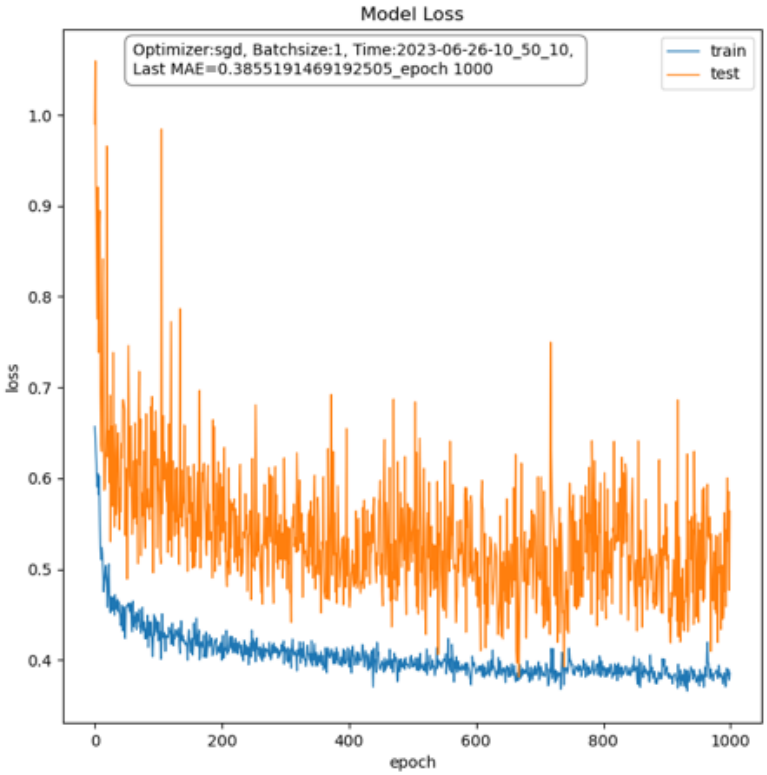
Pada pelatihan ini, terjadi penurunan yang cukup signifikan pada epoch 3000 hingga 3750. Nilai *test* dan train secara bersamaan mengalami perbaikan nilai. Namun, pada epoch 3750 hingga 4500, tidak terjadi perbaikan nilai yang signifikan. Kemudian, pada epoch 4500 hingga 5000, terdapat indikasi overfitting di mana nilai *test* cenderung meningkat sedangkan nilai train cenderung menurun.

Berdasarkan eksperimen yang telah dilakukan, optimizer Adamax berhasil menghasilkan perbaikan nilai yang signifikan pada epoch 0 hingga 1000. Pada beberapa kasus lainnya, seperti pada epoch 3000 hingga 4000, juga terjadi perbaikan nilai yang signifikan. Namun, pada epoch 1000 hingga 3000 dan 4000 hingga 5000, sering terjadi overfitting pada model. Keterbatasan jumlah data juga menjadi faktor yang mempersulit perbaikan model pada epoch di atas 1000. Oleh karena itu, melalui eksperimen berbagai kombinasi pelatihan model, akan dicari

model yang memiliki nilai MAE terendah. Terdapat 2 model dengan nilai MAE terendah yakni model yang dilatih menggunakan optimizer adamax dengan *batch size* 1 pada epoch 3000 dan 5000. Model pada epoch 3000 memiliki nilai MAE sebesar 0.2448507845401764 sedangkan model pada epoch 5000 memiliki nilai MAE sebesar 0.23888996243476868. Namun kedua model tersebut terindikasi mengalami overfitting. Adapun model lainnya yang memiliki nilai loss mae terendah namun tidak mengalami indikasi overfitting yakni model dengan optimizer adamax dan batchsize sebesar 7 pada epoch 1000. Model tersebut memiliki nilai mae sebesar 0.333776.

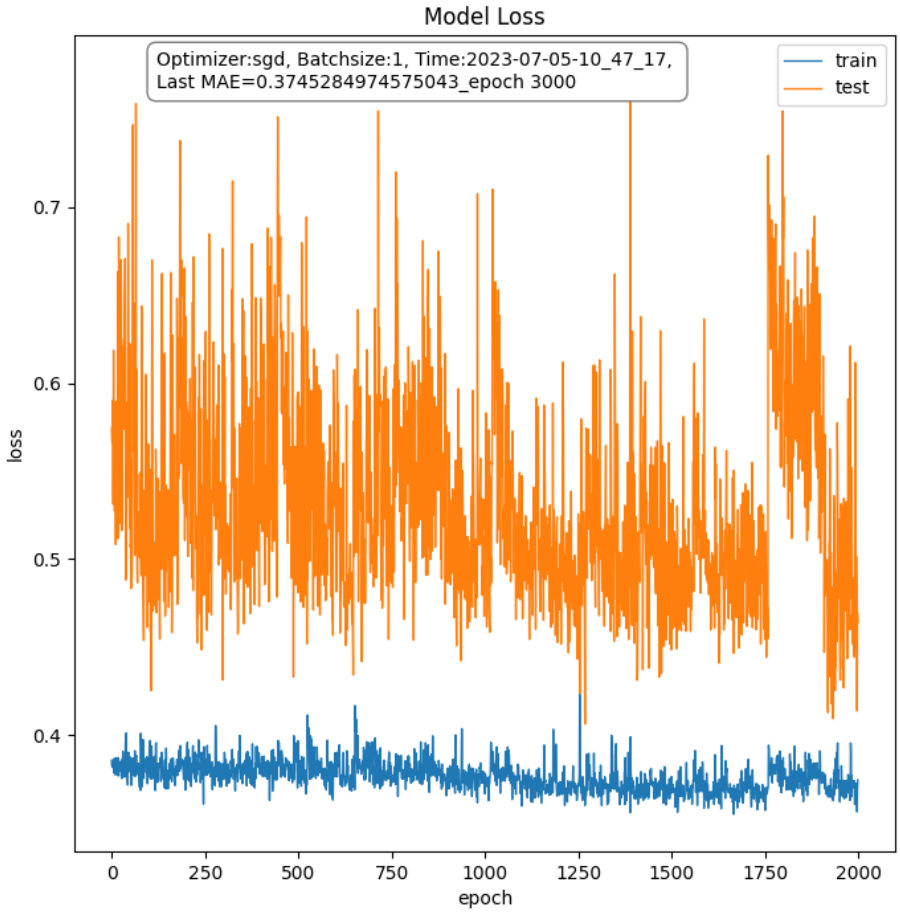
4.2.4. Model ANN dengan Optimizer SGD dan *Batch Size* 1

Pada tahap ini, model akan dilatih menggunakan optimizer SGD dengan *batch size* sebesar 1. Perkembangan model pada epoch ke-1000, ke-3000, dan ke-5000 akan dievaluasi. Nilai MAE akan menjadi parameter apakah model sudah memiliki performa yang cukup baik atau belum. Selain itu, akan dilihat pula apakah ada indikasi terjadinya overfitting atau underfitting pada model.



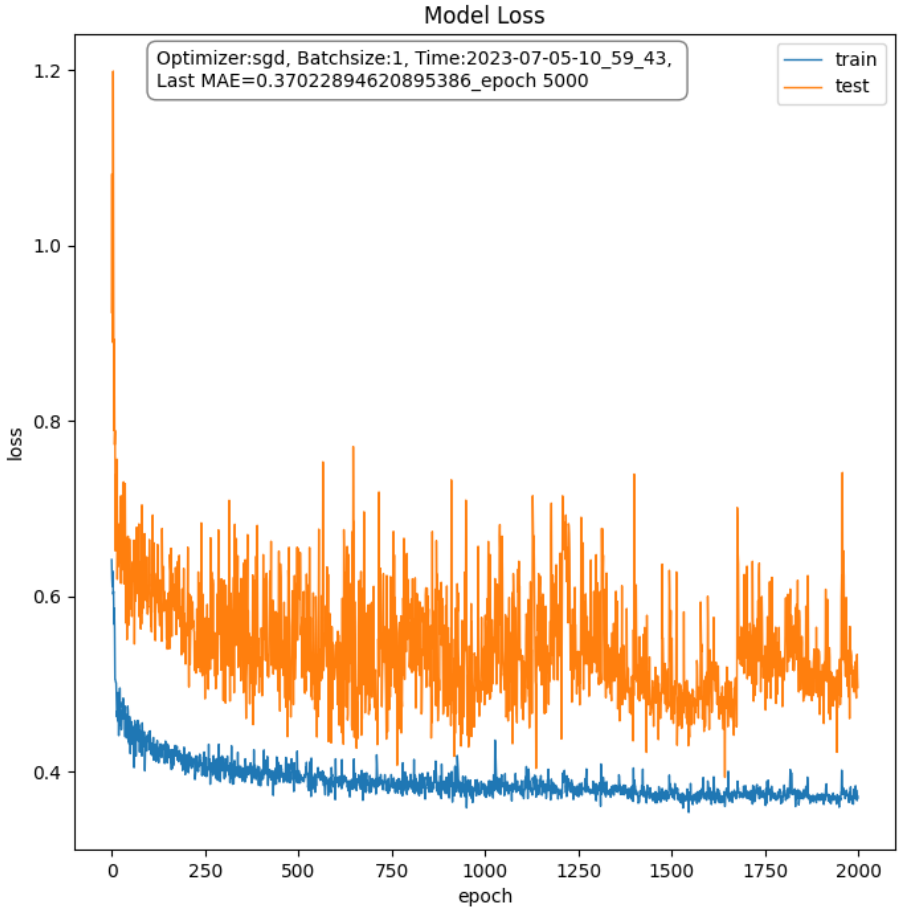
Gambar 4.16 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 1 pada epoch 1000

Pelatihan dilakukan hingga mencapai epoch 1000. Terlihat pada Gambar 4.16, terjadi penurunan signifikan pada nilai train dan *test*. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.385, yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Pelatihan model kembali dilanjutkan hinnga epoch 3000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.17.



Gambar 4.17 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 1 pada epoch 3000

Seperti yang terlihat pada Gambar 4.16, tidak terjadi perbaikan yang signifikan pada model. Nilai loss MAE pada train maupun *test* tidak mengalami perubahan yang signifikan. Nilai loss MAE pada *test* berfluktuasi antara 0.45 hingga 0.7, sementara nilai loss MAE pada train hanya berkisar antara 0.3 hingga 0.4. Model mencatatkan nilai MAE sebesar 0.37452, dimana nilai tersebut yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Pelatihan model kembali dilanjutkan hinnga epoch 5000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.18.



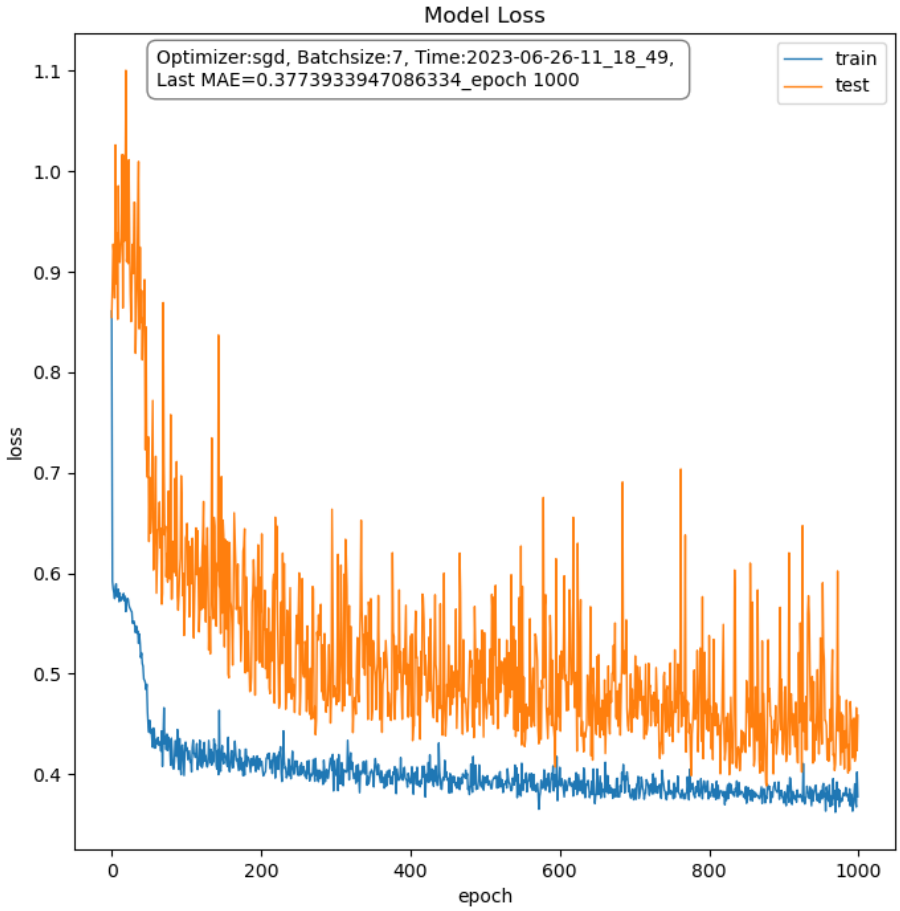
Gambar 4.18 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 1 pada epoch 5000

Seperti yang terlihat pada Gambar 4.17, tidak terjadi perbaikan yang signifikan pada model. Namun nilai loss pada train mengalami penurunan sedikit demi sedikit. Nilai loss MAE pada *test* berkisar antara 0.45 hingga 0.7, sementara nilai loss MAE pada train hanya berkisar pada nilai 0.4. Model mencatatkan nilai MAE sebesar 0.3702, dimana nilai tersebut yang masih sedikit jauh dari acuan awal yang telah ditetapkan.

4.2.5. Model ANN dengan Optimizer SGD dan *Batch Size* 7

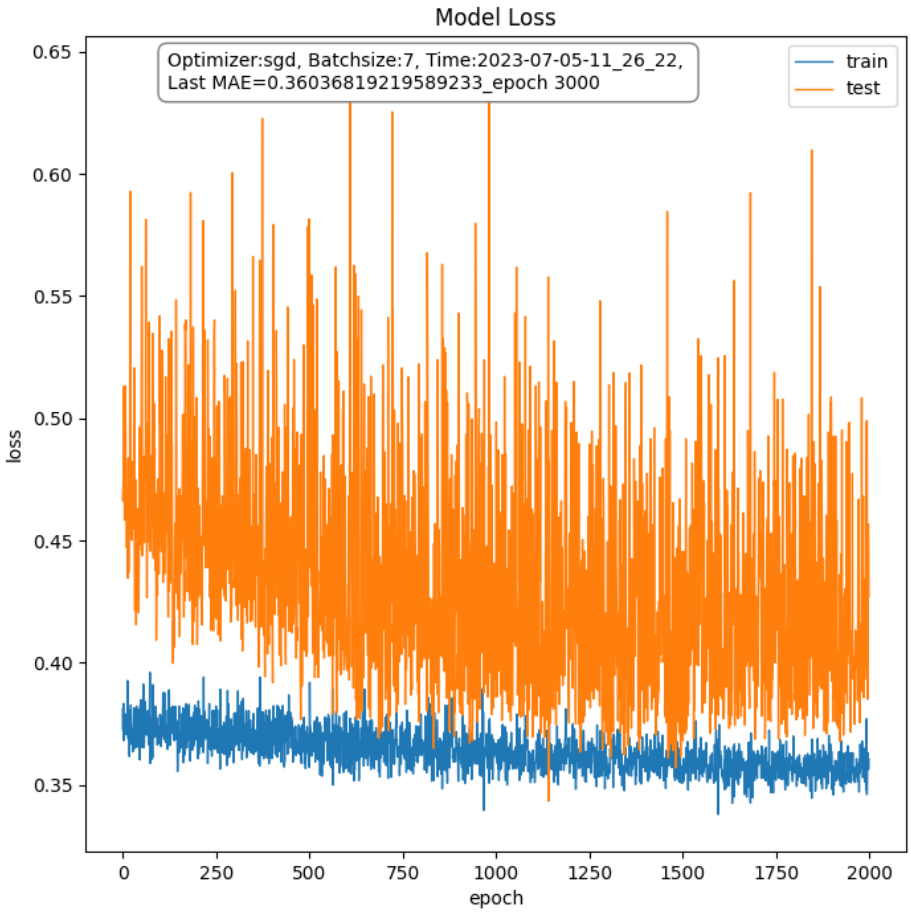
Pada tahap ini, model akan dilatih menggunakan optimizer SGD dengan *batch size* sebesar 7. Perkembangan model pada epoch ke-1000, ke-3000, dan ke-5000 akan dievaluasi. Nilai MAE akan menjadi parameter apakah model sudah memiliki

performa yang cukup baik atau belum. Selain itu, akan dilihat pula apakah ada indikasi terjadinya overfitting atau underfitting pada model.



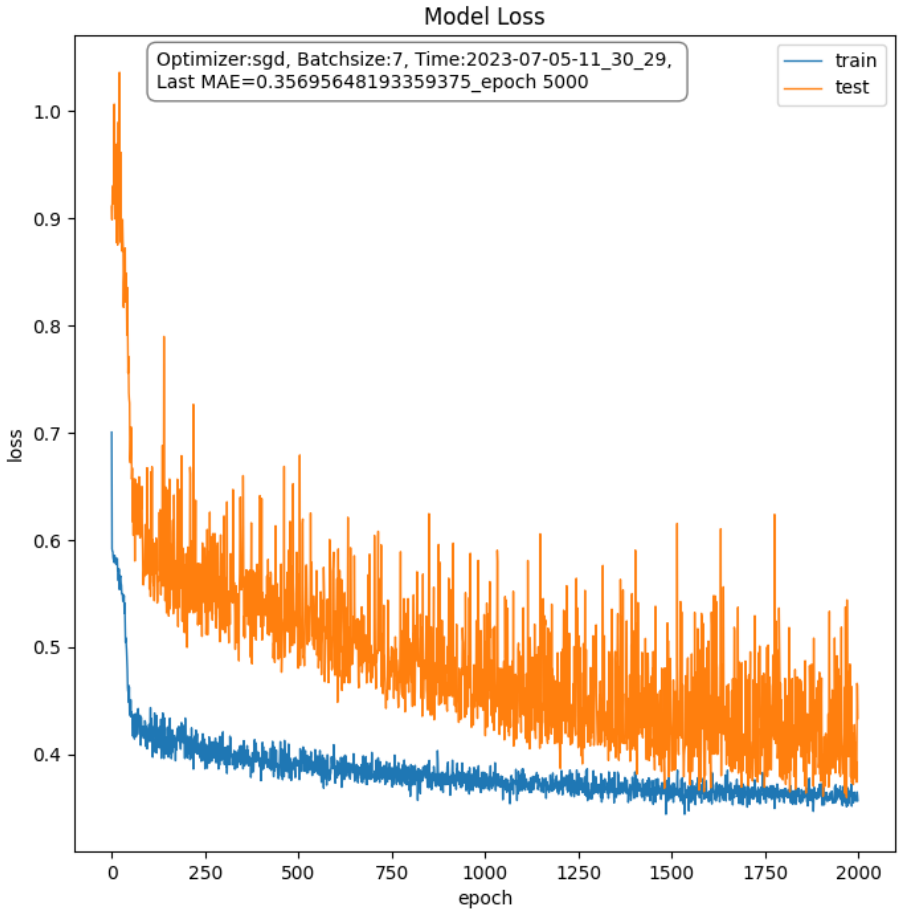
Gambar 4.19 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 7 pada epoch 1000

Pelatihan dilakukan hingga mencapai epoch 1000. Terlihat pada Gambar 4.19, terjadi penurunan signifikan pada nilai train dan *test*. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.3773933, yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Pelatihan model kembali dilanjutkan hinnga epoch 3000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.20.



Gambar 4.20 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 7 pada epoch 3000

Seperti yang terlihat pada Gambar 4.20. tidak terjadi perbaikan yang signifikan pada model. Nilai loss MAE pada train maupun *test* tidak mengalami perubahan yang signifikan. Nilai loss MAE pada *test* berfluktuasi antara 0.4 hingga 0.6, sementara nilai loss MAE pada train hanya berkisar antara 0.35 hingga 0.4. Model mencatatkan nilai MAE sebesar 0.360368, dimana nilai tersebut yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Tidak adanya perbaikan yang signifikan merupakan indikasi terjadinya overfitting pada model ini. Pelatihan model kembali dilanjutkan hinnga epoch 5000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.21.

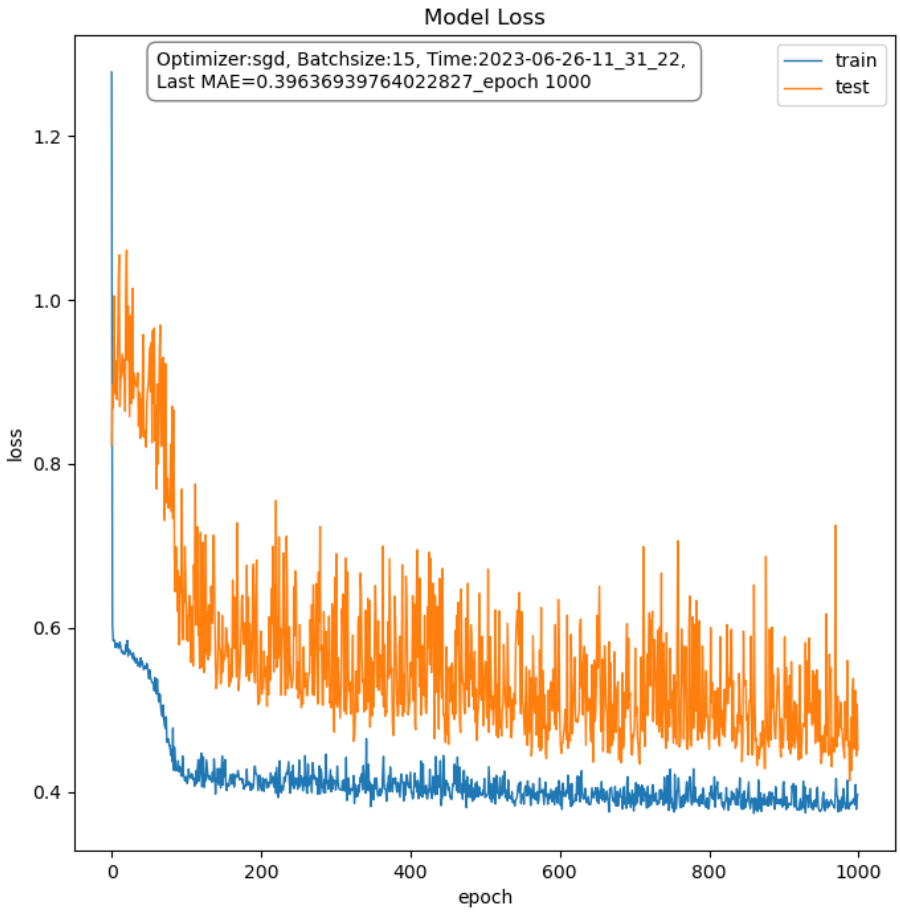


Gambar 4.21 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 7 pada epoch 5000

Pelatihan dilakukan hingga mencapai epoch 1000. Terlihat pada Gambar 4.21, terjadi penurunan signifikan pada nilai *train* dan *test*. Nilai *test* yang awalnya berkisar pada 0.6 mengalami penurunan hingga mencapai nilai 0.4. Sedangkan nilai train mengalami penurunan sedikit demi sedikit pada kisaran nilai 0.4. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.3569, yang masih sedikit jauh dari acuan awal yang telah ditetapkan.

4.2.6. Model ANN dengan Optimizer SGD dan *Batch Size* 15

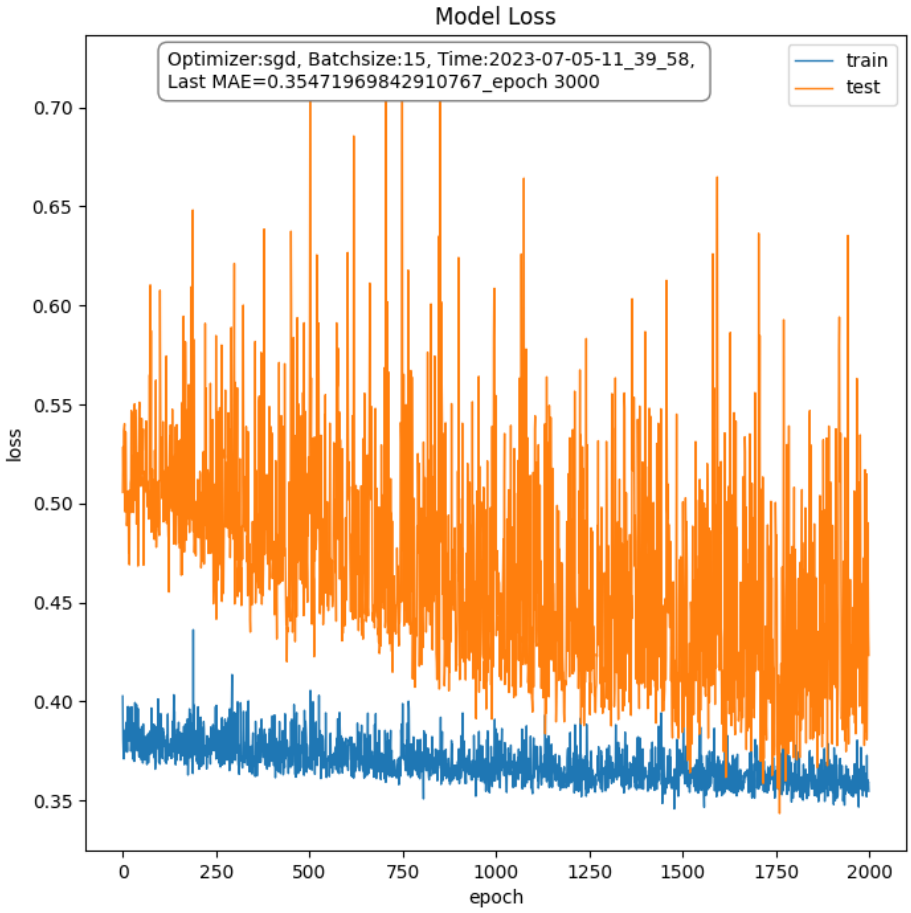
Pada tahap ini, model akan dilatih menggunakan optimizer SGD dengan *batch size* sebesar 15. Perkembangan model pada epoch ke-1000, ke-3000, dan ke-5000 akan dievaluasi. Nilai MAE akan menjadi parameter apakah model sudah memiliki performa yang cukup baik atau belum. Selain itu, akan dilihat pula apakah ada indikasi terjadinya overfitting atau underfitting pada model.



Gambar 4.22 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 15 pada epoch 1000

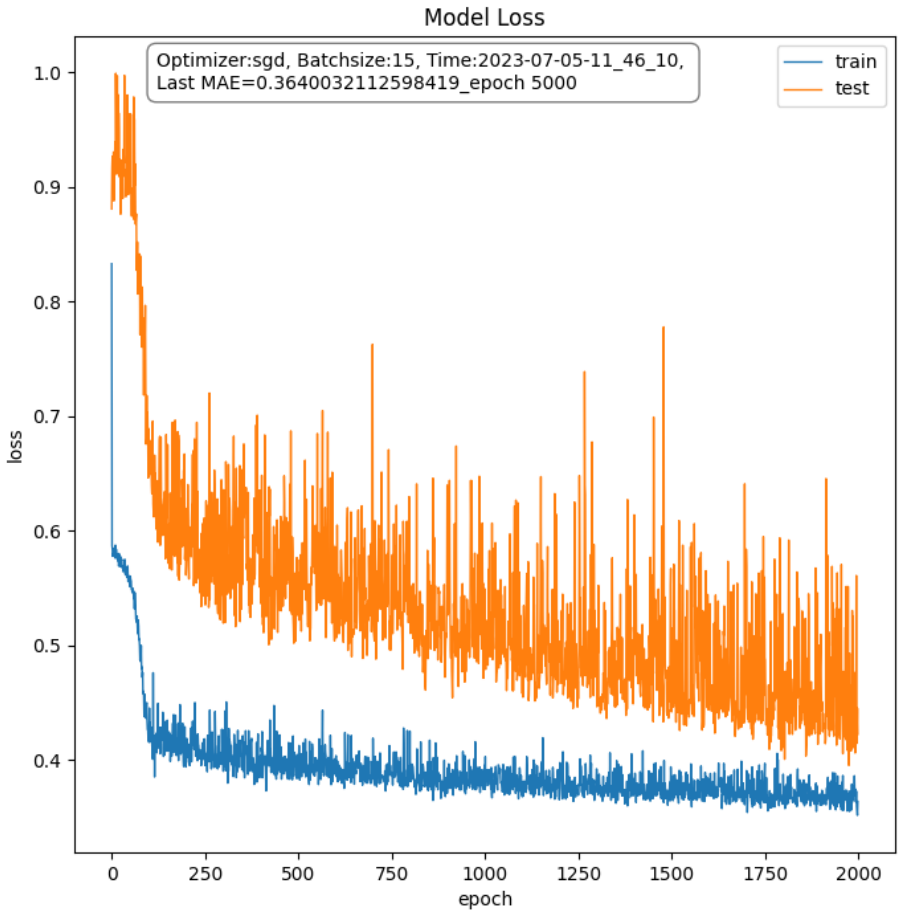
Pelatihan dilakukan hingga mencapai epoch 1000. Terlihat pada Gambar 4.22, nilai loss mae *test* mengalami penurunan yang awalnya berkisar pada nilai 0.6 hingga pada epoch 1000 berkisar pada nilai 0.5. Sedangkan nilai loss mae train tidak mengalami penurunan yang signifikan. Pada epoch 200 hingga epoch 1000 tetap berkisar pada 0.4. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.396369, yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Pelatihan

model kembali dilanjutkan hingga epoch 3000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.23.



Gambar 4.23 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 15 pada epoch 3000

Seperti yang terlihat pada Gambar 4.23. Tidak terjadi perbaikan yang signifikan pada model. Nilai loss MAE pada train maupun *test* tidak mengalami perubahan yang signifikan. Nilai loss MAE pada *test* berfluktuasi antara 0.4 hingga 0.6, sementara nilai loss MAE pada train hanya berkisar antara 0.35 hingga 0.4. Model mencatatkan nilai MAE sebesar 0.360368, dimana nilai tersebut yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Tidak adanya perbaikan yang signifikan merupakan indikasi terjadinya overfitting pada model ini. Pelatihan model kembali dilanjutkan hinnga epoch 5000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.24.



Gambar 4.24 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 15 pada epoch 5000

Pelatihan dilakukan hingga mencapai epoch 5000. Terlihat pada Gambar 4.24, terjadi penurunan signifikan pada nilai train dan *test*. Nilai *test* yang awalnya berkisar pada 0.6 mengalami penurunan hingga mencapai nilai 0.45. Sedangkan nilai train mengalami penurunan sedikit demi sedikit pada kisaran nilai 0.4. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.36400321, dimana nilai tersebut sedikit jauh dari acuan awal yang telah ditetapkan.

Setelah melakukan beberapa eksperimen kombinasi parameter diatas maka didapatkan 2 model yang memiliki nilai mae terkecil yaitu sebesar 0.35695648193359375 pada model yang dilatih menggunakan *batch size* sebesar 7 dan dilatih hingga epoch 5000. Lalu ada pula model yang memiliki nilai mae sebesar 0.35471969842910767 yaitu model yang dilatih menggunakan *batch size* sebesar 15 dan dilatih hingga epoch 3000.

4.2.7. Evaluasi model ANN Prediksi Jarak berdasarkan Citra RGB

Setelah bereksperimen dengan berbagai kombinasi parameter training, maka disimpulkan untuk menggunakan model ANN dengan optimizer adamax, *batch size* 7 pada epoch 1000. Model tersebut memiliki nilai mae 0.333776. Model tersebut digunakan karena tidak mengalami indikasi overfitting maupun underfitting. Sehingga diharapkan model tersebut dapat melakukan prediksi secara stabil. Hasil seluruh eksperimen model ANN dapat dilihat pada tabel 4.2.

Tabel 4.2 Perbandingan Model ANN

Model	Batch Size	Epoch	Last Train MAE
adamax	1	1000	0.5775669217
adamax	1	3000	0.2448507845
adamax	1	5000	0.2388899624
adamax	7	1000	0.3337768912
adamax	7	3000	0.2724565864
adamax	7	5000	0.2823437452
adamax	15	1000	0.3718636036
adamax	15	3000	0.3173060715
adamax	15	5000	0.3213479519
sgd	1	1000	0.3855191469
sgd	1	3000	0.3745284975
sgd	1	5000	0.3702289462
sgd	7	1000	0.3773933947
sgd	7	3000	0.3603681922
sgd	7	5000	0.3569564819
sgd	15	1000	0.3963693976
sgd	15	3000	0.3547196984
sgd	15	5000	0.3640032113

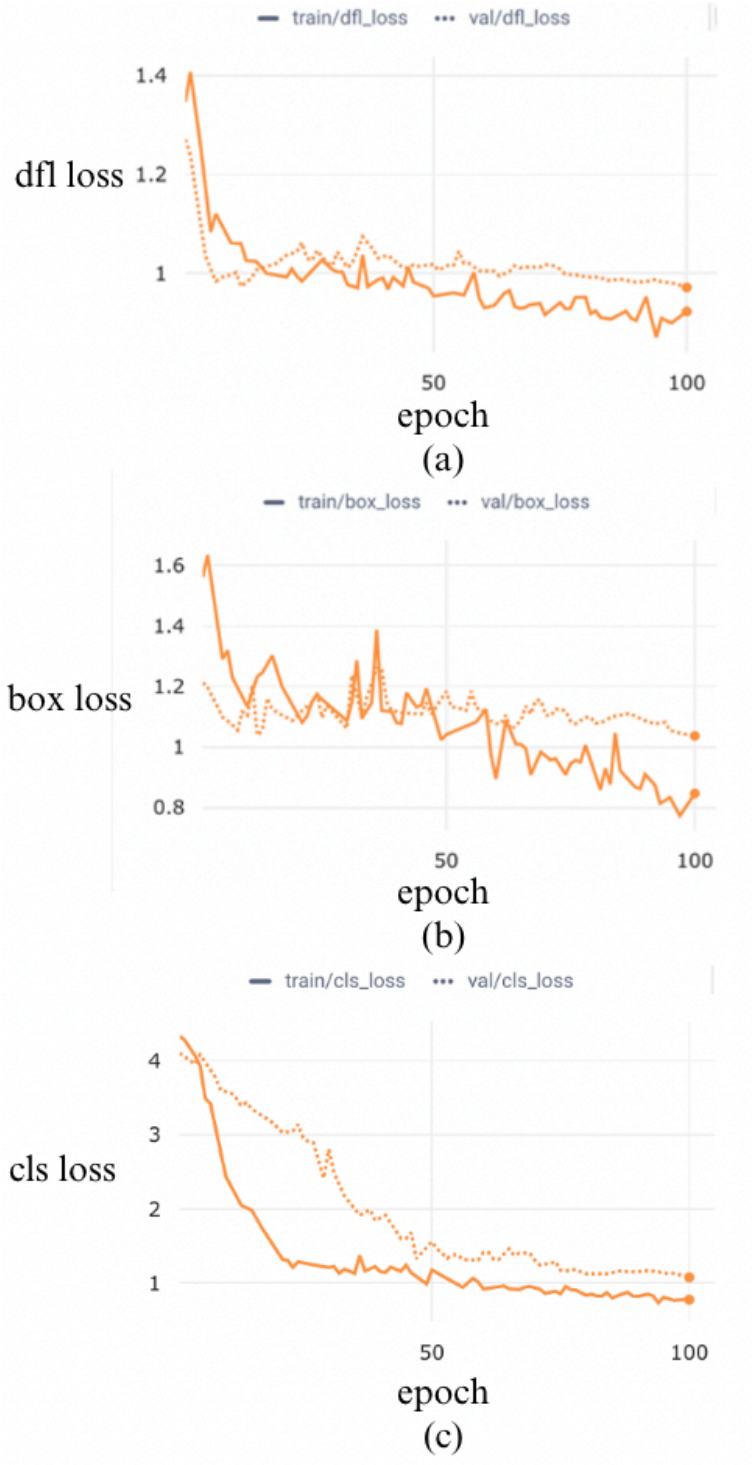
4.3. Eksperimen Model Deteksi Buah Kakao

Pada pelatihan model CNN YOLO, digunakan 80 gambar dan anotasi yang menggambarkan buah kakao matang dan belum matang dengan lima variasi epoch, yaitu 100, 300, 500, 700, 1000. Pada pelatihan ini, menggunakan optimizer SGD dan kombinasi dari beberapa fungsi loss. YOLOv8 menggunakan fungsi loss CIoU dan DFL untuk loss kotak pembatas dan binary cross-entropy untuk loss klasifikasi. Pelatihan akan dilakukan dengan menggunakan 2 skala arsitektur yaitu pada skala nano dan medium. Perbedaan skala arsitektur terdapat pada perbedaan jumlah parameter yang digunakan pada setiap layernya. Model dengan arsitektur nano juga akan memproduksi model dengan ukuran *file* yang lebih kecil ketimbang model dengan medium.

Pelatihan dilakukan selama 100 epoch dengan optimizer SGD dan *batch size* sebesar 16. Ukuran gambar input adalah 800x800, dan model akan disimpan setelah pelatihan selesai. Tidak digunakan cache, dan perangkat yang digunakan akan disesuaikan secara otomatis. Selain itu, dilakukan pengolahan data paralel dengan 8 workers. Informasi tentang pelatihan, seperti verbose, seed, dan deterministic juga telah ditentukan. Model ini dapat mendeteksi multiple kelas, dan tidak menggunakan bobot gambar atau training rectangular. Pada evaluasi, akan diperhatikan nilai ambang batas kepercayaan dan IoU threshold, serta jumlah deteksi maksimum. Grafik pelatihan akan ditampilkan, tetapi tidak akan menampilkan gambar deteksi secara visual. Tidak ada penyimpanan dalam format teks, confidence map, atau crop objek deteksi. Label objek dan confidence score akan ditampilkan, serta tebal garis boks deteksi sebesar 3.

4.3.1. Model YOLOV8n dengan 100 epoch

Dilakukan pelatihan model hingga epoch 100 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan *test* bisa dilihat pada gambar 4.25.



Gambar 4.25 loss model YOLOV8n dengan 100 epoch
(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.25 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-100. Pergerakan

nilai pelatihan *(train)* dan pengujian *(test)* tidak menunjukkan indikasi adanya overfitting atau underfitting. Nilai loss terus menurun seiring berjalannya pelatihan hingga epoch 100. Hasil akhir pelatihan dapat dilihat pada tabel 4.3.

Tabel 4.3 Hasil Pelatihan YOLOV8n pada epoch 100

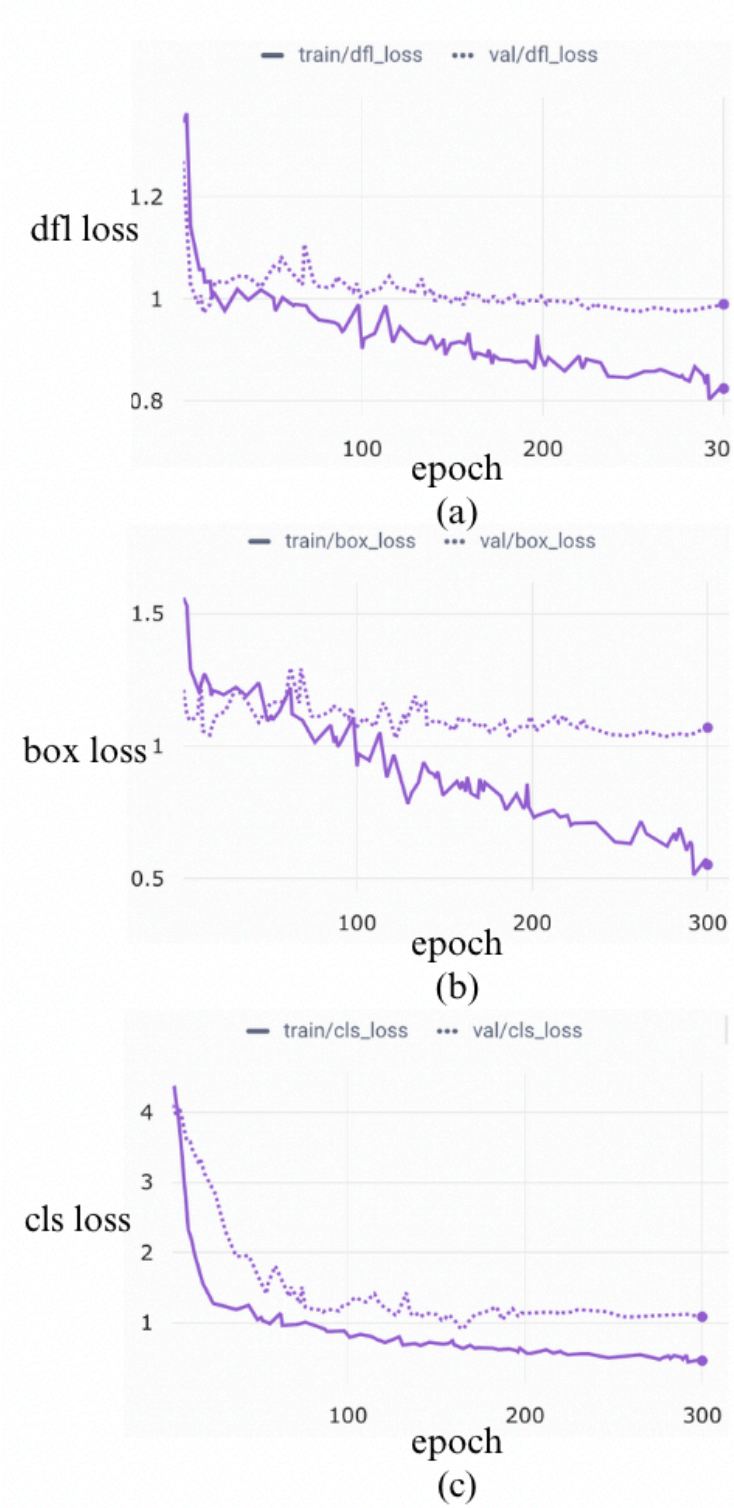
Class	**Box(P)**	**R**
all	0.907	0.704
ripe cocoa	0.925	0.792
unripe cocoa	0.889	0.617

Setelah melatih model YOLOV8M selama 100 epoch, dapat disimpulkan bahwa hasil pelatihan menunjukkan kinerja yang baik dalam melakukan deteksi buah kakao matang dan belum matang. Keseluruhan model mencapai akurasi deteksi sebesar 83.4% dengan recall sebesar 82%. Hasil yang lebih baik diperoleh untuk deteksi buah kakao matang, dengan akurasi mencapai 88% dan recall sebesar 91.7%. Namun, performa deteksi pada buah kakao belum matang sedikit lebih rendah, dengan akurasi sebesar 78.8% dan recall sebesar 72.3%.

Meskipun demikian, secara keseluruhan model telah mampu melakukan deteksi dengan baik pada dataset yang digunakan. Namun, ada ruang untuk pengembangan lebih lanjut dalam meningkatkan performa deteksi pada buah kakao belum matang agar sejajar dengan deteksi buah kakao matang. Dengan demikian, dapat dilakukan penyesuaian atau peningkatan model untuk mencapai akurasi deteksi yang lebih tinggi dan recall yang lebih baik pada kelas buah kakao belum matang.

4.3.2. Model YOLOV8n dengan 300 epoch

Dilakukan pelatihan model hingga epoch 300 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan *test* bisa dilihat pada gambar 4.26.



Gambar 4.26 loss model YOLOV8n dengan 300 epoch
(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.26 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-300. Nilai loss

terus menurun seiring berjalannya pelatihan hingga epoch 300. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss *train* dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.4.

Tabel 4.4 Hasil Pelatihan YOLOV8n pada epoch 300

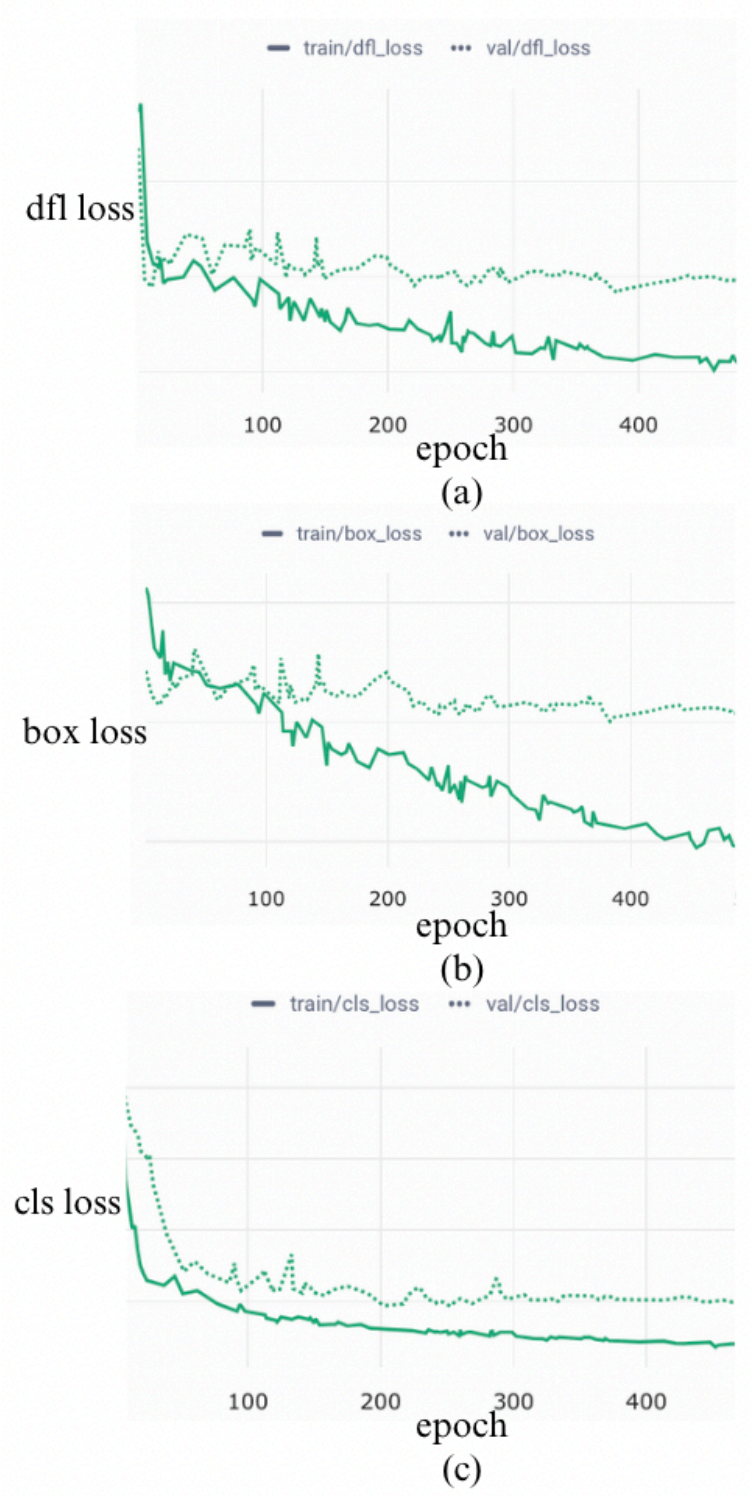
Class	**Box(P)**	**R**
all	0.799	0.799
ripe cocoa	0.734	0.920
unripe cocoa	0.864	0.678

Setelah melatih model, diperoleh hasil pelatihan yang menunjukkan kinerja yang cukup baik dalam melakukan deteksi buah kakao matang dan belum matang. Dalam hal ini, model mencapai akurasi deteksi sebesar 79.9% dan recall (kemampuan mengidentifikasi dengan benar) sebesar 79.9% untuk semua kelas. Meskipun akurasi dan recall secara keseluruhan seimbang, terdapat perbedaan dalam kinerja deteksi antara kelas buah kakao matang dan belum matang.

Untuk kelas buah kakao matang, model mencapai akurasi deteksi sebesar 73.4% dengan recall sebesar 92%. Hal ini menunjukkan bahwa model mampu mengenali dengan baik buah kakao yang telah matang. Namun, untuk kelas buah kakao belum matang, model memiliki akurasi deteksi yang sedikit lebih tinggi sebesar 86.4% namun recall yang sedikit lebih rendah, yakni sebesar 67.8%. Secara keseluruhan, model telah mencapai kinerja yang cukup baik dalam melakukan deteksi buah kakao matang dan belum matang.

4.3.3. Model YOLOV8n dengan 500 epoch

Dilakukan pelatihan model hingga epoch 500 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan *test* dapat dilihat pada gambar 4.27.



Gambar 4.27 loss model YOLOV8n dengan 500 epoch
(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.27 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-200. Nilai loss

terus menurun seiring berjalannya pelatihan hingga epoch 200. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss *train* dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.5.

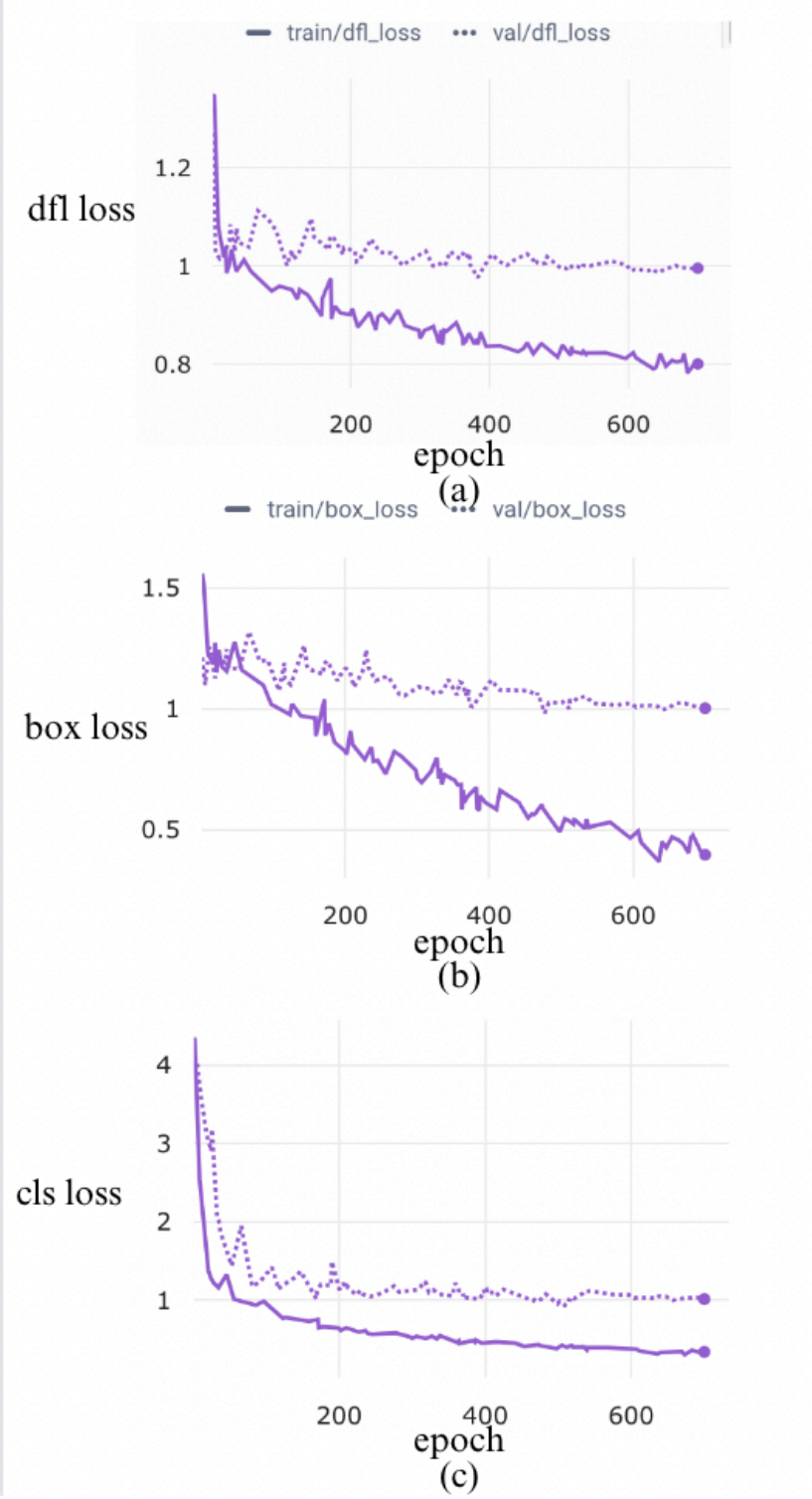
Tabel 4.5 Hasil Pelatihan YOLOV8n pada epoch 500

Class	**Box(P)**	**R**
all	0.860	0.809
ripe cocoa	0.867	1.000
unripe cocoa	0.853	0.618

Berdasarkan hasil pelatihan model YOLOV8N selama 500 epoch, secara keseluruhan model ini memiliki tingkat akurasi deteksi yang baik sebesar 0,86 dan presisi yang cukup tinggi sebesar 0,809. Model ini mampu dengan efektif mendeteksi dan mengklasifikasikan cokelat matang ("ripe cocoa") dengan tingkat akurasi yang tinggi sebesar 0,867 dan recall sempurna sebesar 1. Namun, model menghadapi beberapa kesulitan dalam mendeteksi cokelat yang belum matang ("unripe cocoa"), dengan tingkat akurasi yang sedikit lebih rendah sebesar 0,853 dan recall sebesar 0,618. Untuk meningkatkan performa model dalam mengklasifikasikan cokelat yang belum matang, perlu dilakukan penyesuaian pada proses pelatihan, seperti penambahan data latihan yang lebih representatif atau penyetelan parameter model yang lebih optimal. Secara keseluruhan, meskipun model YOLOV8N telah menunjukkan kinerja yang baik dalam mendeteksi objek secara umum, masih diperlukan peningkatan dalam mengklasifikasikan cokelat yang belum matang dengan lebih baik.

4.3.4. Model YOLOV8n dengan 700 epoch

Dilakukan pelatihan model hingga epoch 700 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan *test* bisa dilihat pada gambar 4.28.



Gambar 4.28 loss model YOLOV8n dengan 700 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.28 terlihat bahwa model berhasil meningkatkan akurasi dengan baik. Nilai loss terus menurun seiring

berjalannya pelatihan model. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.6.

Tabel 4.6 Hasil Pelatihan YOLOV8n pada epoch 700

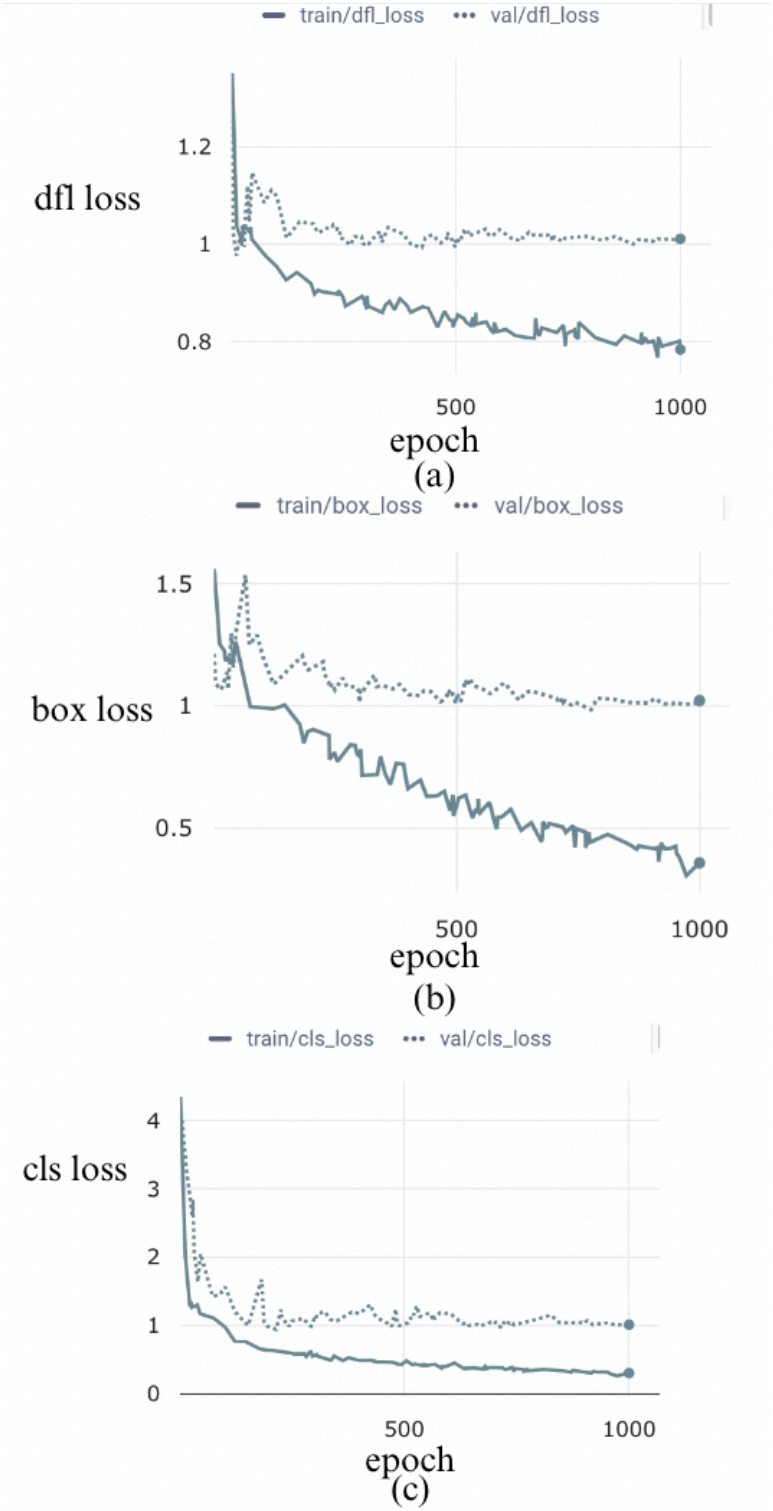
Class	**Box(P)**	**R**
all	0.917	0.707
ripe cocoa	0.947	0.743
unripe cocoa	0.887	0.671

Dalam hasil pelatihan model YOLOV8N selama 700 epoch, terjadi peningkatan yang signifikan dalam kinerja model. Secara keseluruhan, model mencapai tingkat akurasi deteksi yang tinggi sebesar 0,917 dengan nilai presisi sebesar 0,707. Ini menunjukkan bahwa model berhasil meningkatkan kemampuannya dalam mendeteksi objek secara umum. Hasil yang lebih baik ini dapat memberikan kepercayaan lebih dalam penggunaan model untuk mendeteksi berbagai objek di dalam gambar.

Ketika fokus pada kategori "ripe cocoa", model menunjukkan peningkatan yang konsisten dengan tingkat akurasi sebesar 0,947 dan presisi sebesar 0,743. Meskipun recall masih dapat ditingkatkan, peningkatan ini menunjukkan bahwa model semakin mampu mengklasifikasikan cokelat matang dengan akurasi yang lebih tinggi. Namun, untuk kategori "unripe cocoa", meskipun terjadi peningkatan, model masih menghadapi beberapa tantangan dalam mendeteksi dan mengklasifikasikan cokelat yang belum matang. Dengan tingkat akurasi sebesar 0,887 dan recall sebesar 0,671, masih ada ruang untuk perbaikan lebih lanjut.

4.3.5. Model YOLOV8n dengan 1000 epoch

Dilakukan pelatihan model hingga epoch 100 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.29.



Gambar 4.29 dfl loss, box loss, cls loss model YOLOV8n dengan 1000 epoch (a) dfl loss, (b) box loss dan (c) cls

Seperti yang dapat dilihat pada ketiga gambar 4.29 terlihat bahwa model berhasil meningkatkan akurasi dengan baik. Nilai loss terus menurun seiring

berjalannya pelatihan model. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-400 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil Pelatihan YOLOV8n pada epoch 1000

Class	**Box(P)**	**R**
all	0.869	0.767
ripe cocoa	0.775	0.917
unripe cocoa	0.963	0.617

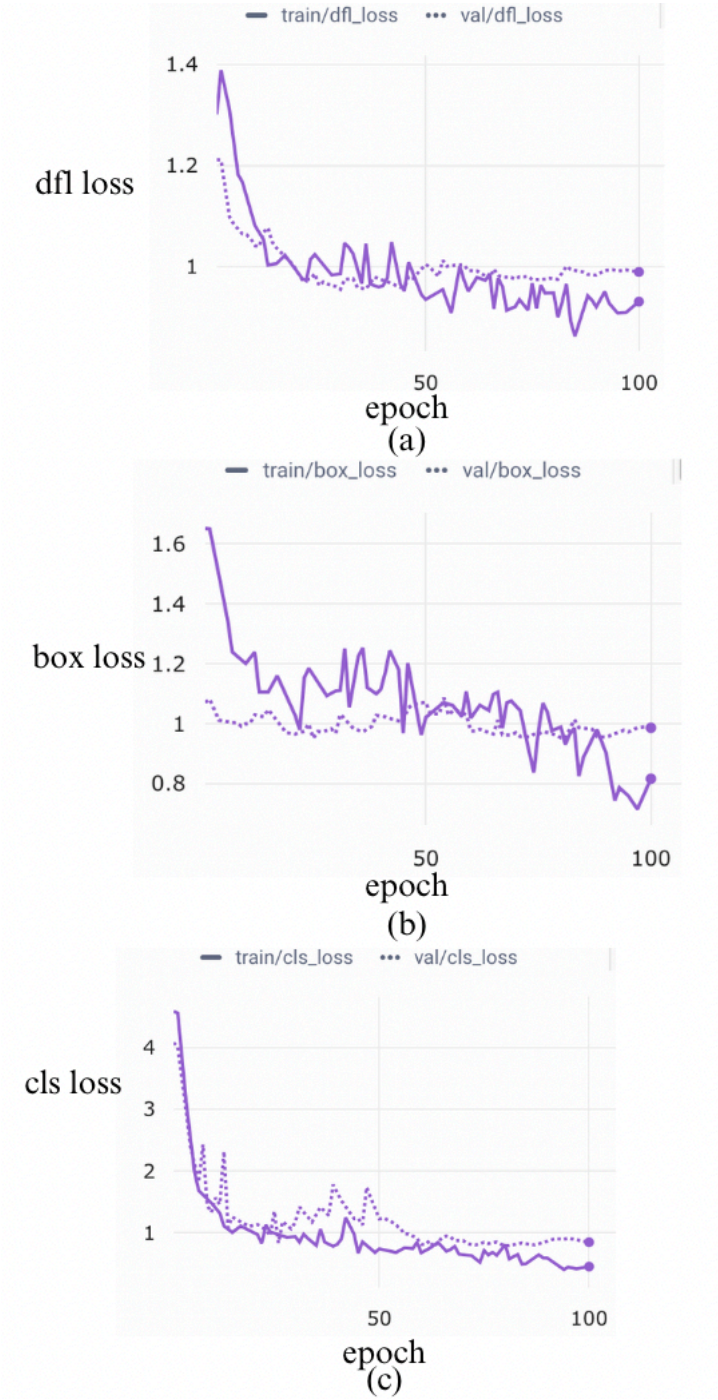
Dalam pelatihan model YOLOV8N, terdapat hasil yang menarik untuk setiap kategori. Secara keseluruhan, model mencapai tingkat akurasi deteksi sebesar 0,869 dengan presisi sebesar 0,767. Meskipun tingkat presisi yang tinggi menunjukkan kemampuan model dalam mengenali objek secara spesifik, recall yang sebesar 0,767 menandakan adanya ruang untuk perbaikan dalam mencakup semua objek yang ada dalam gambar secara lebih lengkap.

Dalam kategori "ripe cocoa", model menunjukkan tingkat presisi sebesar 0,775 yang cukup baik. Namun, recall yang rendah sebesar 0,917 mengindikasikan bahwa model masih melewatkan beberapa cokelat matang yang seharusnya terdeteksi. Sementara itu, dalam kategori "unripe cocoa", model berhasil mencapai tingkat presisi yang tinggi sebesar 0,963. Namun, recall yang rendah sebesar 0,617 menunjukkan bahwa model masih menghadapi kesulitan dalam mendeteksi sebagian besar cokelat yang belum matang.

Secara keseluruhan, model YOLOV8N telah menunjukkan kemajuan yang baik dalam deteksi objek, namun masih ada aspek-aspek yang perlu ditingkatkan. Peningkatan pada recall dalam kedua kategori "ripe cocoa" dan "unripe cocoa" akan menjadi prioritas dalam pelatihan selanjutnya. Dengan demikian, model dapat mengenali dan mengklasifikasikan objek dengan lebih akurat dan menyeluruh, menghasilkan hasil yang lebih baik dalam aplikasi deteksi objek yang berkaitan dengan cokelat..

4.3.6. Model YOLOV8m dengan 100 epoch

Dilakukan pelatihan model hingga epoch 100 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.30.



Gambar 4.30 loss model YOLOV8n dengan 100 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.30 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-100. Pergerakan nilai pelatihan *(train)* dan pengujian *(test)* tidak menunjukkan indikasi adanya *overfitting* atau *underfitting*. Nilai *loss* terus menurun seiring berjalannya pelatihan hingga epoch 100. Hasil akhir pelatihan dapat dilihat pada tabel 4.8.

Tabel 4.8 Hasil Pelatihan YOLOV8M pada epoch 100

Class	**Box(P)**	**R**
all	0.834	0.820
ripe cocoa	0.880	0.917
unripe cocoa	0.788	0.723

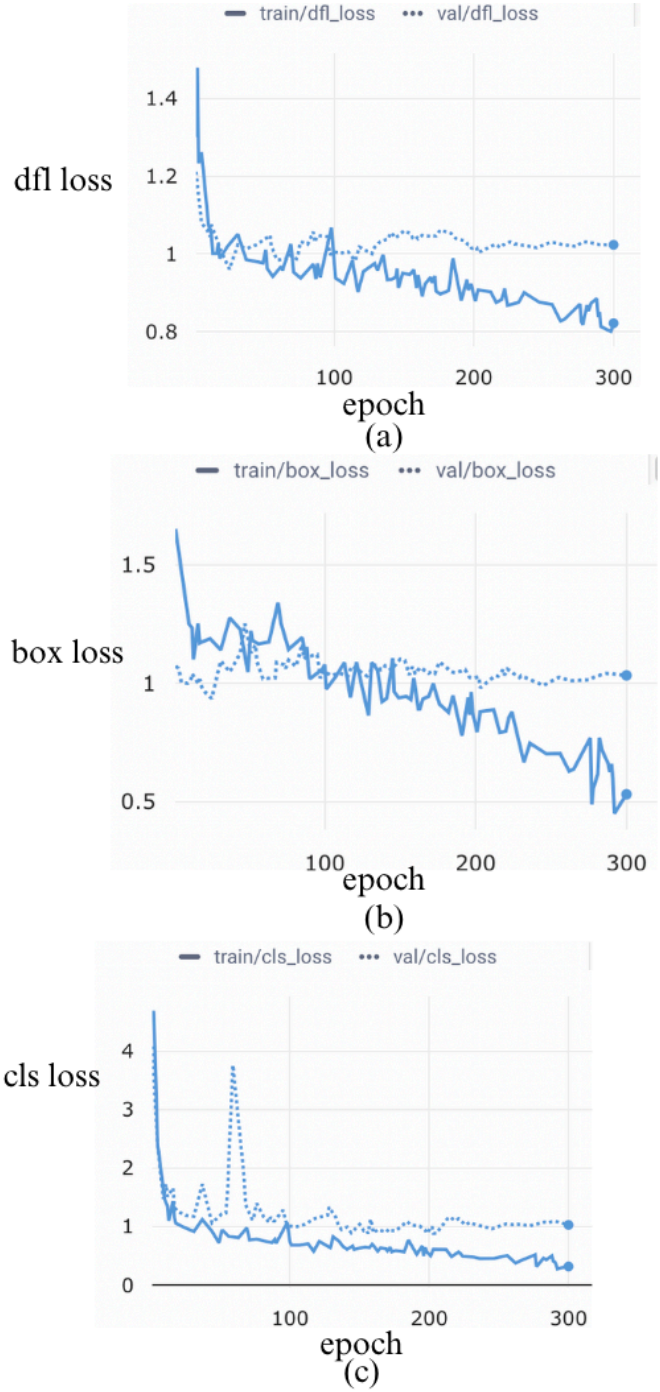
Model YOLOV8M, setelah melalui pelatihan selama 100 epoch, menunjukkan hasil yang cukup baik. Dalam kategori "all", model mencapai tingkat akurasi deteksi sebesar 0,834 dengan presisi sebesar 0,82. Hasil ini menunjukkan kemampuan model dalam mendeteksi objek secara umum dalam dataset yang digunakan. Tingkat akurasi yang tinggi ini memberikan kepercayaan bahwa model dapat mengenali objek dengan baik.

Ketika berfokus pada kategori "ripe cocoa", model YOLOV8M mencapai tingkat akurasi yang lebih tinggi sebesar 0,88 dengan recall sebesar 0,917. Hal ini menunjukkan kemampuan model dalam mengklasifikasikan cokelat matang dengan akurasi yang baik dan mampu mendeteksi sebagian besar objek yang ada. Tingkat recall yang tinggi juga menandakan bahwa model dapat mengenali sebagian besar cokelat matang yang ada dalam dataset.

Namun, dalam kategori "unripe cocoa", model menghadapi beberapa tantangan dengan tingkat akurasi sebesar 0,788 dan recall sebesar 0,723. Hal ini mengindikasikan bahwa model masih perlu ditingkatkan dalam mendeteksi dan mengklasifikasikan cokelat yang belum matang. Secara keseluruhan, model YOLOV8M menunjukkan kemampuan yang baik dalam mendeteksi objek secara umum dan khususnya dalam kategori "ripe cocoa".

4.3.7. Model YOLOV8m dengan 300 epoch

Dilakukan pelatihan model hingga epoch 300 menggunakan arsitektur YOLOV8m. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.31.



Gambar 4.31 dfl loss, box loss, cls loss model YOLOV8n dengan 300 epoch (a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.31 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-200. Nilai loss terus menurun seiring berjalannya pelatihan. Namun pada metrik dfl loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.9.

Tabel 4.9 Hasil Pelatihan YOLOV8m pada epoch 300

Class	**Box(P)**	**R**
all	0.820	0.725
ripe cocoa	0.903	0.875
unripe cocoa	0.737	0.574

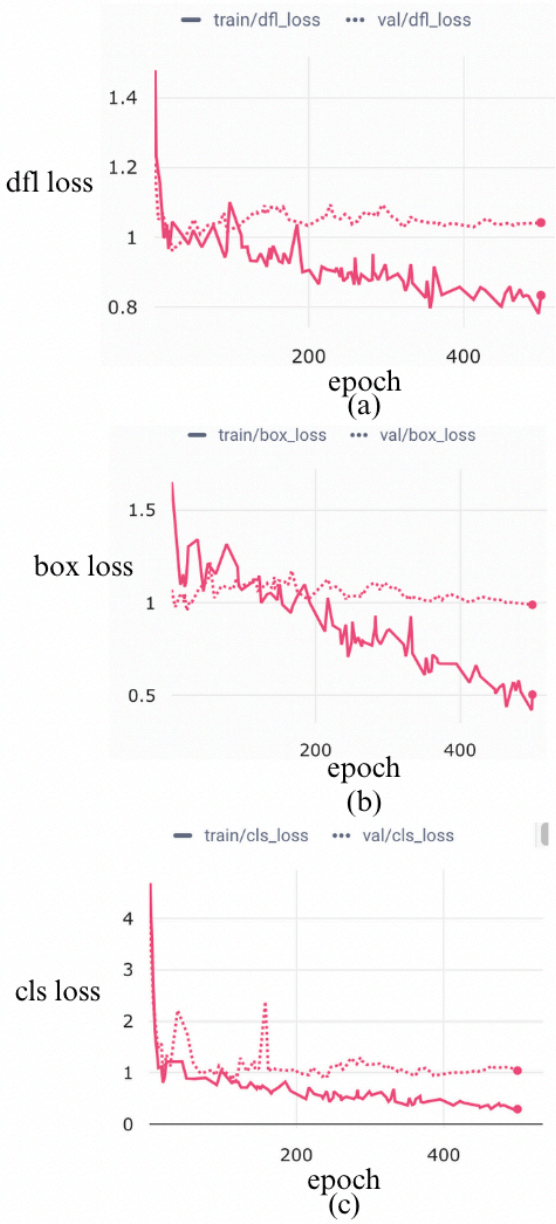
Setelah melalui pelatihan selama 300 epoch, model YOLOV8M menunjukkan hasil yang menarik. Secara keseluruhan, model ini mencapai tingkat akurasi deteksi sebesar 0,82 dengan presisi sebesar 0,725 dalam kategori "all". Meskipun tingkat akurasi yang cukup baik, recall yang sebesar 0,725 menunjukkan bahwa model mungkin masih melewatkan sebagian objek yang ada dalam gambar. Dalam kategori "ripe cocoa", model menunjukkan peningkatan performa dengan tingkat akurasi sebesar 0,903 dan recall sebesar 0,875. Hasil ini mengindikasikan kemampuan model dalam mengklasifikasikan cokelat matang dengan akurasi yang tinggi dan mampu mendeteksi sebagian besar objek yang ada. Peningkatan tersebut menunjukkan adanya kemajuan dalam pelatihan model.

Namun, dalam kategori "unripe cocoa", model masih mengalami beberapa kendala dengan tingkat akurasi sebesar 0,737 dan recall sebesar 0,574. Hal ini menandakan bahwa model masih menghadapi kesulitan dalam mendeteksi dan mengklasifikasikan cokelat yang belum matang dengan akurasi dan kelengkapan yang lebih baik. Secara keseluruhan, model YOLOV8M telah menunjukkan kemajuan dalam pelatihan selama 300 epoch. Meskipun tingkat akurasi dan presisi

dalam kategori "all" cukup baik, recall masih perlu ditingkatkan. Peningkatan signifikan terlihat dalam kategori "ripe cocoa" dengan akurasi dan recall yang lebih baik.

4.3.8. Model YOLOV8m dengan 500 epoch

Dilakukan pelatihan model hingga epoch 500 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.32.



Gambar 4.32 dfl loss, box loss, cls loss model YOLOV8n dengan 500 epoch (a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.32 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-300. Nilai loss terus menurun seiring berjalannya pelatihan hingga epoch 300. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.10.

Tabel 4.10 Hasil Pelatihan YOLOV8m pada epoch 500

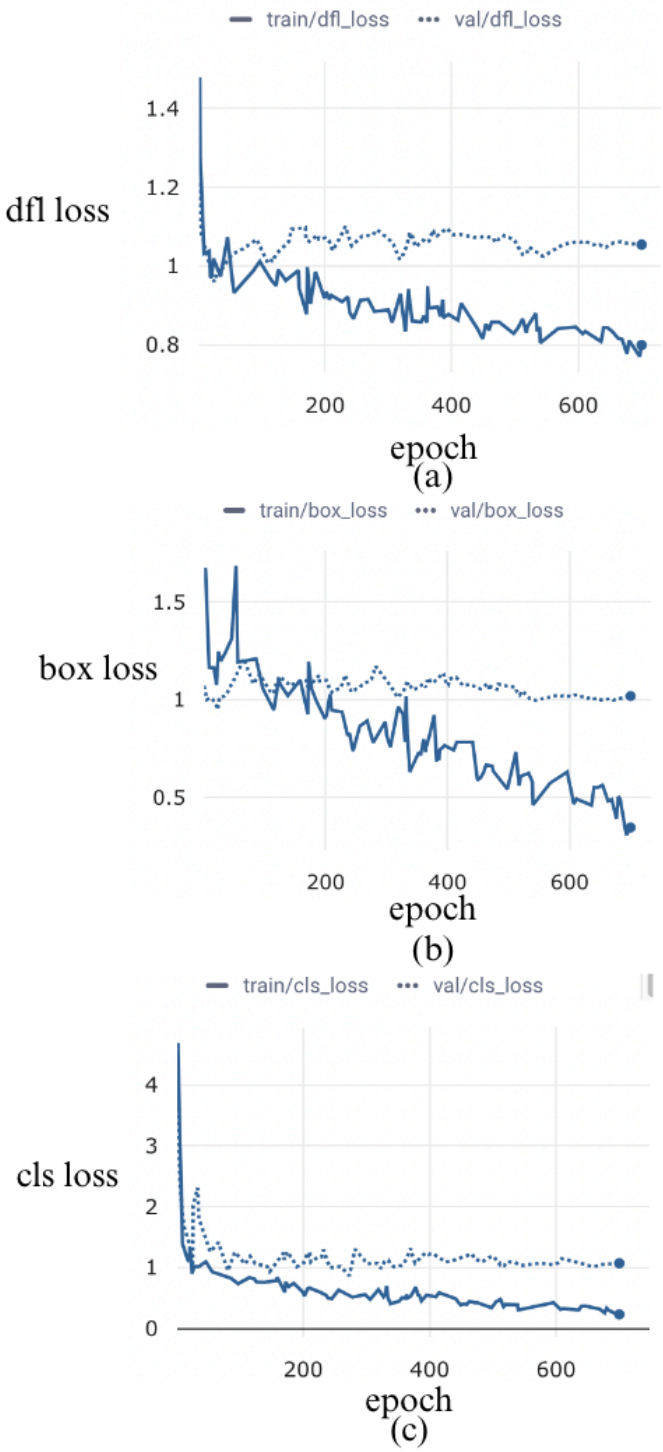
Class	**Box(P)**	**R**
all	0.910	0.704
ripe cocoa	0.959	0.792
unripe cocoa	0.861	0.617

Setelah melalui pelatihan selama 500 epoch, model YOLOV8M menunjukkan hasil yang menggembirakan. Secara keseluruhan, model ini mencapai tingkat akurasi deteksi yang tinggi sebesar 0,91 dengan presisi sebesar 0,704 dalam kategori "all". Meskipun tingkat akurasi yang tinggi, recall yang sebesar 0,704 menunjukkan bahwa model mungkin masih melewatkan sebagian objek yang ada dalam gambar.

Dalam kategori "ripe cocoa", model menunjukkan performa yang sangat baik dengan tingkat akurasi sebesar 0,959 dan recall sebesar 0,792. Hasil ini menunjukkan kemampuan model dalam mengklasifikasikan cokelat matang dengan akurasi yang tinggi dan mampu mendeteksi sebagian besar objek yang ada. Namun, dalam kategori "unripe cocoa", model masih menghadapi beberapa kendala dengan tingkat akurasi sebesar 0,861 dan recall sebesar 0,617. Hal ini menandakan bahwa model masih mengalami kesulitan dalam mendeteksi dan mengklasifikasikan cokelat yang belum matang dengan akurasi dan kelengkapan yang lebih baik.

4.3.9. Model YOLOV8m dengan 700 epoch

Dilakukan pelatihan model hingga epoch 700 menggunakan arsitektur YOLOV8m. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.33.



Gambar 4.33 loss model YOLOV8n dengan 700 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.33 terlihat bahwa model berhasil meningkatkan akurasi dengan baik. Nilai loss terus menurun seiring berjalannya pelatihan hingga epoch 300. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.11.

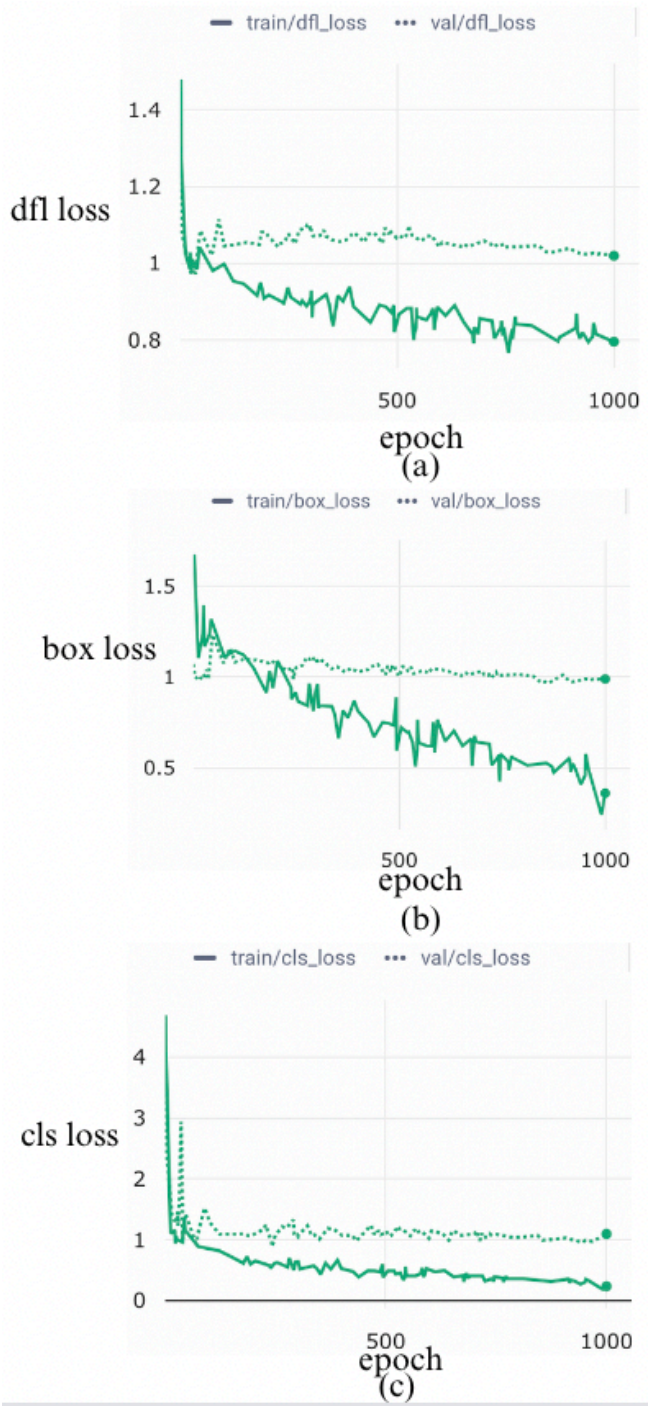
Tabel 4.11 Hasil Pelatihan YOLOV8m pada epoch 700

Class	**Box(P)**	**R**
all	0.748	0.808
ripe cocoa	0.610	0.958
unripe cocoa	0.885	0.658

Model YOLOV8M dilatih selama 700 epoch dan mencapai akurasi deteksi sebesar 0,748 dan presisi 0,808 dalam kategori "semua". Namun, recall 0,808 menunjukkan ruang untuk meningkatkan deteksi objek dalam gambar. Dalam kategori "kakao matang", akurasi 0,61 dengan recall tinggi 0,958 menunjukkan kemampuan model mengklasifikasikan kakao matang, meskipun dengan positif palsu. Dalam kategori "kakao mentah", akurasi 0,885 dengan recall 0,658 menunjukkan model masih kesulitan mendeteksi kakao mentah. Model perlu meningkatkan recall dalam kategori ini. Secara keseluruhan, YOLOV8M menunjukkan kemajuan selama 700 epoch. Dalam kategori "semua", recall perlu ditingkatkan untuk mendeteksi lebih banyak objek. Dalam kategori "kakao matang", recall tinggi menunjukkan kemampuan model, tetapi perlu penanganan positif palsu. Dalam kategori "kakao mentah", meningkatkan recall menjadi fokus utama untuk meningkatkan deteksi kakao mentah dengan akurasi yang lebih tinggi.

4.3.10. Model YOLOV8m dengan 1000 epoch

Dilakukan pelatihan model hingga epoch 1000 menggunakan arsitektur YOLOV8m. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.34.



Gambar 4.34 model YOLOV8n dengan 1000 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.34 terlihat bahwa model berhasil meningkatkan akurasi dengan baik. Nilai loss terus menurun seiring berjalannya pelatihan hingga epoch 300. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai *loss train* dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.12.

Tabel 4.12 Hasil Pelatihan YOLOV8m pada epoch 1000

Class	**Box(P)**	**R**
all	0.816	0.768
ripe cocoa	0.746	0.856
unripe cocoa	0.886	0.681

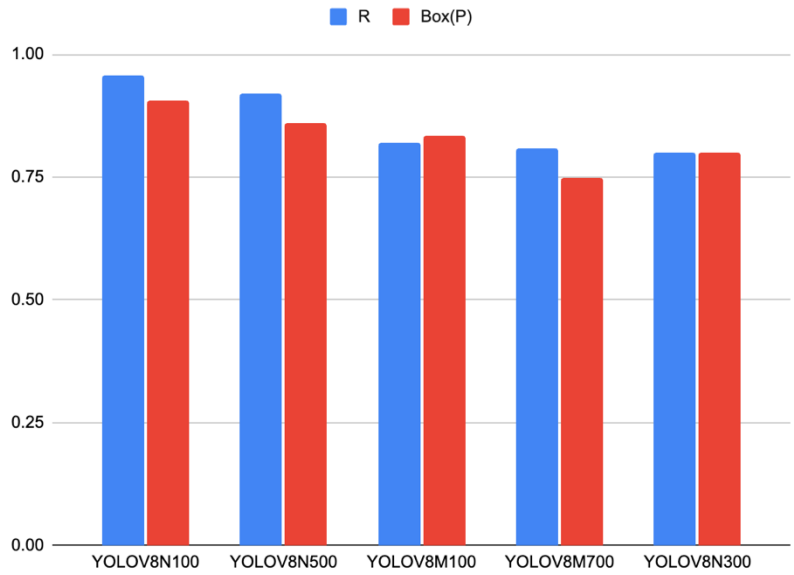
Model YOLOV8M telah dilatih selama 1000 epoch dan menunjukkan hasil yang menarik. Dalam kategori "semua", model ini mencapai presisi sebesar 0,816 dan recall sebesar 0,768. Ini menunjukkan kemampuan model dalam mengenali dan mengklasifikasikan objek dengan akurasi yang cukup tinggi. Dalam kategori "kakao matang", model mencapai presisi sebesar 0,746 dengan recall yang lebih tinggi, yaitu 0,856. Ini menunjukkan kemampuan model dalam mendeteksi dan mengklasifikasikan kakao matang dengan baik, meskipun masih terdapat ruang untuk meningkatkan presisi. Di sisi lain, dalam kategori "kakao mentah", model mencapai presisi sebesar 0,886 dengan recall sebesar 0,681.

Meskipun akurasi cukup baik, recall yang rendah menunjukkan bahwa model masih kesulitan dalam mendeteksi sebagian besar kakao mentah. Meningkatkan recall dalam kategori ini menjadi fokus utama untuk meningkatkan kemampuan model dalam mengklasifikasikan kakao mentah dengan akurasi yang lebih tinggi. Secara keseluruhan, YOLOV8M telah menunjukkan kemajuan dalam 1000 epoch pelatihan. Meskipun memiliki akurasi yang baik dalam kategori "semua" dan "kakao matang", meningkatkan recall akan membantu model dalam menangkap lebih banyak objek secara akurat. Untuk kategori "kakao mentah",

meningkatkan recall menjadi hal yang penting untuk meningkatkan kemampuan model dalam mendeteksi kakao mentah dengan lebih baik.

4.3.11. Evaluasi 5 Model YOLOV8 dengan performa terbaik

Untuk menentukan model YOLOV8 yang memiliki akurasi paling baik maka berikut akan ditampilkan 5 model dengan nilai R dan Box(P) (box loss) yang paling tinggi diantara model-model lainnya pada gambar 4.35.



Gambar 4.35 Grafik 5 Model Dengan Nilai Recall Tertinggi

Dari model dengan arsitektur YOLOV8 diatas maka disimpulkan model terbaik adalah model dengan arsitektur YOLOV8 nano dengan 100 epoch.

4.4. Sistem Prediksi Jarak Objek pada Citra

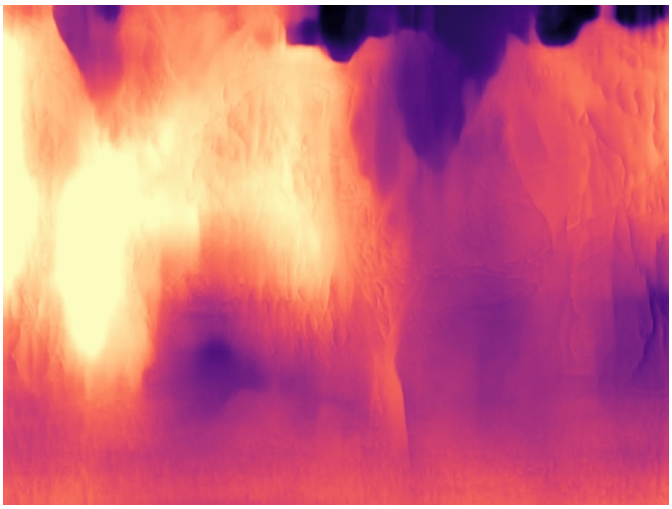
Sistem ini akan melakukan Prediksi Jarak Objek terhadap kamera. Prediksi dilakukan menggunakan nilai *rgb* sebagai input. Proses dimulai dengan memilih titik koordinat salah satu piksel yang berada pada area objek sebagai input proses segmentasi. Kemudian dilakukan segmentasi pada citra untuk mendapatkan area objek. Kemudian dilakukan prediksi kedalaman citra menggunakan Model CNN *Monocular Depth Estimation.* Area yang telah didapatkan digunakan untuk mengambil warna pada citra hasil prediksi kedalaman pada area yang diinginkan.

Kemudian dari kumpulan warna tersebut dicari nilai mediannya sehingga didapatkan nilai median rgb untuk digunakan pada input model ANN prediksi jarak. Sehingga kemudian didapatkan nilai jarak objek terhadap kamera.

4.4.1. Mengunggah Gambar dan Prediksi Kedalaman

Proses awal yang harus dilakukan yakni mengunggah gambar tanaman kakao yang akan digunakan. Setelah itu dilakukan proses prediksi kedalaman menggunakan Model CNN *Monocular Depth Estimation.* Sehingga dihasilkan citra yang merepresentasikan kedalaman atau dapat disebut sebagai citra kedalaman. Citra inilah yang nantinya akan digunakan nilai rgbnya sebagai input untuk memprediksi jarak objek terhadap kamera.

Gambar 4.36 Citra Tanaman Kakao Asli



Gambar 4.37 Citra Kedalaman Tanaman Kakao

4.4.2. Menentukan Titik Piksel pada Objek

Karena sistem yang dikembangkan belum dapat mengidentifikasi bagian batang tanaman kakao secara otomatis, maka pada penelitian ini masih diperlukan *input manual* untuk menentukan titik piksel yang merupakan bagian dari tanaman kakao. Proses ini dilakukan dengan melihat terlebih dahulu gambar serta *axis* untuk melakukan taksiran lokasi x,y piksel yang akan digunakan.

Gambar 4.38 Tampilan awal untuk penentuan titik pada objek

Setelah melakukan taksiran maka pada gambar ini akan digunakan titik pada koordinat 2750, 3000. Nilai x,y tersebut akan ditampilkan dengan simbol bintang seperti pada gambar berikut.

Gambar 4.39 Tampilan gambar dengan titik piksel yang telah dipilih

Setelah menemukan titik piksel yang akan digunakan, maka pada proses selanjutnya titik tersebut akan digunakan sebagai input pada proses segmentasi.

4.4.3. Proses Segmentasi dengan Model CNN *Segment Anything*

Pada proses ini akan dilakukan segmentasi untuk mendapatkan area objek yang akan digunakan (Tanaman Kakao). Titik koordinat yang telah ditentukan menjadi input/acuan model untuk memprediksi area sekitarnya yang masih merupakan bagian dari objek. Hal ini dilakukan untuk mempermudah pemilihan area objek.

Gambar 4.40 Hasil Segmentasi Area Objek Tanaman Kakao

Area/*mask* tersebut kemudian akan digunakan untuk mengambil nilai rgb pada citra kedalaman.

4.4.4. Pengambilan nilai median RGB dan Prediksi Jarak

Setelah mendapatkan area objek tanaman kakao, maka nilai rgb pada citra kedalaman yang beririsan dengan area objek tanaman kakao akan diambil. Dari nilai-nilai tersebut kemudian akan didapatkan nilai median RGB. Nilai median

RGB kemudian digunakan untuk memprediksi jarak. Dengan menggunakan model ANN yang telah dikembangkan, dilakukan prediksi jarak menggunakan nilai rgb sebagai input. Sehingga kemudian didapatkan nilai jarak objek tanaman kakao terhadap kamera dalam satuan meter.

4.4.5. Evaluasi Sistem Prediksi Jarak Objek Pada Citra

Penggunaan model CNN *Segment Anything* belum bisa optimal untuk mengambil bagian tanaman saja karena adanya noise pada background objek tersebut. Hal ini dipengaruhi oleh kondisi lingkungan perkebunan kakao yang padat dengan tanaman kakao.

4.5. Sistem Prediksi Koordinat Tanaman Kakao pada Citra

Pada bagian ini akan dilakukan prediksi nilai koordinat tanaman kakao. Pada tahap ini diperlukan beberapa variabel sebagai input yaitu nilai koordinat longitude dan latitude tanaman kakao, nilai derajat arah hadap kamera, serta jarak objek terhadap kamera. Pada proses ini akan digunakan rumus yang bernama *Vincenty Formula.* Hasil dari kalkulasi menggunakan rumus tersebut yakni titik koordinat longitude serta latitude objek tanaman kakao. Untuk menguji akurasi dari rumus tersebut, pada bagian selanjutnya akan dilakukan percobaan kalkulasi pada 5 titik lokasi.

4.5.1. Evaluasi Akurasi *Vincenty Formula*

Pada bagian ini akan dilakukan pengujian akurasi dari hasil kalkulasi *Vinceny Formula.* Pengujian akan dilakukan pada 5 lokasi pada tabel 4.13.

Tabel 4.13 Ground Truth Titik Koordinat

No	Building		Destination	
	Name	Coordinate	Name	Coordinate
1	Alun-Alun Tugu Malang	-7.97692544511029, 112.634055029002	Stasiun Malang Kota Baru	-7.97720169566579, 112.637112747216

Tabel 4. 14 Ground Truth Titik Koordinat (lanjutan)

No	Building		Destination	
	Name	Coordinate	Name	Coordinate
2	Alun-Alun Tugu Malang	-7.97692544511029, 112.634055029002	Bulan Photocopy & Print	-7.97419601839356, 112.634292801417
3	Gerbang UB Soekarno hatta	-7.94984, 112.615411	Soekarno Hatta Bridge	-7.9496079959561, 112.615839888356
4	Gerbang UB Soekarno hatta	-7.94984, 112.615411	Kober Mie Setan	-7.9481629265072, 112.61676260122
5	Gerbang UB Soekarno hatta	-7.94984, 112.615411	Mixue Suhat Malang	-7.94637779324508, 112.618050061491

Tabel 4. 15 Arah dan Jarak

No	Heading	Distance to Desstination (km)
1	96,607°	0.28
2	4,124	0.3
3	45,456°	0.069
4	43,014°	0.22
5	-40,579°	0.45

Tabel 4. 16 Hasil Prediksi dan Selisih

No	Predicted Coordinate	Differences
1	-7.97721516624731, 112.636580818575	-1.347×10^{-5}, -5.319×10^{-4}
2	-7.97423446585066, 112.634250919231	-3.845×10^{-5}, -4.188×10^{-5}
3	-7.94936783429258, 112.615817547406	-3.845×10^{-5}, -4.188×10^{-5}

Tabel 4. 17 Hasil Prediksi dan Selisih (lanjutan)

No	Predicted Coordinate	Differences
4	-7.94839333872107, 112.616773784662	-2.304 x 10^{-4}, 1.118 x 10^{-5}
5	-7.9467662955678, 112.618069047894	-3.885 x 10^{-4}, 1.899 x 10^{-5}

Berdasarkan lima percobaan yang telah dilakukan, dapat disimpulkan bahwa rumus yang digunakan memiliki akurasi yang cukup baik dan kesalahan yang relatif kecil. Perbedaan rata-rata antara hasil yang diperoleh dari rumus dan titik koordinat sebenarnya adalah sangat kecil, yaitu sebesar -0,00009966528437. Selain itu, perbedaan maksimum antara hasil yang diperoleh dari rumus dan nilai sebenarnya juga cukup kecil, hanya sekitar 0,0002401616635. Sehingga dapat disimpulkan rumus *Vincenty Formula* tersebut dapat digunakan untuk melakukan kalkulasi titik koordinat suatu objek berdasarkan titik koordinat asal, derajat arah serta jarak terhadap objek.

4.6. Sistem Estimasi Jumlah Buah Kakao

Sistem ini melakukan deteksi buah kakao. Buah kakao yang terdeteksi dalam bentuk bounding box akan dihitung jumlahnya. Setelah terdeteksi maka setiap buah dalam setiap bounding box tersebut diklasifikasi untuk memprediksi buah tersebut telah memasuki usia matang atau belum. Proses pada sistem ini cukup sederhana, pengguna cukup mengunggah citra tanaman kakao. Lalu sistem akan melakukan deteksi buah kakao, serta mengkalkulasi bounding box yang muncul. Sehingga akan ditampilkan kepada pengguna berapa jumlah buah kakao yang ada pada tanaman kakao tersebut. Adapun deteksi ini dilakukan dengan menggunakan confidence threshold sebesar 0.25.

4.6.1. Evaluasi Sistem Estimasi Jumlah Buah Kakao

Pada tahap ini akan dilakukan evaluasi pada 5 model CNN YOLOV8 terbaik yang digunakan untuk melakukan deteksi buah kakao. Selain metriks pelatihan

seperti box loss, dfl loss dan cls loss akan dilakukan pula pengujian manual secara visual. Model akan dijalankan untuk melakukan proses deteksi, lalu hasil deteksi akan dibandingkan dengan hasil penghitungan buah kakao manual secara visual. Adapun berikut 2 citra yang akan digunakan untuk melakukan evaluasi sistem estimasi jumlah buah kakao yaitu gambar 4.41 dan 4.42.

Gambar 4. 41 Citra Tanaman Kakao dengan Buah muda

Dapat dilihat pada gambar 4.40 terdapat 19 buah muda. Buah kakao yang masih muda akan nampak berwarna hijau. Lama-kelamaan akan muncul titik titik kecoklatan pada buah kakao. Beberapa buah kakao pada gambar 4.40 berada pada posisi yang sulit untuk dideteksi seperti buah yang tertutupi oleh buah lainnya, serta buah yang berada dibalik batang. Buah pada posisi yang sulit dijangkau ini akan sulit untuk dideteksi oleh model CNN YOLO.

Gambar 4. 42 Citra Tanaman Kakao dengan Buah Matang

Dapat dilihat pada gambar 4.41 terdapat 20 buah matang. Pada kedua citra diatas yaitu gambar 4.40 dan gambar 4.41 akan dilakukan deteksi menggunakan model YOLOV8n dengan 100 epoch. Hasil deteksi dapat dilihat pada gambar 4.43 dan 4.44.

Gambar 4. 43 Hasil Deteksi Buah Kakao dengan model YOLOV8n 100 epoch

Pada gambar 4.43 terdapat 8 bounding box. Artinya terdapat 8 buah kakao matang yang terdeteksi. Kedelapan bounding box tersebut juga memiliki confidece threshold yang tinggi, yaitu pada nilai diatas 0.9 yang artinya model cukup yakin bahwa prediksinya akurat. Namun masih terdapat 10 buah yang tidak terdeteksi. Buah yang tidak terdeteksi kebanyakan memiliki posisi yang susah untuk dideteksi seperti berada dibalik batang, dibalik buah lainnya, serta ukurannya sangat kecil. Deteksi dilakukan pula pada tanaman kakao dengan buah muda. Hasil deteksi dapat dilihat pada gambar 4.44.

Gambar 4. 44 Hasil Deteksi Buah Kakao dengan model YOLOV8n 100 epoch

Seperti yang dapat dilihat pada gambar 4.44, terdapat 9 bounding box/buah yang berhasil terdeteksi. Namun pada ground truth terdapat 19 buah, sehingga masih terdapat 10 buah. Deteksi dilakukan menggunakan kelima model pada 2 gambar tersebut. Hasil evaluasi pada kelima model dapat dilihat pada tabel 4.15.

Tabel 4. 18 Evaluasi 5 model terbaik pada 2 contoh gambar

model	filename	prediction result	
		num of ripe cocoa	num of unripe cocoa
YOLOV8n 100 epoch	IMG_20230621_160151.jpg		9
	IMG_20230621_160747.jpg	8	

Tabel 4. 19 Evaluasi 5 model terbaik pada 2 contoh gambar (lanjutan)

model	filename	prediction result	
		num of ripe cocoa	num of unripe cocoa
YOLOV8n 500 epoch	IMG_20230621_160151.jpg		12
	IMG_20230621_160747.jpg	10	
YOLOV8m 100 epoch	IMG_20230621_160151.jpg	2	14
	IMG_20230621_160747.jpg	17	
YOLOV8m 700 epoch	IMG_20230621_160151.jpg	3	11
	IMG_20230621_160747.jpg	11	
YOLOV8n 300 epoch	IMG_20230621_160151.jpg		10
	IMG_20230621_160747.jpg	11	

Secara keseluruhan model belum berhasil mendeteksi semua buah kakao yang ada pada tanaman kakao. Hal ini disebabkan karena banyak posisi buah kakao yang tertutupi oleh objek lain seperti batang tanaman kakao, buah kakao, serta ukurannya yang kecil. Hal ini dapat dikembangkan dengan cara melengkapi dataset yang lebih bervariatif seperti menambahkan anotasi bounding box pada buah kakao yang tertutupi. Namun hal tersebut juga perlu diimbangi dengan adanya metode yang dapat melokalisasi hasil deteksi karena apabila sebuah kakao terletak dibalik batang sehingga seolah-olah nampak terdapat dua buah kakao maka menjadi akan bias bagi model.

Apabila ketiga sistem diatas digabungkan, sistem dapat digunakan sebagai alat untuk melakukan monitoring perkebunan kakao dengan metode geotagging. Pada penelitian ini sistem hanya mampu untuk melakukan prediksi titik koordinat

tanaman kakao serta deteksi buah kakao. Sehingga sistem dapat menghasilkan *output* berupa titik koordinat serta jumlah buah kakao. Sistem ini merupakan alternatif murah untuk mengetahui perkembangan dari setiap tanaman kakao dari waktu ke waktu.

BAB IV HASIL DAN PEMBAHASAN

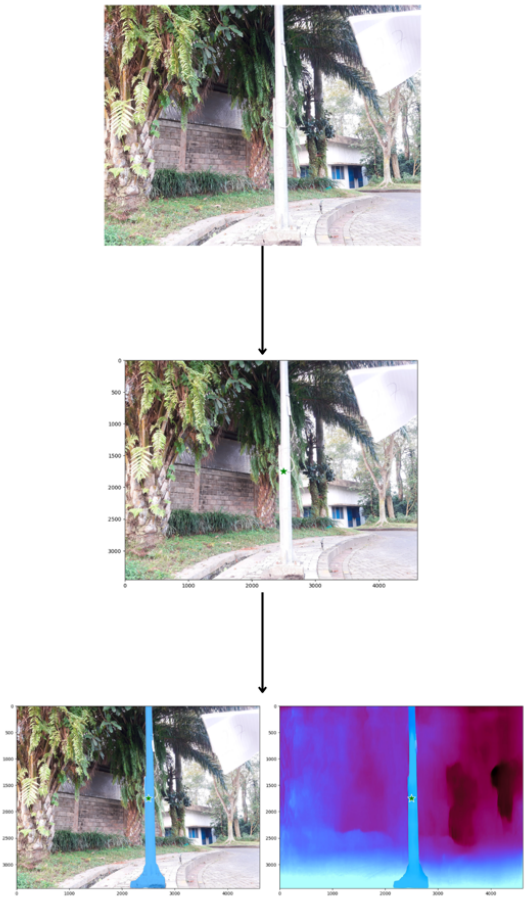
4.1. Dataset

Pada penelitian ini, dilakukan pengumpulan data untuk membangun dataset yang akan digunakan dalam proses pelatihan model *Artificial Neural Network* (ANN) untuk memprediksi jarak serta model *Convolutional Neural Network* (CNN) YOLO untuk mendeteksi buah kakao. Pengumpulan data dilakukan dengan cara mengambil sampel gambar-gambar buah kakao yang sudah matang dan yang belum matang. Selama proses pengumpulan data, gambar-gambar tersebut diolah dan diberikan anotasi. Anotasi ini mencakup informasi mengenai koordinat atau bounding box yang menandai letak buah kakao pada gambar, serta label yang menunjukkan jenis atau kelas buah kakao yang terdeteksi. Setelah pengumpulan data selesai, dataset yang terdiri dari gambar-gambar dan anotasinya akan digunakan dalam proses pelatihan model. Model ANN akan dilatih menggunakan data jarak antara kamera dan buah kakao sebagai input, serta data yang sesuai dengan jarak tersebut sebagai output yang diharapkan. Sementara itu, model CNN YOLO akan dilatih menggunakan gambar-gambar buah kakao beserta anotasinya. Proses pelatihan ini bertujuan untuk mengajarkan model untuk mendeteksi dan mengidentifikasi buah kakao dalam gambar.

4.1.1. Pembuatan Dataset Nilai Rgb dan Jarak Objek

Pembuatan dataset dimulai dengan mencari objek yang menyerupai tiang untuk menggantikan batang tanaman kakao. Setelah objek yang sesuai ditemukan, jarak antara kamera dan objek ditentukan. Setelah jarak kamera dengan objek ditentukan, citra diambil menggunakan kamera atau perangkat lainnya. Pada langkah ini, citra yang telah diambil akan diproses menggunakan model CNN *monodepth estimation*. Model ini akan memperkirakan kedalaman objek berdasarkan nilai RGB pada citra. Setelah estimasi kedalaman dilakukan, ditentukan titik koordinat piksel (x, y) pada area objek yang diestimasi

kedalamannya. Titik koordinat ini akan digunakan dalam proses segmentasi selanjutnya. Pada langkah ini, model CNN *segment anything* digunakan untuk melakukan segmentasi pada citra hasil estimasi kedalaman. Model CNN *segment anything* melakukan proses segmentasi berdasarkan titik koordinat yang telah ditentukan. Titik tersebut menjadi dasar penunjuk objek yang akan disegmentasi areanya. Segmentasi akan mengidentifikasi bagian objek yang menyerupai tiang dan memisahkannya dari latar belakang. Setelah proses segmentasi, nilai RGB pada bagian objek yang diidentifikasi akan diperoleh. Nilai RGB ini merupakan warna piksel pada citra hasil estimasi kedalaman. Setelah mendapatkan kumpulan nilai RGB, nilai median dari kumpulan tersebut diambil. Nilai median memberikan representasi warna tengah dari kumpulan tersebut, yang akan digunakan sebagai representasi nilai RGB untuk jarak objek yang diestimasi. Setelah semua langkah di atas dilakukan untuk setiap objek yang dipilih, data yang terkumpul ditambahkan ke dataset. Setiap data terdiri dari pasangan nilai RGB (nilai median) dan jarak kamera terhadap objek yang sesuai.



Gambar 4.1 Alur Pembuatan Dataset

4.1.2. Profil Dataset Nilai RGB dan Jarak Objek

Pada penelitian ini, terdapat dataset yang berisi pasangan nilai Red,Green,Blue (RGB) terhadap jarak objek dalam satuan meter. Namun, perlu dicatat bahwa nilai RGB yang ada dalam dataset tersebut bukanlah nilai RGB dari citra asli, melainkan nilai RGB yang berasal dari proses estimasi kedalaman menggunakan model CNN *monodepth estimation*. Nilai RGB akan digunakan sebagai input untuk memperkirakan jarak objek. Proses estimasi kedalaman menggunakan model CNN *monodepth estimation*. Dataset ini terdiri dari 112 data, dimana setiap data terdiri dari nilai RGB dan jarak objek yang sesuai.

Tabel 4.1 Dataset RGB dan Jarak Objek

id	r	g	b	distance
0	251.0	138.0	99.0	2.90
1	141.0	41.0	128.0	4.22
2	220.0	72.0	107.0	1.00
3	80.0	18.0	123.0	3.00
4	225.0	76.0	103.0	3.75
...	...	...	...	...
110	222.0	75.0	124.0	2.70
111	252.0	158.0	112.0	1.20
112	156.0	46.0	126.0	2.40

4.1.3. Dataset Buah Kakao

Dataset Buah Kakao dilakukan dengan melakukan pengambilan citra di perkebunan kakao yang terletak di Puslitkoka, Jember. Dilakukan pengambilan citra pada tanaman kakao yang sedang berbuah baik yang buahnya sudah matang maupun yang belum matang. Pengambilan citra juga divariasikan jaraknya yaitu pada rentang jarak sebesar 0,8 hingga 2,8 m sesuai dengan variasi yang terdapat pada dataset nilai rgb dan jarak objek (Bab 4.1.2). Dataset Buah Kakao terdiri dari

total 67 gambar buah kakao dengaan 446 anotasi. Untuk melakukan proses *training* digunakan 46 gambar (sekitar 69% dari total dataset). Selanjutnya, untuk memvalidasi performa model, 13 gambar (sekitar 19% dari total dataset) akan digunakan sebagai *validation set*. Terakhir, 8 gambar (sekitar 12% dari total dataset) akan dijadikan sebagai *test set* untuk menguji akurasi model yang telah dilatih. Dalam setiap gambar, terdapat rata-rata 6 anotasi (label) untuk buah kakao. Dataset ini memiliki dua kelas, yaitu buah kakao matang dan belum matang. Dengan pembagian ini, diharapkan model dapat mempelajari dan mengklasifikasikan gambar-gambar buah kakao dengan akurasi yang tinggi.

Gambar 4.2 Semua Dataset

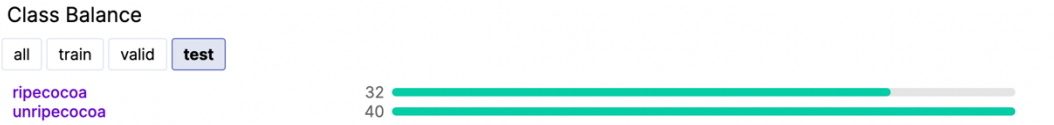
Secara keseluruhan, jumlah anotasi pada kedua kelas, yaitu kelas cocoa matang *(ripe cocoa)* dan kelas cocoa belum matang *(unripe cocoa)*, tergolong cukup seimbang. Terdapat 227 anotasi pada kelas *cocoa* matang dan 217 anotasi pada kelas *cocoa* belum matang. Perbedaan jumlah anotasi antara kedua kelas tersebut tidak terlalu signifikan.

Gambar 4.3 Pembagian Dataset untuk Train

Data pelatihan ini memiliki jumlah sampel yang cukup representatif untuk kedua kelas, yaitu 171 sampel pada kelas *cocoa* matang *(ripe cocoa)* dan 130 sampel pada kelas *cocoa* belum matang *(unripe cocoa)*. Dengan jumlah yang seimbang antara kedua kelas, model pembelajaran mesin dapat mempelajari pola dan informasi yang relevan dari masing-masing kelas. Dengan menggunakan data pelatihan yang representatif ini, model yang dihasilkan memiliki kemampuan yang lebih baik dalam mengklasifikasikan *cocoa* berdasarkan tingkat kematangannya.

Gambar 4.4 Pembagian Dataset untuk Valid

Dataset validasi yang mencakup kedua kelas, yaitu *cocoa* matang *(ripe cocoa)* dengan 24 sampel dan *cocoa* belum matang *(unripe cocoa)* dengan 47 sampel, memungkinkan pengujian kehandalan model dalam mengklasifikasikan tingkat kematangan *cocoa*. Evaluasi menggunakan dataset validasi ini memberikan gambaran yang lebih akurat tentang kemampuan model dalam dunia nyata.



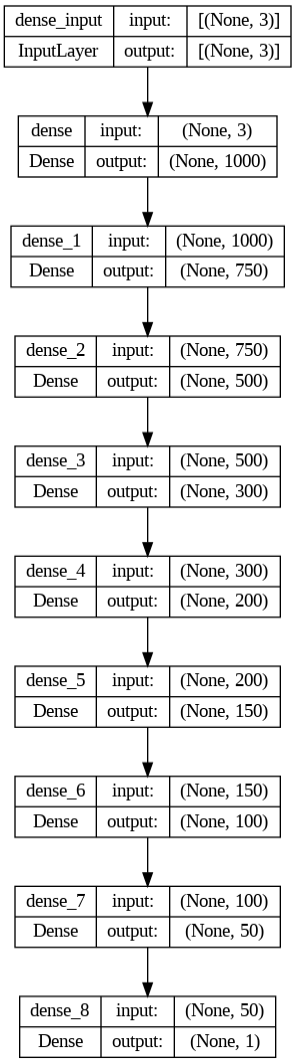
Gambar 4.5 Pembagian Dataset untuk Test

Pada dataset uji *(test set)*, terdapat 32 sampel anotasi pada kelas *cocoa* matang *(ripe cocoa)* dan 40 sampel anotasi pada kelas *cocoa* belum matang *(unripe cocoa)*. Dataset uji ini digunakan untuk menguji performa model yang telah dilatih pada data pelatihan dan divalidasi pada data validasi. Dengan menggunakan dataset uji yang mencakup kedua kelas, model dapat dievaluasi lebih lanjut dalam kemampuannya mengklasifikasikan *cocoa* berdasarkan tingkat kematangannya.

4.2. Eksperimen Model ANN Prediksi Jarak Objek pada Citra

Pada penelitian ini, dilakukan beberapa percobaan pelatihan *Artificial Neural Network* (ANN) untuk mengetahui korelasi antara variabel RGB piksel dengan jarak aktualnya. Percobaan ini menggunakan model CNN *Monodepth Estimation* yang menghasilkan citra RGB dengan nilai yang sangat bervariasi. Percobaan dilakukan dengan menggunakan dua jenis optimizer, yaitu Adamax dan SGD. Optimizer digunakan untuk mengatur proses pembelajaran ANN dengan menyesuaikan bobot dan bias agar mencapai hasil yang optimal. Model akan dilatih dengan sebanyak 112 data.

Selain itu, juga dilakukan variasi pada 3 jumlah epoch (1000, 3000, 5000) dan 3 variasi *batch size* (1, 7, 15). Epoch merupakan iterasi yang dilakukan saat melatih model, sedangkan *batch size* menentukan jumlah sampel yang digunakan dalam satu iterasi. Dengan menggabungkan variasi optimizer, epoch, dan *batch size*, percobaan ini bertujuan untuk mencari kombinasi yang paling baik dalam menghasilkan model ANN yang dapat memprediksi jarak aktual berdasarkan nilai RGB piksel. Sebagai tambahan informasi, pada pengembangan model ANN ini digunakan *Mean Absolute Error* (MAE) sebagai *loss metric* untuk mengetahui akurasi model yang telah dilatih. Sebagai acuan, pada penelitian ini telah disepakati untuk mengembangkan model ANN yang memiliki nilai MAE dibawah 0.3. Adapun arsitektur ANN yang akan digunakan dapat dilihat pada gambar 4.6.

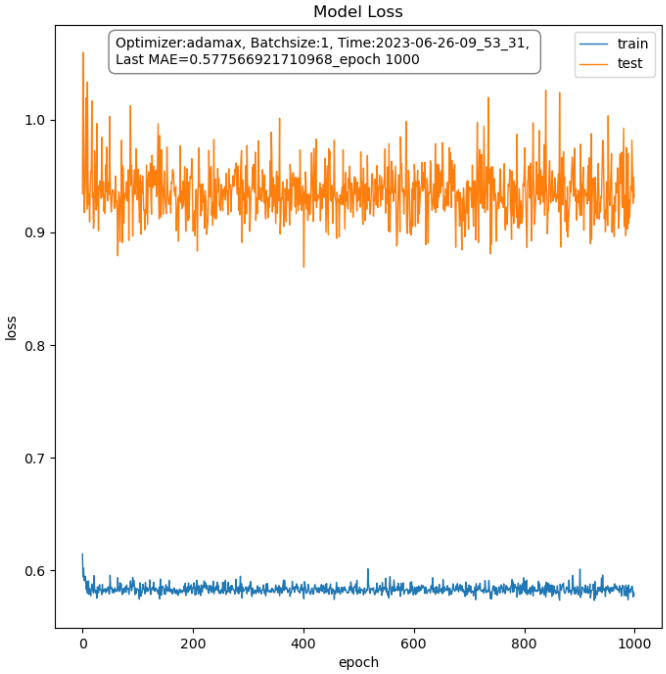


Gambar 4.6 Arsitektur Model ANN

Dari kombinasi epoch, *batch size* serta *optimizer* dihasilkan 18 model ANN berikut.

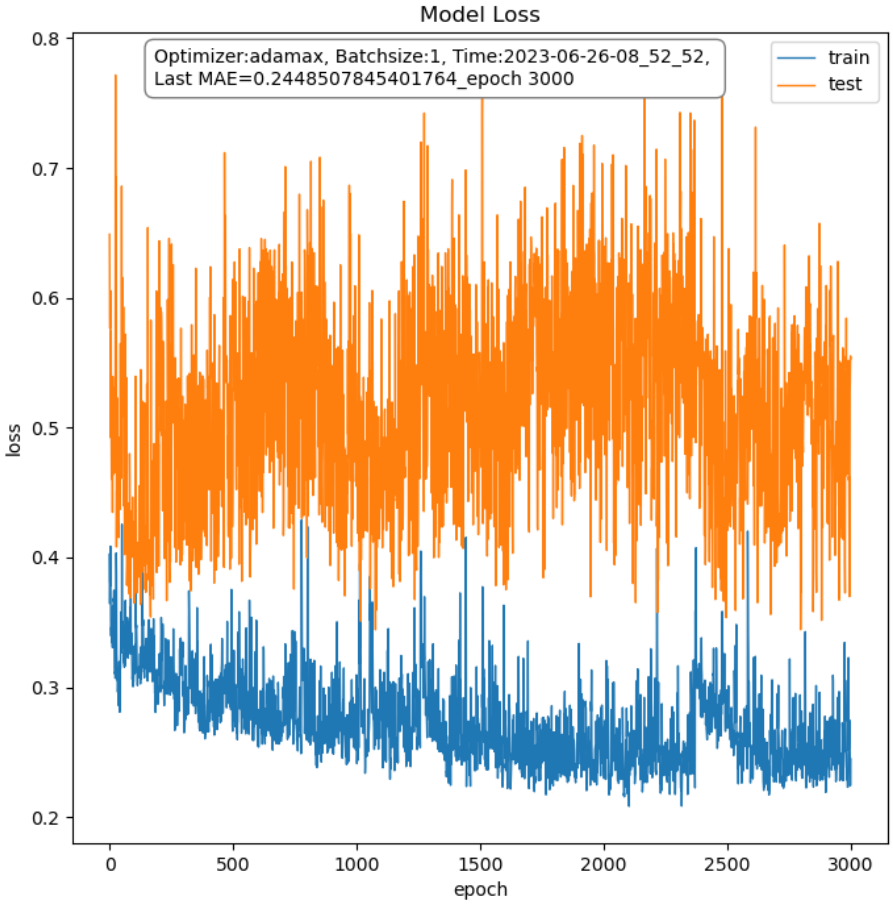
4.2.1. Model ANN dengan Optimizer Adamax dan *Batch Size* 1

Pada model ANN ini, digunakan optimizer Adamax dengan parameter *batch size* sebesar 1. Dilakukan training sampai dengan epoch ke 1000. Setelah melalui pelatihan awal yang berlangsung hingga mencapai 1000 epoch, ditemukan bahwa nilai *Mean Absolute Error* (MAE) model sebesar 0.577566921710968. Data histori pelatihan model dapat dilihat pada gambar 4.6.



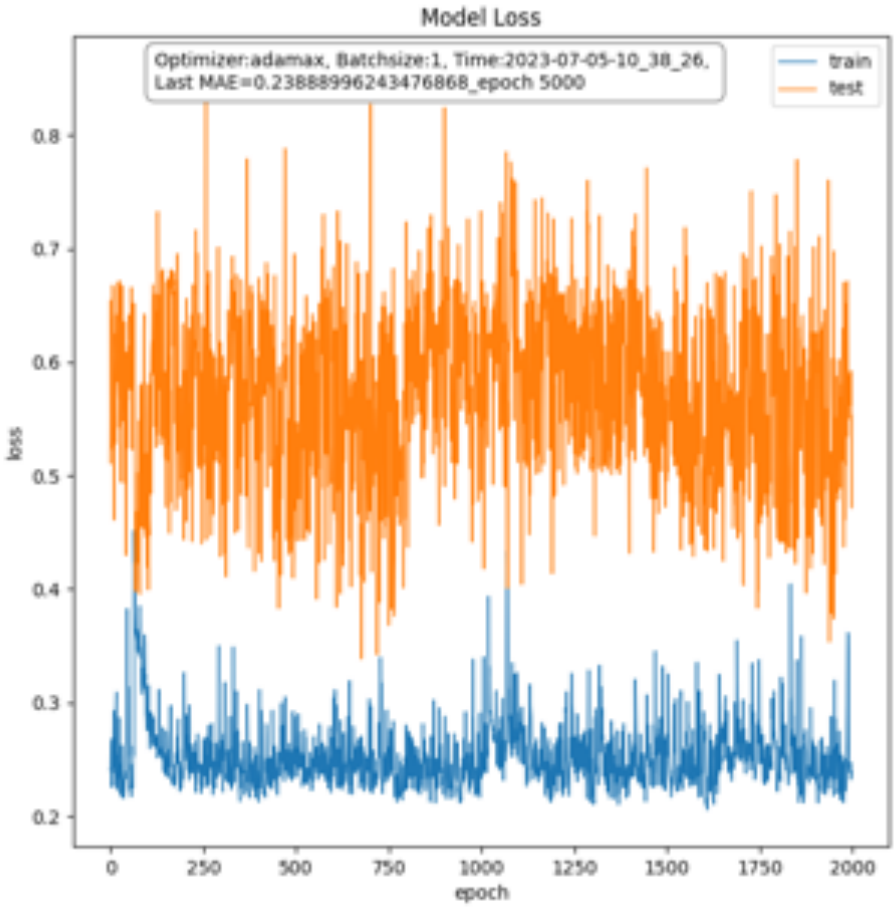
Gambar 4.7 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 1 pada epoch 1000

Meskipun demikian, belum terlihat perbaikan yang signifikan pada nilai loss baik pada data latih maupun data uji. Nilai loss *test* tidak menurun secara signifikan dan terus berubah pada kisaran 1 hingga 0,9. Sedangkan nilai loss train mengalami penurunan sedikit dari 0,6 menjadi 0,577. Meskipun penurunannya tidak signifikan, terdapat perbaikan nilai yang terjadi. Namun nilai MAE 0,577 masih terlalu besar dari acuan awal. Proses training dilanjutkan pada epoch 1000 hingga 3000. Pada gambar 4.7 akan disajikan hasil perbaikan nilai loss train dan *test* pada epoch 3000.



Gambar 4.8 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 1 pada epoch 3000

Setelah melatih model ini hingga epoch 3000, terlihat indikasi bahwa model mengalami overfitting karena terdapat perbedaan yang signifikan antara nilai loss pada data uji *(test)* dan data latih *(train)*. Namun, meskipun demikian, model ini berhasil mencapai *Mean Absolute Error* (MAE) yang cukup kecil dan sesuai dengan target yang diinginkan, yaitu 0,24485. Selanjutnya pelatihan model dilanjutkan pada epoch 3000 hingga 5000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.8.

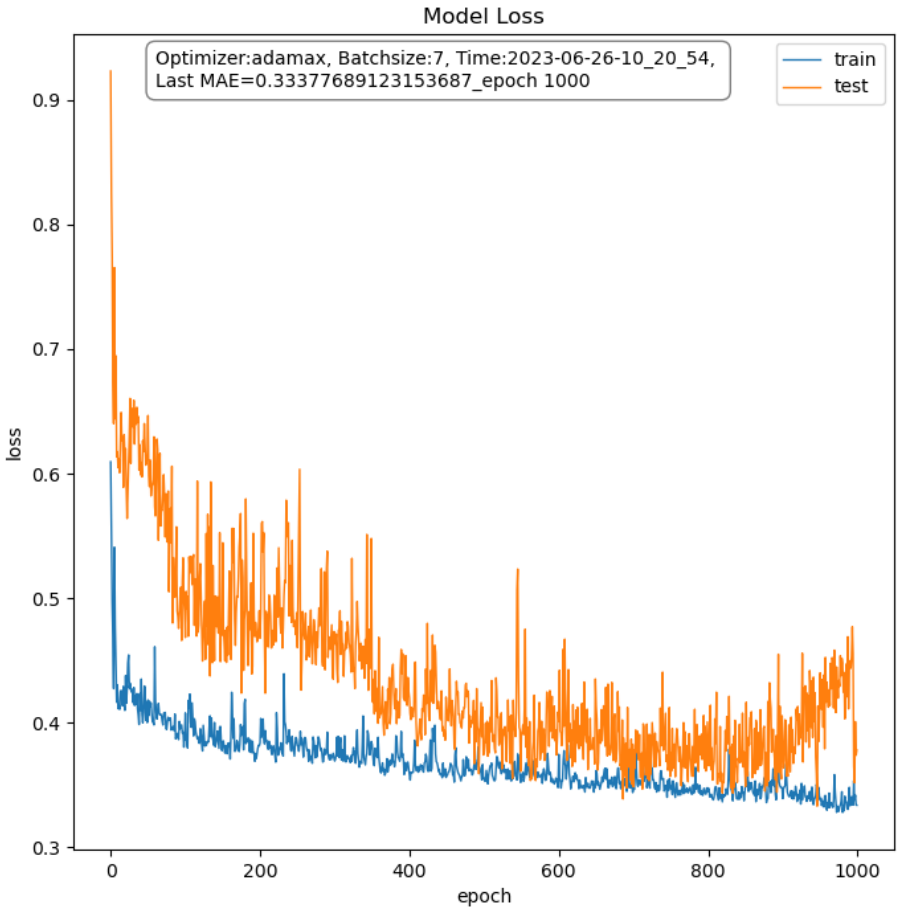


Gambar 4.9 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 1 pada epoch 5000

Setelah dilakukan pelatihan ulang hingga epoch 5000, model ini terindikasi mengalami *overfitting* karena terdapat perbedaan yang signifikan antara nilai loss pada data *test* dan data train. Tidak terlihat adanya perbaikan yang signifikan pada nilai loss juga. Nilai loss pada data *test* bervariasi antara 0,4 hingga 0,8, sedangkan nilai loss pada data train berkisar antara 0,2 hingga 0,4. Meskipun demikian, model ini berhasil mencapai *Mean Absolute Error* (MAE) yang cukup kecil dan sesuai dengan target yang diinginkan, yaitu sebesar 0,2388.

4.2.2. Model ANN dengan Optimizer Adamax dan *Batch Size* 7

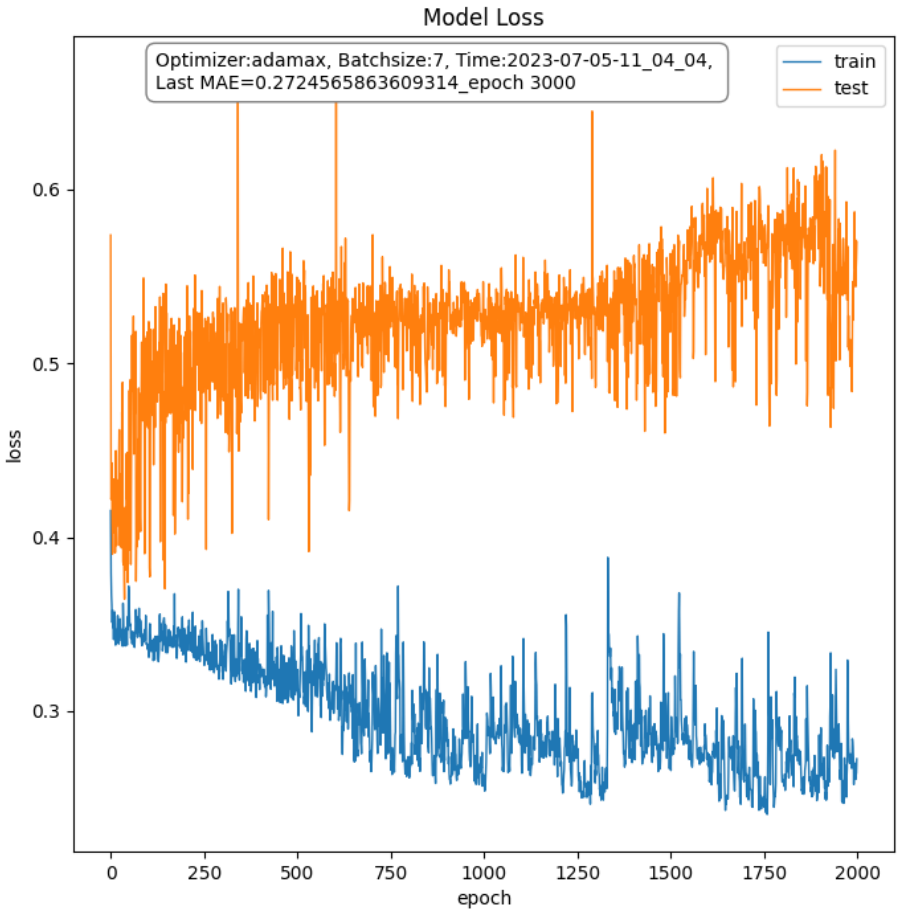
Eksperimen dilanjutkan dengan melakukan pelatihan model menggunakan optimizer Adamax dan menggunakan *batch size* sebesar 7. Pelatihan model dilakukan hingga epoch 1000. Hasil dari pelatihan tersebut dapat dilihat pada gambar 4.9.



Gambar 4.10 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 7 pada epoch 1000

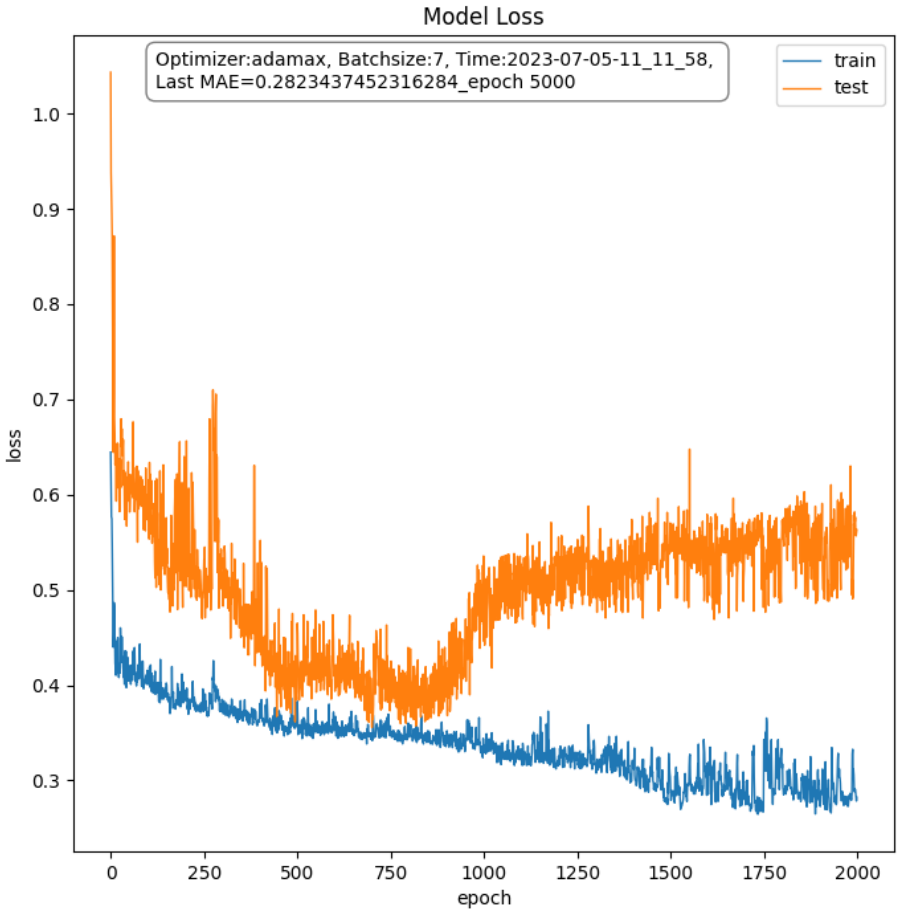
Pada grafik terlihat terjadi penurunan nilai loss yang signifikan pada proses train dan *test*. Hal ini mengindikasikan adanya perbaikan akurasi model. Nilai loss *test* awalnya berkisar pada 0,7, namun mengalami penurunan yang signifikan hingga mencapai kisaran nilai 0,35 pada epoch 800. Sementara itu, nilai loss train juga mengalami penurunan yang signifikan, dimulai dari kisaran 0,4 dan mencapai kisaran 0,35 pada epoch 1000. Meskipun nilai tersebut masih lebih tinggi dibandingkan dengan acuan yang ingin dicapai, pada proses pelatihan ini model

berhasil memperbaiki loss dengan baik. Pelatihan pada model ini dilanjutkan hingga epoch 3000. Hasil dari pelatihan tersebut akan disajikan pada gambar 4.10.



Gambar 4.11 Grafik Loss Model ANN dengan optimizer adamax dan batch size 7 pada epoch 3000

Pada grafik terlihat terjadi perbedaan antara tren grafik *train* dan *test.* Grafik *test* cenderung meningkat sedangkan grafik train cenderung menurun. Nilai *test* naik dari kisaran 0,4 hingga 0,55, sementara nilai *train* menurun dari kisaran 0,35 hingga 0,2. Hal ini merupakan indikasi yang kuat terjadinya *overfitting*. Namun, model ini berhasil mencatatkan nilai MAE yang cukup baik, yaitu sebesar 0,27245. Nilai MAE tersebut lebih kecil dibandingkan dengan nilai MAE acuan yang ingin dicapai. Pelatihan kembali dilanjutkan hingga epoch 5000. Hasil pelatihan tersebut akan disajikan pada gambar 4.11.

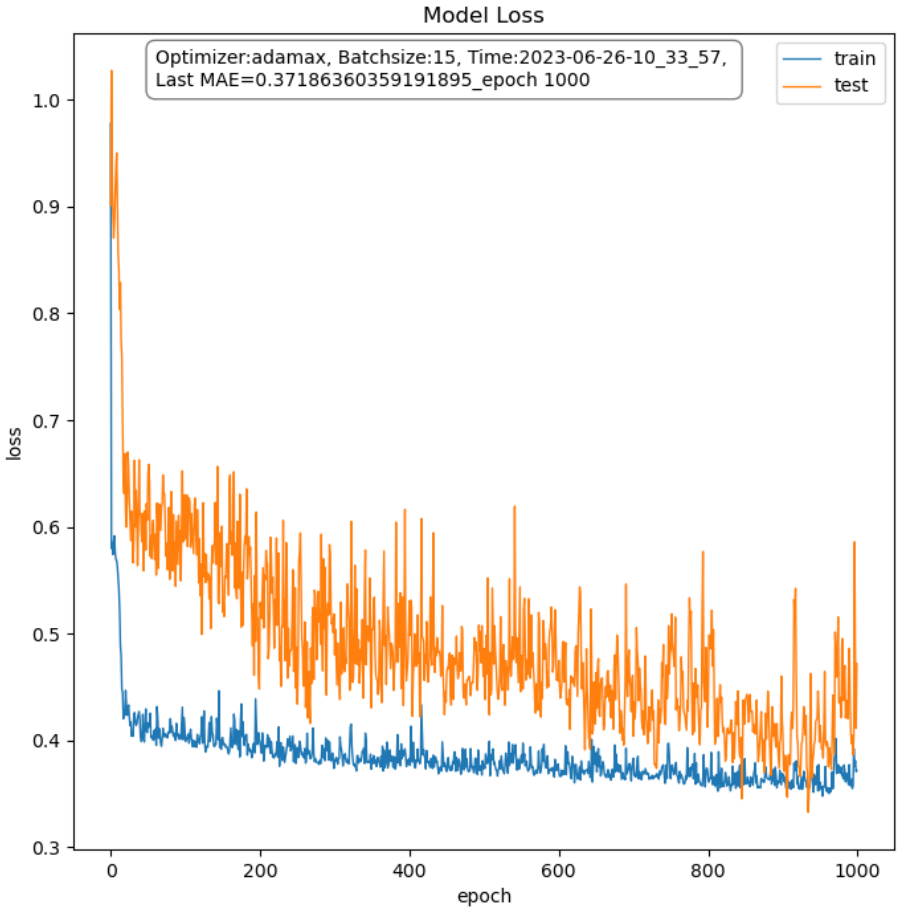


Gambar 4.12 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 7 pada epoch 5000

Dapat dilihat pada grafik di atas, terjadi perbaikan model yang signifikan antara epoch 3000 hingga 3800. Nilai *test* dan train secara bersamaan mengalami penurunan. Namun, setelah epoch 3800, nilai train terus menurun sementara nilai *test* kembali meningkat dan berkisar pada 0,5. Model ini berhasil mencapai MAE yang sesuai dengan acuan awal, yaitu sebesar 0,28234.

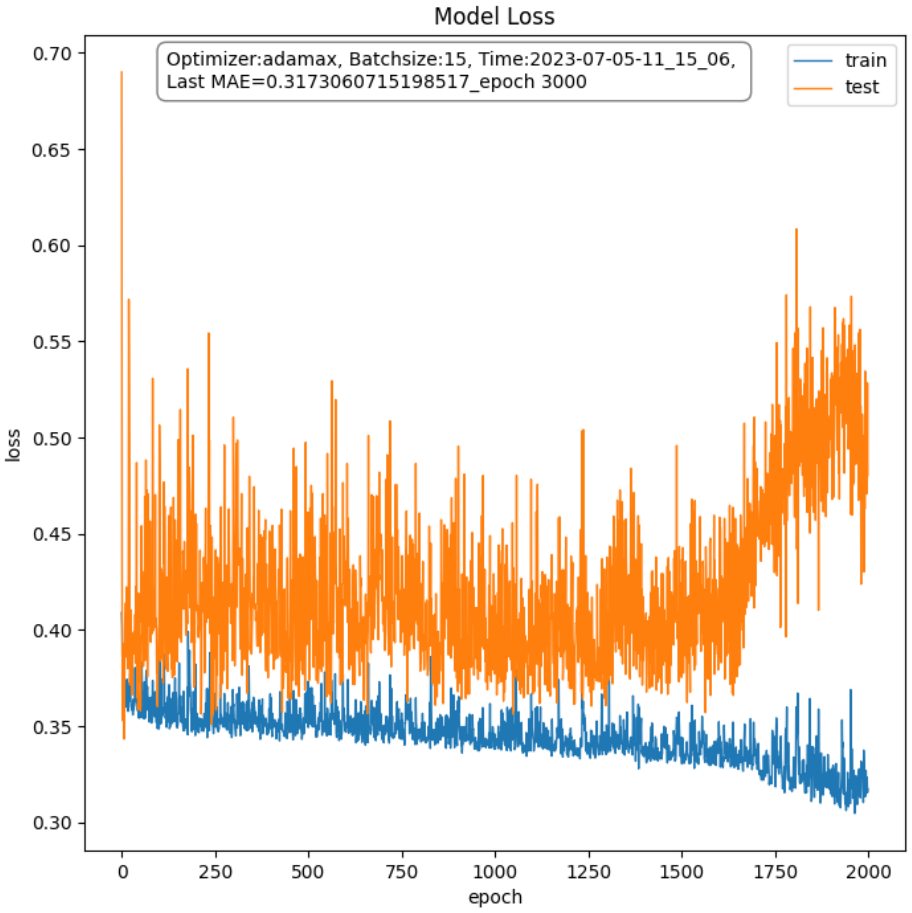
4.2.3. Model ANN dengan Optimizer Adamax dan *Batch Size* 15

Pada model *Artificial Neural Network* (ANN) ini, digunakan optimizer Adamax dengan parameter *batch size* sebesar 15. Dilakukan pelatihan model hingga mencapai epoch ke-1000. Dalam proses tersebut, dilakukan optimisasi menggunakan optimizer Adamax. Pelatihan dimulai hingga epoch 1000, hasil pelatihan tersebut akan disajikan pada gambar 4.12.



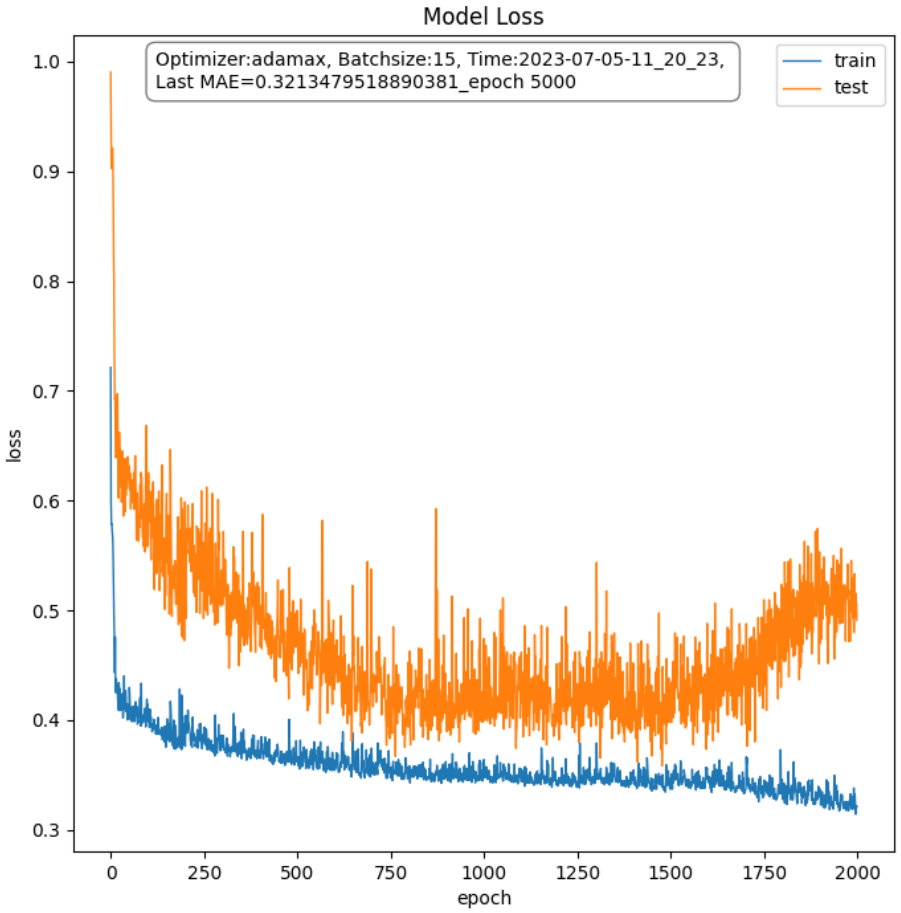
Gambar 4.13 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 15 pada epoch 1000

Pelatihan awal ini berhasil memperbaiki model dengan signifikan, terlihat dari penurunan nilai loss baik pada data train maupun data *test*. Pada epoch 1000, model ini mencapai nilai MAE sebesar 0,37186. Meskipun nilai MAE tersebut masih lebih besar dari acuan awal yang ditetapkan, tidak terlihat indikasi overfitting maupun underfitting pada model tersebut. Hal ini menunjukkan bahwa model tersebut dapat secara baik menyesuaikan diri dengan data pelatihan tanpa kehilangan kemampuan umum untuk memprediksi data baru. Pelatihan model ini dilanjutkan hingga epoch 3000. Hasil pelatihan tersebut akan disajikan pada gambar 4.13.



Gambar 4.14 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 15 pada epoch 3000

Pada pelatihan ini, terdapat indikasi bahwa model mengalami overfitting. Terlihat adanya penyimpangan pada sekitar epoch 1600, di mana nilai *test* cenderung meningkat sedangkan nilai train cenderung menurun. Model ini mencatatkan nilai MAE sebesar 0,317, yang masih lebih besar dari acuan awal yang ditetapkan. Selanjutnya, model akan dilatih kembali hingga epoch 5000, dan hasil pelatihan tersebut akan ditampilkan pada Gambar 4.14.



Gambar 4.15 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 15 pada epoch 5000

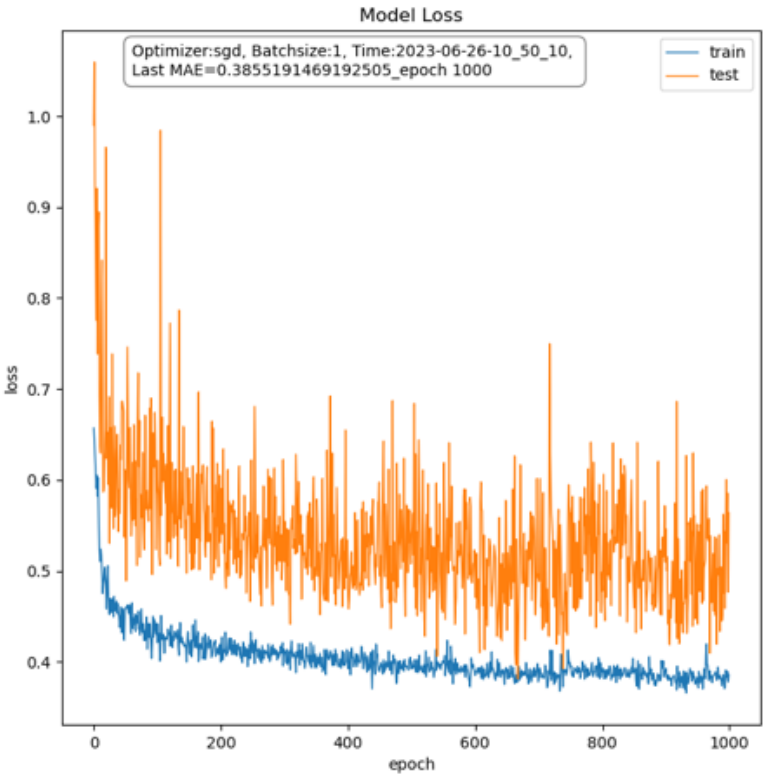
Pada pelatihan ini, terjadi penurunan yang cukup signifikan pada epoch 3000 hingga 3750. Nilai *test* dan train secara bersamaan mengalami perbaikan nilai. Namun, pada epoch 3750 hingga 4500, tidak terjadi perbaikan nilai yang signifikan. Kemudian, pada epoch 4500 hingga 5000, terdapat indikasi overfitting di mana nilai *test* cenderung meningkat sedangkan nilai train cenderung menurun.

Berdasarkan eksperimen yang telah dilakukan, optimizer Adamax berhasil menghasilkan perbaikan nilai yang signifikan pada epoch 0 hingga 1000. Pada beberapa kasus lainnya, seperti pada epoch 3000 hingga 4000, juga terjadi perbaikan nilai yang signifikan. Namun, pada epoch 1000 hingga 3000 dan 4000 hingga 5000, sering terjadi overfitting pada model. Keterbatasan jumlah data juga menjadi faktor yang mempersulit perbaikan model pada epoch di atas 1000. Oleh karena itu, melalui eksperimen berbagai kombinasi pelatihan model, akan dicari

model yang memiliki nilai MAE terendah. Terdapat 2 model dengan nilai MAE terendah yakni model yang dilatih menggunakan optimizer adamax dengan *batch size* 1 pada epoch 3000 dan 5000. Model pada epoch 3000 memiliki nilai MAE sebesar 0.2448507845401764 sedangkan model pada epoch 5000 memiliki nilai MAE sebesar 0.23888996243476868. Namun kedua model tersebut terindikasi mengalami overfitting. Adapun model lainnya yang memiliki nilai loss mae terendah namun tidak mengalami indikasi overfitting yakni model dengan optimizer adamax dan batchsize sebesar 7 pada epoch 1000. Model tersebut memiliki nilai mae sebesar 0.333776.

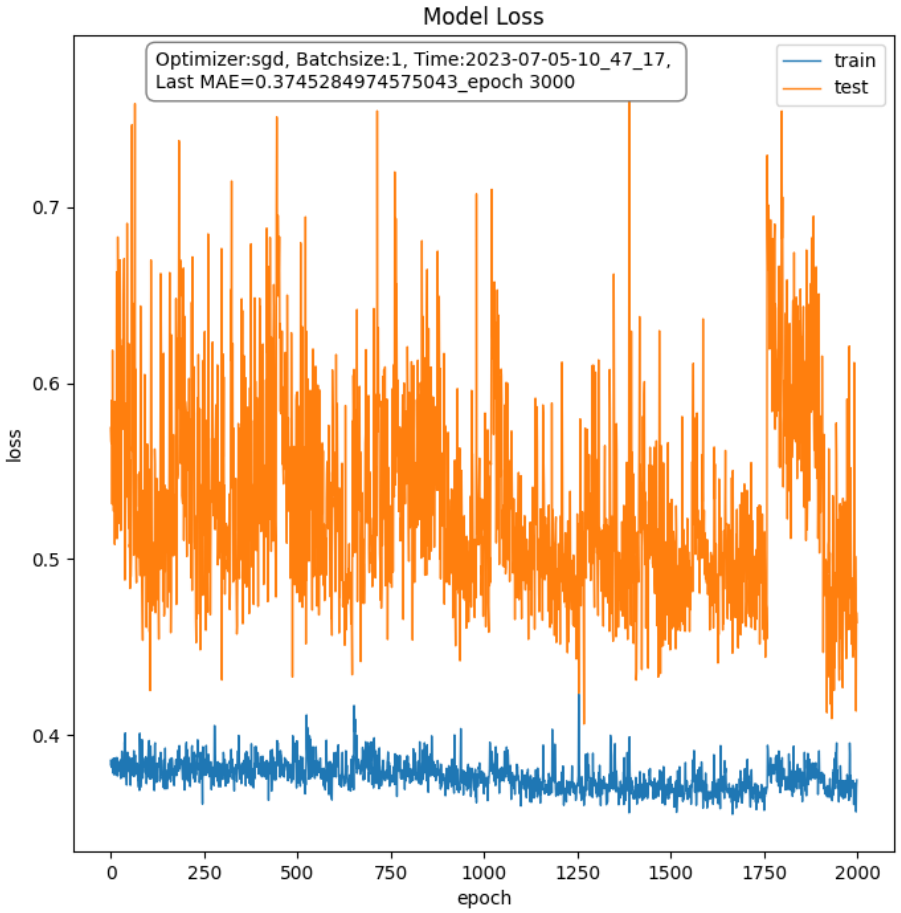
4.2.4. Model ANN dengan Optimizer SGD dan *Batch Size* 1

Pada tahap ini, model akan dilatih menggunakan optimizer SGD dengan *batch size* sebesar 1. Perkembangan model pada epoch ke-1000, ke-3000, dan ke-5000 akan dievaluasi. Nilai MAE akan menjadi parameter apakah model sudah memiliki performa yang cukup baik atau belum. Selain itu, akan dilihat pula apakah ada indikasi terjadinya overfitting atau underfitting pada model.



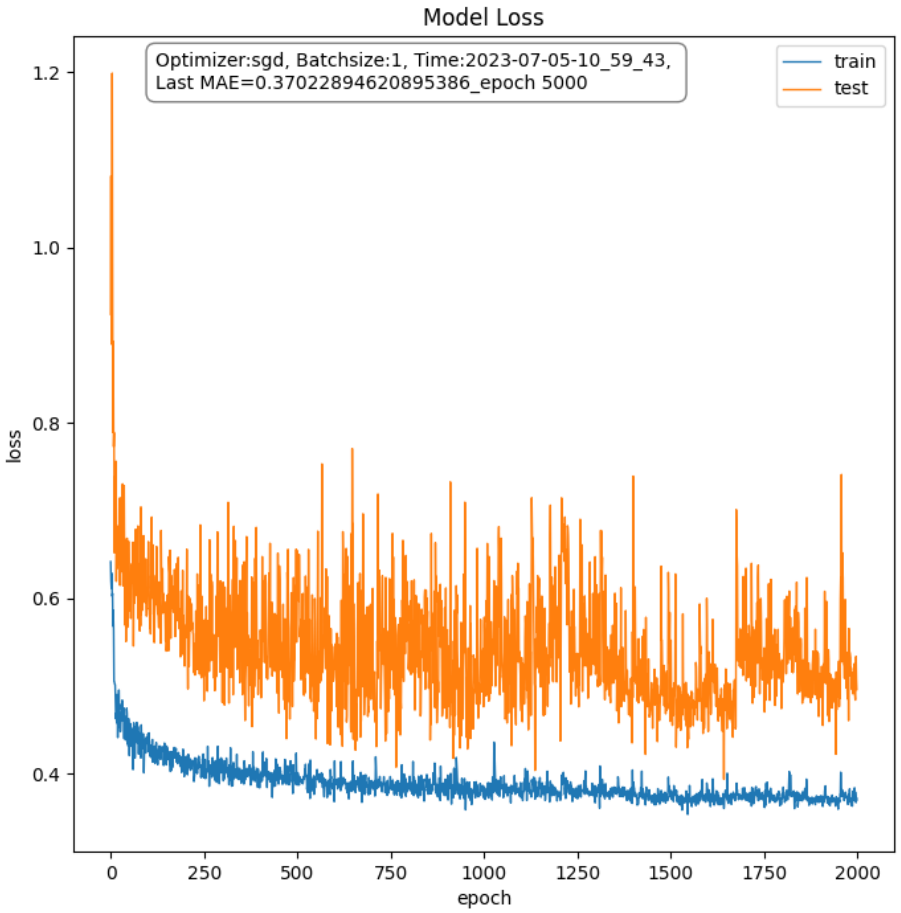
Gambar 4.16 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 1 pada epoch 1000

Pelatihan dilakukan hingga mencapai epoch 1000. Terlihat pada Gambar 4.15, terjadi penurunan signifikan pada nilai train dan *test*. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.385, yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Pelatihan model kembali dilanjutkan hinnga epoch 3000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.16.



Gambar 4.17 Grafik Loss Model ANN dengan optimizer adamax dan *batch size* 1 pada epoch 3000

Seperti yang terlihat pada Gambar 4.16, tidak terjadi perbaikan yang signifikan pada model. Nilai loss MAE pada train maupun *test* tidak mengalami perubahan yang signifikan. Nilai loss MAE pada *test* berfluktuasi antara 0.45 hingga 0.7, sementara nilai loss MAE pada train hanya berkisar antara 0.3 hingga 0.4. Model mencatatkan nilai MAE sebesar 0.37452, dimana nilai tersebut yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Pelatihan model kembali dilanjutkan hinnga epoch 5000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.17.



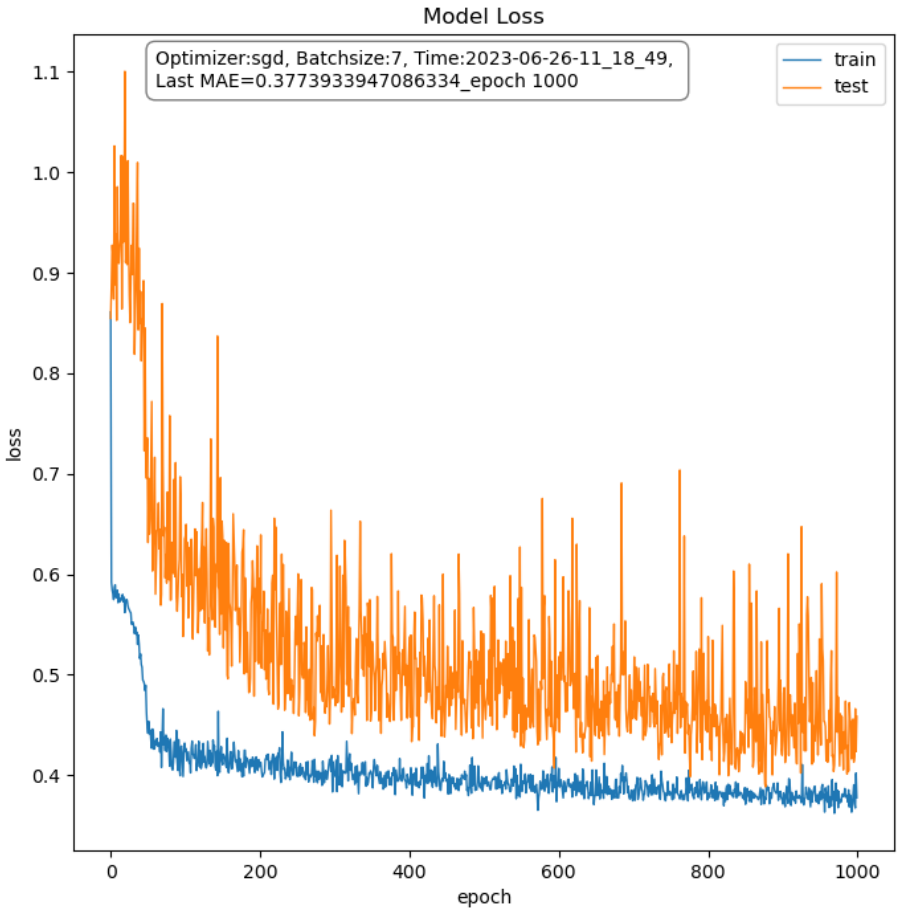
Gambar 4.18 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 1 pada epoch 5000

Seperti yang terlihat pada Gambar 4.17, tidak terjadi perbaikan yang signifikan pada model. Namun nilai loss pada train mengalami penurunan sedikit demi sedikit. Nilai loss MAE pada *test* berkisar antara 0.45 hingga 0.7, sementara nilai loss MAE pada train hanya berkisar pada nilai 0.4. Model mencatatkan nilai MAE sebesar 0.3702, dimana nilai tersebut yang masih sedikit jauh dari acuan awal yang telah ditetapkan.

4.2.5. Model ANN dengan Optimizer SGD dan *Batch Size* 7

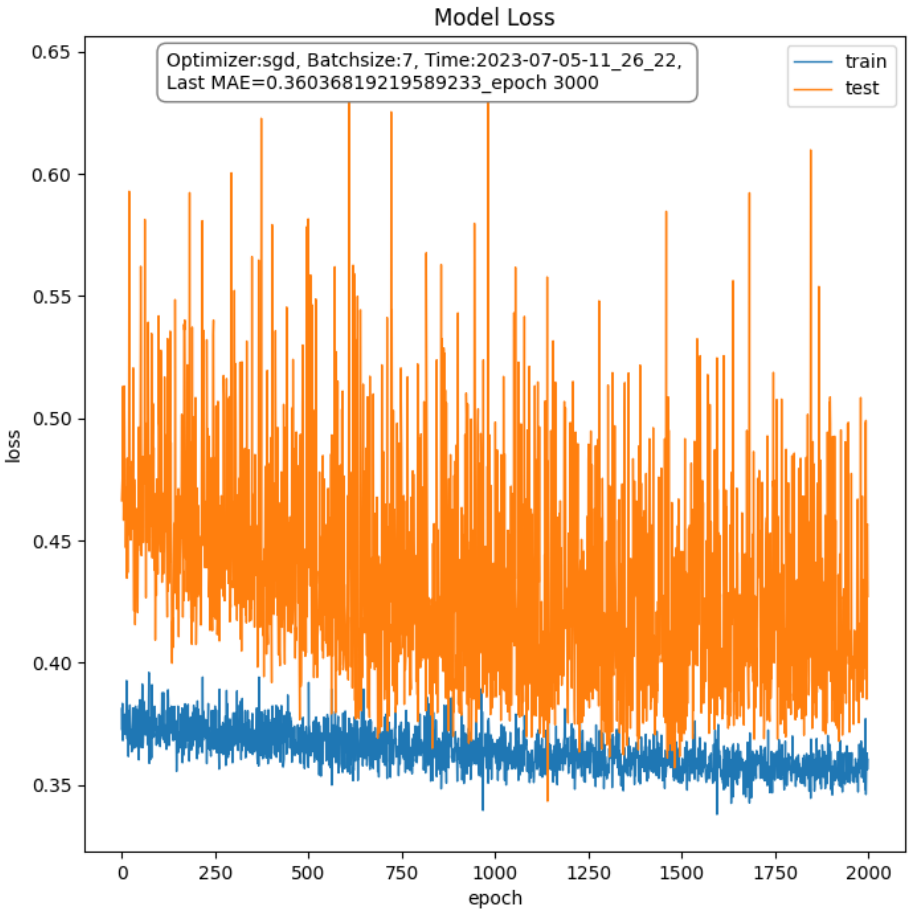
Pada tahap ini, model akan dilatih menggunakan optimizer SGD dengan *batch size* sebesar 7. Perkembangan model pada epoch ke-1000, ke-3000, dan ke-5000 akan dievaluasi. Nilai MAE akan menjadi parameter apakah model sudah memiliki

performa yang cukup baik atau belum. Selain itu, akan dilihat pula apakah ada indikasi terjadinya overfitting atau underfitting pada model.



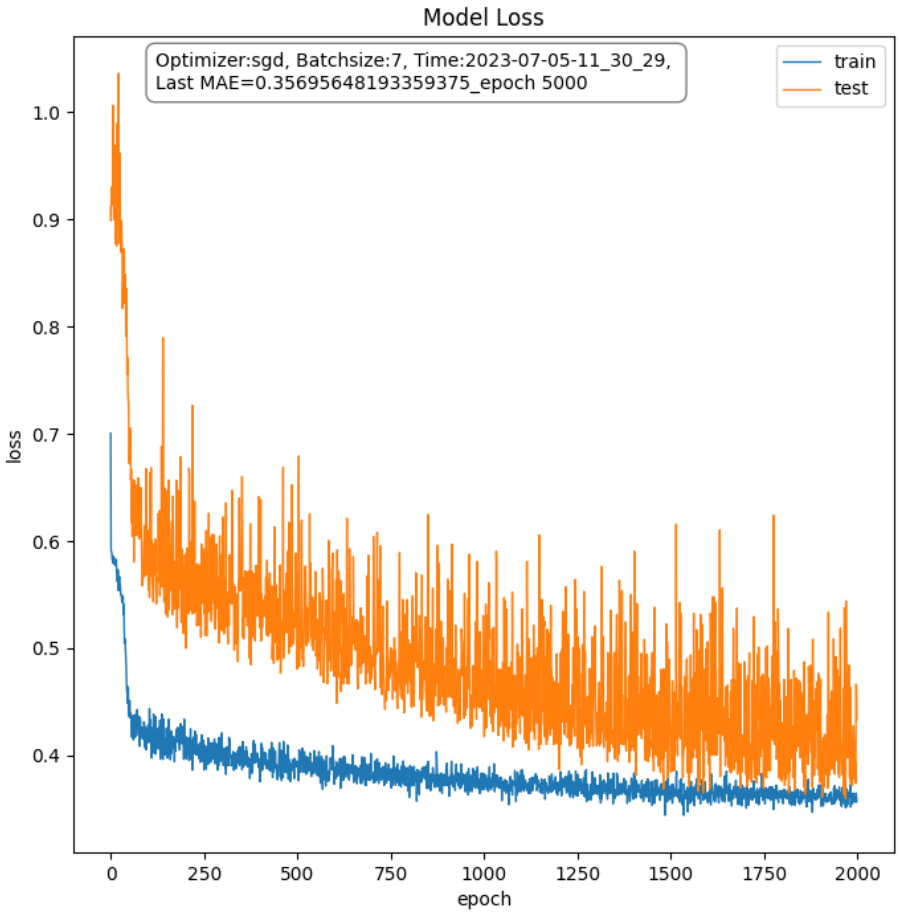
Gambar 4.19 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 7 pada epoch 1000

Pelatihan dilakukan hingga mencapai epoch 1000. Terlihat pada Gambar 4.18, terjadi penurunan signifikan pada nilai train dan *test*. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.3773933, yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Pelatihan model kembali dilanjutkan hinnga epoch 3000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.19.



Gambar 4.20 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 7 pada epoch 3000

Seperti yang terlihat pada Gambar 4.19. tidak terjadi perbaikan yang signifikan pada model. Nilai loss MAE pada train maupun *test* tidak mengalami perubahan yang signifikan. Nilai loss MAE pada *test* berfluktuasi antara 0.4 hingga 0.6, sementara nilai loss MAE pada train hanya berkisar antara 0.35 hingga 0.4. Model mencatatkan nilai MAE sebesar 0.360368, dimana nilai tersebut yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Tidak adanya perbaikan yang signifikan merupakan indikasi terjadinya overfitting pada model ini. Pelatihan model kembali dilanjutkan hinnga epoch 5000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.20.

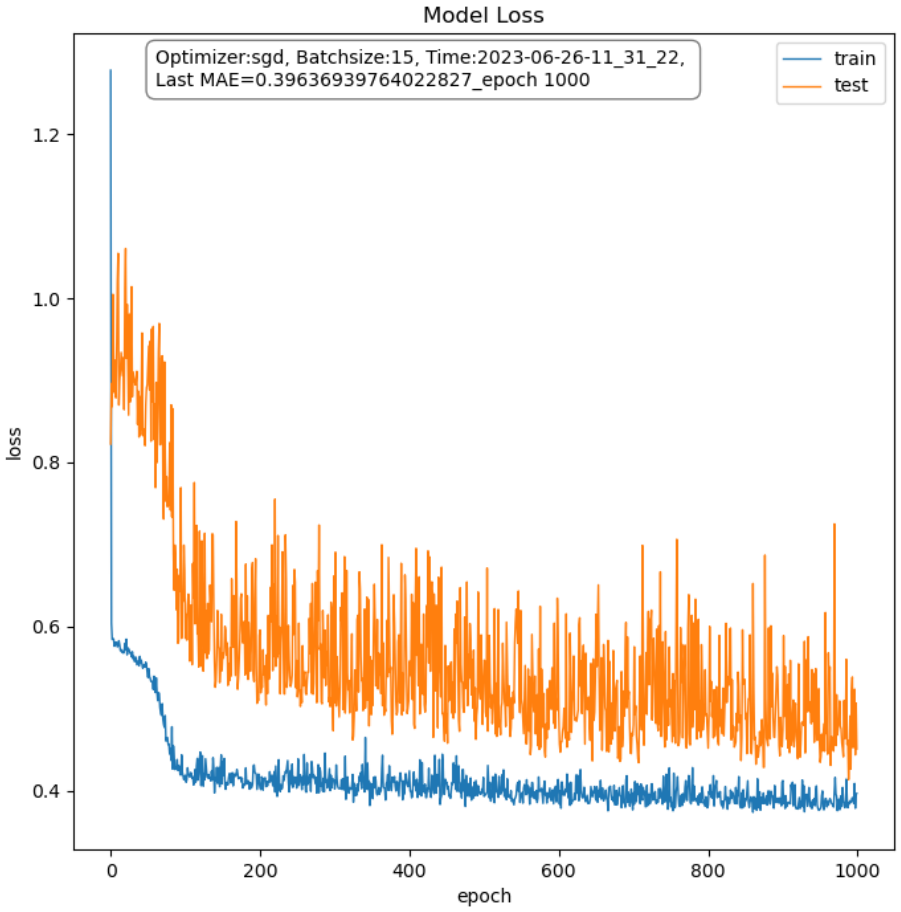


Gambar 4.21 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 7 pada epoch 5000

Pelatihan dilakukan hingga mencapai epoch 1000. Terlihat pada Gambar 4.20, terjadi penurunan signifikan pada nilai *train* dan *test*. Nilai *test* yang awalnya berkisar pada 0.6 mengalami penurunan hingga mencapai nilai 0.4. Sedangkan nilai train mengalami penurunan sedikit demi sedikit pada kisaran nilai 0.4. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.3569, yang masih sedikit jauh dari acuan awal yang telah ditetapkan.

4.2.6. Model ANN dengan Optimizer SGD dan *Batch Size* 15

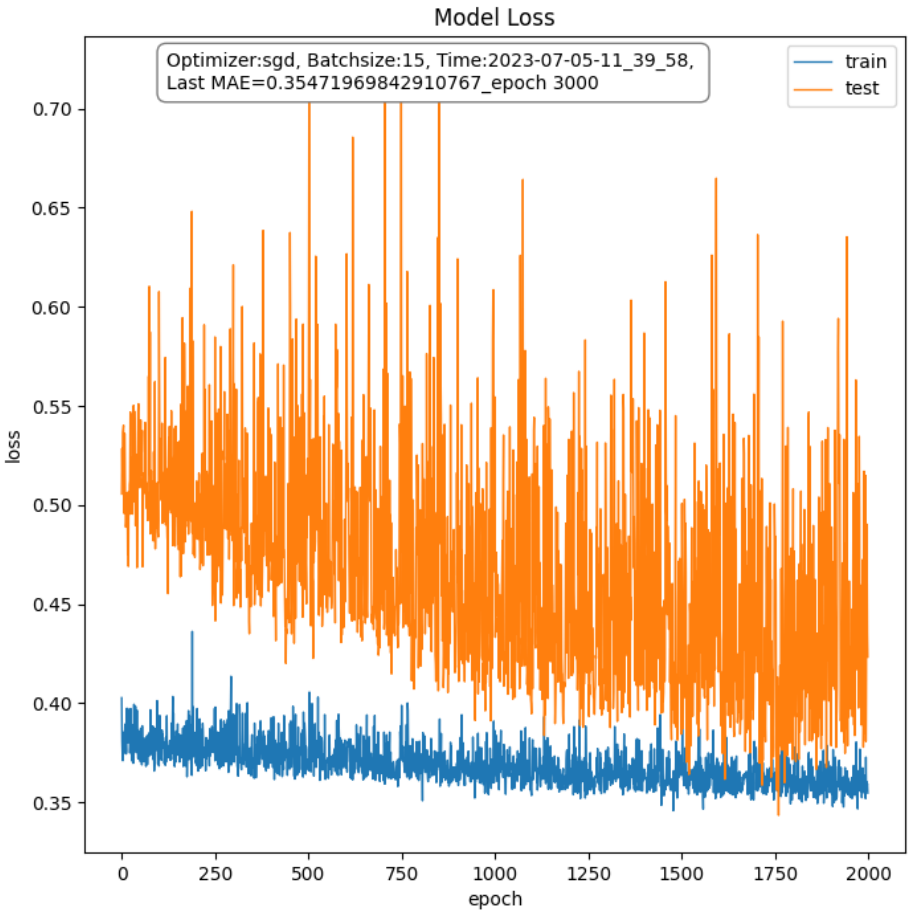
Pada tahap ini, model akan dilatih menggunakan optimizer SGD dengan *batch size* sebesar 15. Perkembangan model pada epoch ke-1000, ke-3000, dan ke-5000 akan dievaluasi. Nilai MAE akan menjadi parameter apakah model sudah memiliki performa yang cukup baik atau belum. Selain itu, akan dilihat pula apakah ada indikasi terjadinya overfitting atau underfitting pada model.



Gambar 4.22 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 15 pada epoch 1000

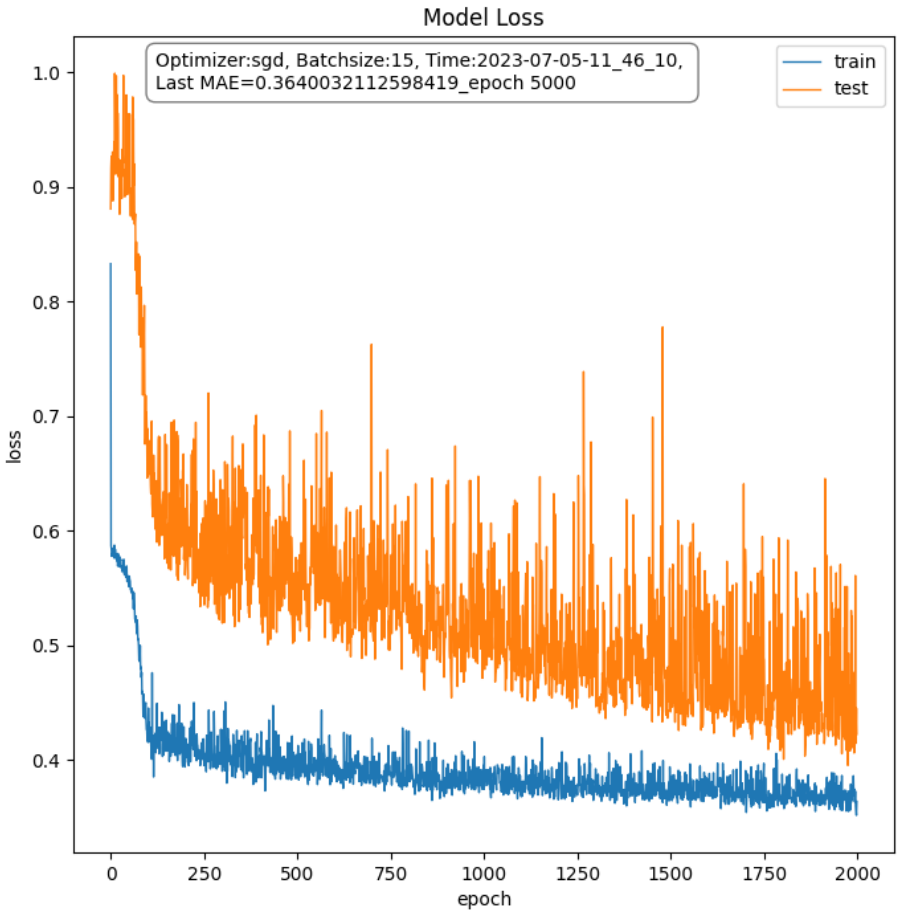
Pelatihan dilakukan hingga mencapai epoch 1000. Terlihat pada Gambar 4.21, nilai loss mae *test* mengalami penurunan yang awalnya berkisar pada nilai 0.6 hingga pada epoch 1000 berkisar pada nilai 0.5. Sedangkan nilai loss mae train tidak mengalami penurunan yang signifikan. Pada epoch 200 hingga epoch 1000 tetap berkisar pada 0.4. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.396369, yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Pelatihan

model kembali dilanjutkan hingga epoch 3000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.22.



Gambar 4.23 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 15 pada epoch 3000

Seperti yang terlihat pada Gambar 4.22. Tidak terjadi perbaikan yang signifikan pada model. Nilai loss MAE pada train maupun *test* tidak mengalami perubahan yang signifikan. Nilai loss MAE pada *test* berfluktuasi antara 0.4 hingga 0.6, sementara nilai loss MAE pada train hanya berkisar antara 0.35 hingga 0.4. Model mencatatkan nilai MAE sebesar 0.360368, dimana nilai tersebut yang masih sedikit jauh dari acuan awal yang telah ditetapkan. Tidak adanya perbaikan yang signifikan merupakan indikasi terjadinya overfitting pada model ini. Pelatihan model kembali dilanjutkan hinnga epoch 5000. Hasil pelatihan tersebut dapat dilihat pada gambar 4.23.



Gambar 4.24 Grafik Loss Model ANN dengan optimizer sgd dan *batch size* 15 pada epoch 5000

Pelatihan dilakukan hingga mencapai epoch 5000. Terlihat pada Gambar 4.23, terjadi penurunan signifikan pada nilai train dan *test*. Nilai *test* yang awalnya berkisar pada 0.6 mengalami penurunan hingga mencapai nilai 0.45. Sedangkan nilai train mengalami penurunan sedikit demi sedikit pada kisaran nilai 0.4. Pada evaluasi akhir, model ini memiliki akurasi MAE sebesar 0.36400321, dimana nilai tersebut sedikit jauh dari acuan awal yang telah ditetapkan.

Setelah melakukan beberapa eksperimen kombinasi parameter diatas maka didapatkan 2 model yang memiliki nilai mae terkecil yaitu sebesar 0.35695648193359375 pada model yang dilatih menggunakan *batch size* sebesar 7 dan dilatih hingga epoch 5000. Lalu ada pula model yang memiliki nilai mae sebesar 0.35471969842910767 yaitu model yang dilatih menggunakan *batch size* sebesar 15 dan dilatih hingga epoch 3000.

4.2.7. Evaluasi model ANN Prediksi Jarak berdasarkan Citra RGB

Setelah bereksperimen dengan berbagai kombinasi parameter training, maka disimpulkan untuk menggunakan model ANN dengan optimizer adamax, *batch size* 7 pada epoch 1000. Model tersebut memiliki nilai mae 0.333776. Model tersebut digunakan karena tidak mengalami indikasi overfitting maupun underfitting. Sehingga diharapkan model tersebut dapat melakukan prediksi secara stabil. Hasil seluruh eksperimen model ANN dapat dilihat pada tabel 4.2.

Tabel 4.2 Perbandingan Model ANN

Model	Batch Size	Epoch	Last Train MAE
adamax	1	1000	0.5775669217
adamax	1	3000	0.2448507845
adamax	1	5000	0.2388899624
adamax	7	1000	0.3337768912
adamax	7	3000	0.2724565864
adamax	7	5000	0.2823437452
adamax	15	1000	0.3718636036
adamax	15	3000	0.3173060715
adamax	15	5000	0.3213479519
sgd	1	1000	0.3855191469
sgd	1	3000	0.3745284975
sgd	1	5000	0.3702289462
sgd	7	1000	0.3773933947
sgd	7	3000	0.3603681922
sgd	7	5000	0.3569564819
sgd	15	1000	0.3963693976
sgd	15	3000	0.3547196984
sgd	15	5000	0.3640032113

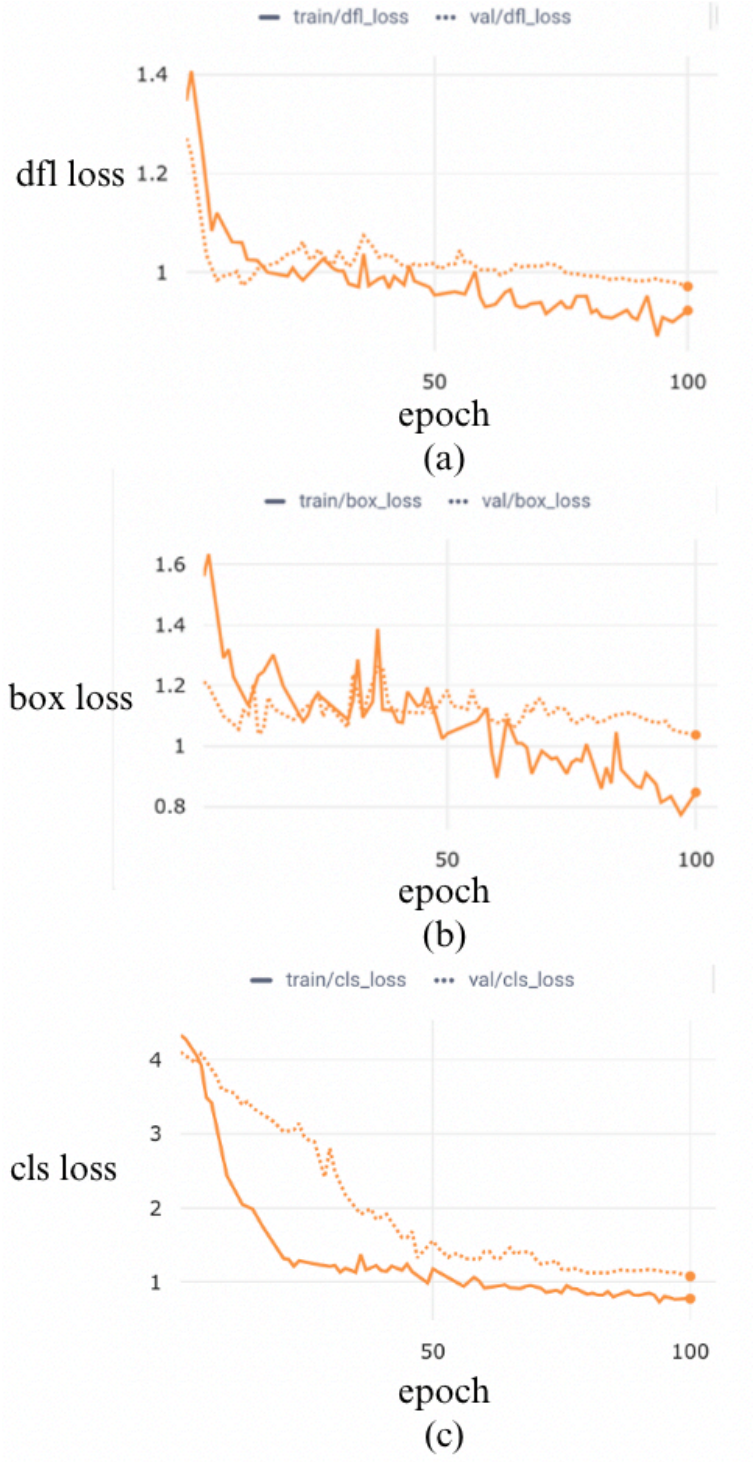
4.3. Eksperimen Model Deteksi Buah Kakao untuk Mengestimasi Jumlah Buah Kakao

Pada pelatihan model CNN YOLO, digunakan 80 gambar dan anotasi yang menggambarkan buah kakao matang dan belum matang dengan lima variasi epoch, yaitu 100, 300, 500, 700, 1000. Pada pelatihan ini, menggunakan optimizer SGD dan kombinasi dari beberapa fungsi loss. YOLOv8 menggunakan fungsi loss CIoU dan DFL untuk loss kotak pembatas dan binary cross-entropy untuk loss klasifikasi. Pelatihan akan dilakukan dengan menggunakan 2 skala arsitektur yaitu pada skala nano dan medium. Perbedaan skala arsitektur terdapat pada perbedaan jumlah parameter yang digunakan pada setiap layernya. Model dengan arsitektur nano juga akan memproduksi model dengan ukuran *file* yang lebih kecil ketimbang model dengan medium.

Pelatihan dilakukan selama 100 epoch dengan optimizer SGD dan *batch size* sebesar 16. Ukuran gambar input adalah 800x800, dan model akan disimpan setelah pelatihan selesai. Tidak digunakan cache, dan perangkat yang digunakan akan disesuaikan secara otomatis. Selain itu, dilakukan pengolahan data paralel dengan 8 workers. Informasi tentang pelatihan, seperti verbose, seed, dan deterministic juga telah ditentukan. Model ini dapat mendeteksi multiple kelas, dan tidak menggunakan bobot gambar atau training rectangular. Pada evaluasi, akan diperhatikan nilai ambang batas kepercayaan dan IoU threshold, serta jumlah deteksi maksimum. Grafik pelatihan akan ditampilkan, tetapi tidak akan menampilkan gambar deteksi secara visual. Tidak ada penyimpanan dalam format teks, confidence map, atau crop objek deteksi. Label objek dan confidence score akan ditampilkan, serta tebal garis boks deteksi sebesar 3.

4.3.1. Model YOLOV8n dengan 100 epoch

Dilakukan pelatihan model hingga epoch 100 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan *test* bisa dilihat pada gambar 4.24.



Gambar 4.25 loss model YOLOV8n dengan 100 epoch
(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.24 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-100. Pergerakan

nilai pelatihan *(train)* dan pengujian *(test)* tidak menunjukkan indikasi adanya overfitting atau underfitting. Nilai loss terus menurun seiring berjalannya pelatihan hingga epoch 100. Hasil akhir pelatihan dapat dilihat pada tabel 4.3.

Tabel 4.3 Hasil Pelatihan YOLOV8n pada epoch 100

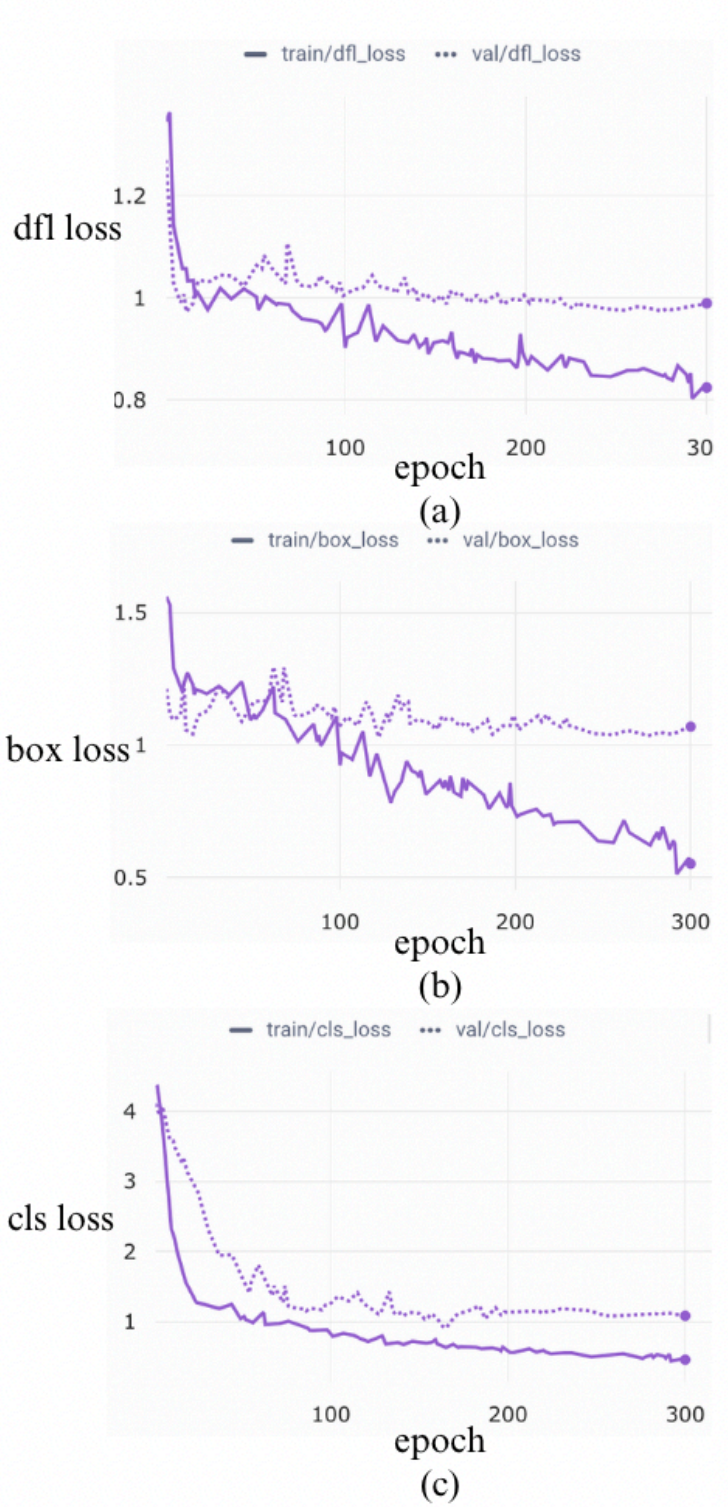
Class	**Box(P)**	**R**
all	0.907	0.704
ripe cocoa	0.925	0.792
unripe cocoa	0.889	0.617

Setelah melatih model YOLOV8M selama 100 epoch, dapat disimpulkan bahwa hasil pelatihan menunjukkan kinerja yang baik dalam melakukan deteksi buah kakao matang dan belum matang. Keseluruhan model mencapai akurasi deteksi sebesar 83.4% dengan recall sebesar 82%. Hasil yang lebih baik diperoleh untuk deteksi buah kakao matang, dengan akurasi mencapai 88% dan recall sebesar 91.7%. Namun, performa deteksi pada buah kakao belum matang sedikit lebih rendah, dengan akurasi sebesar 78.8% dan recall sebesar 72.3%.

Meskipun demikian, secara keseluruhan model telah mampu melakukan deteksi dengan baik pada dataset yang digunakan. Namun, ada ruang untuk pengembangan lebih lanjut dalam meningkatkan performa deteksi pada buah kakao belum matang agar sejajar dengan deteksi buah kakao matang. Dengan demikian, dapat dilakukan penyesuaian atau peningkatan model untuk mencapai akurasi deteksi yang lebih tinggi dan recall yang lebih baik pada kelas buah kakao belum matang.

4.3.2. Model YOLOV8n dengan 300 epoch

Dilakukan pelatihan model hingga epoch 300 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan *test* bisa dilihat pada gambar 4.25.



Gambar 4.26 loss model YOLOV8n dengan 300 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.25 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-300. Nilai loss

terus menurun seiring berjalannya pelatihan hingga epoch 300. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss *train* dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.4.

Tabel 4.4 Hasil Pelatihan YOLOV8n pada epoch 300

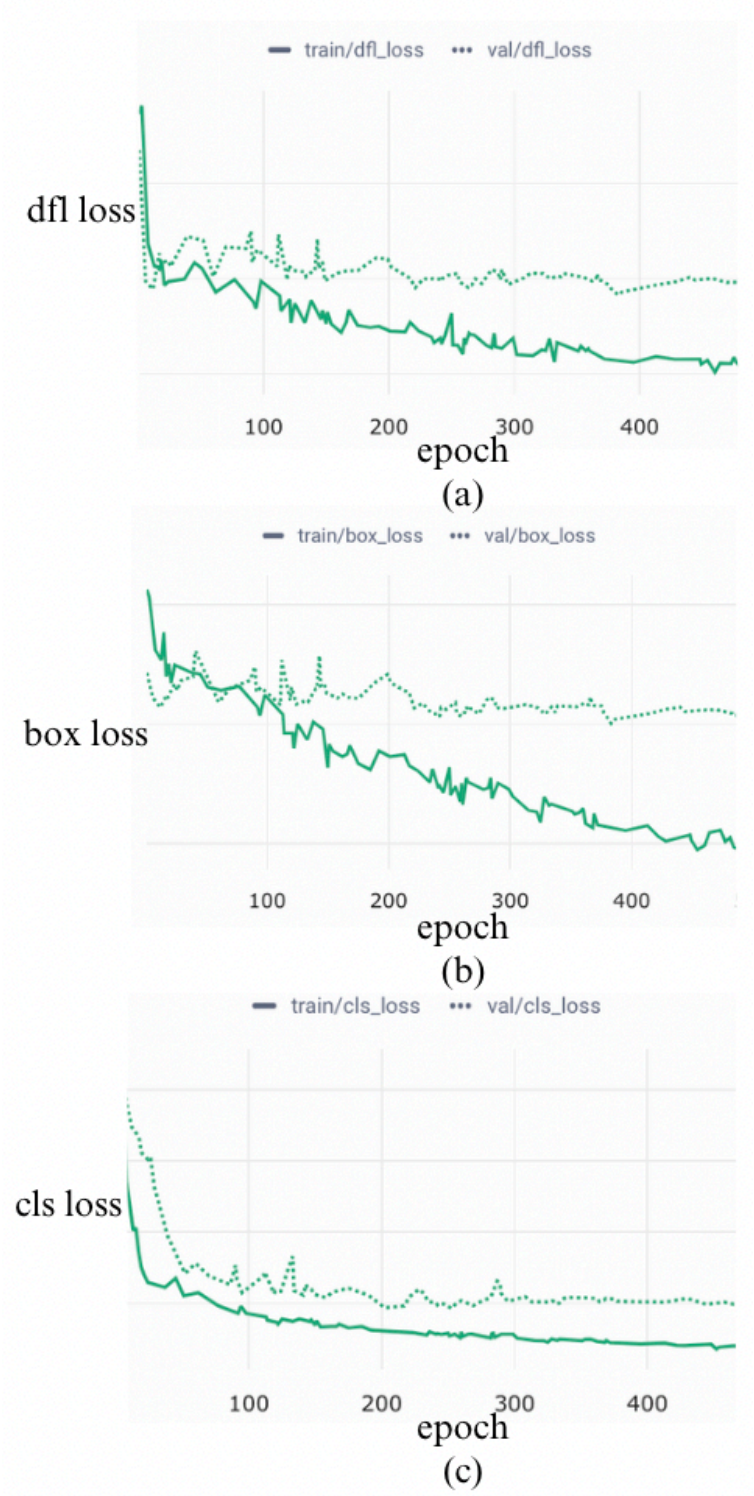
Class	**Box(P)**	**R**
all	0.799	0.799
ripe cocoa	0.734	0.920
unripe cocoa	0.864	0.678

Setelah melatih model, diperoleh hasil pelatihan yang menunjukkan kinerja yang cukup baik dalam melakukan deteksi buah kakao matang dan belum matang. Dalam hal ini, model mencapai akurasi deteksi sebesar 79.9% dan recall (kemampuan mengidentifikasi dengan benar) sebesar 79.9% untuk semua kelas. Meskipun akurasi dan recall secara keseluruhan seimbang, terdapat perbedaan dalam kinerja deteksi antara kelas buah kakao matang dan belum matang.

Untuk kelas buah kakao matang, model mencapai akurasi deteksi sebesar 73.4% dengan recall sebesar 92%. Hal ini menunjukkan bahwa model mampu mengenali dengan baik buah kakao yang telah matang. Namun, untuk kelas buah kakao belum matang, model memiliki akurasi deteksi yang sedikit lebih tinggi sebesar 86.4% namun recall yang sedikit lebih rendah, yakni sebesar 67.8%. Secara keseluruhan, model telah mencapai kinerja yang cukup baik dalam melakukan deteksi buah kakao matang dan belum matang.

4.3.3. Model YOLOV8n dengan 500 epoch

Dilakukan pelatihan model hingga epoch 500 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan *test* dapat dilihat pada gambar 4.26.



Gambar 4.27 loss model YOLOV8n dengan 500 epoch
(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.26 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-200. Nilai loss

terus menurun seiring berjalannya pelatihan hingga epoch 200. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss *train* dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.5.

Tabel 4.5 Hasil Pelatihan YOLOV8n pada epoch 500

Class	**Box(P)**	**R**
all	0.860	0.809
ripe cocoa	0.867	1.000
unripe cocoa	0.853	0.618

Berdasarkan hasil pelatihan model YOLOV8N selama 500 epoch, secara keseluruhan model ini memiliki tingkat akurasi deteksi yang baik sebesar 0,86 dan presisi yang cukup tinggi sebesar 0,809. Model ini mampu dengan efektif mendeteksi dan mengklasifikasikan cokelat matang ("ripe cocoa") dengan tingkat akurasi yang tinggi sebesar 0,867 dan recall sempurna sebesar 1. Namun, model menghadapi beberapa kesulitan dalam mendeteksi cokelat yang belum matang ("unripe cocoa"), dengan tingkat akurasi yang sedikit lebih rendah sebesar 0,853 dan recall sebesar 0,618. Untuk meningkatkan performa model dalam mengklasifikasikan cokelat yang belum matang, perlu dilakukan penyesuaian pada proses pelatihan, seperti penambahan data latihan yang lebih representatif atau penyetelan parameter model yang lebih optimal. Secara keseluruhan, meskipun model YOLOV8N telah menunjukkan kinerja yang baik dalam mendeteksi objek secara umum, masih diperlukan peningkatan dalam mengklasifikasikan cokelat yang belum matang dengan lebih baik.

4.3.4. Model YOLOV8n dengan 700 epoch

Dilakukan pelatihan model hingga epoch 700 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan *test* bisa dilihat pada gambar 4.27.

Gambar 4.28 loss model YOLOV8n dengan 700 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.27 terlihat bahwa model berhasil meningkatkan akurasi dengan baik. Nilai loss terus menurun seiring

berjalannya pelatihan model. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.6.

Tabel 4.6 Hasil Pelatihan YOLOV8n pada epoch 700

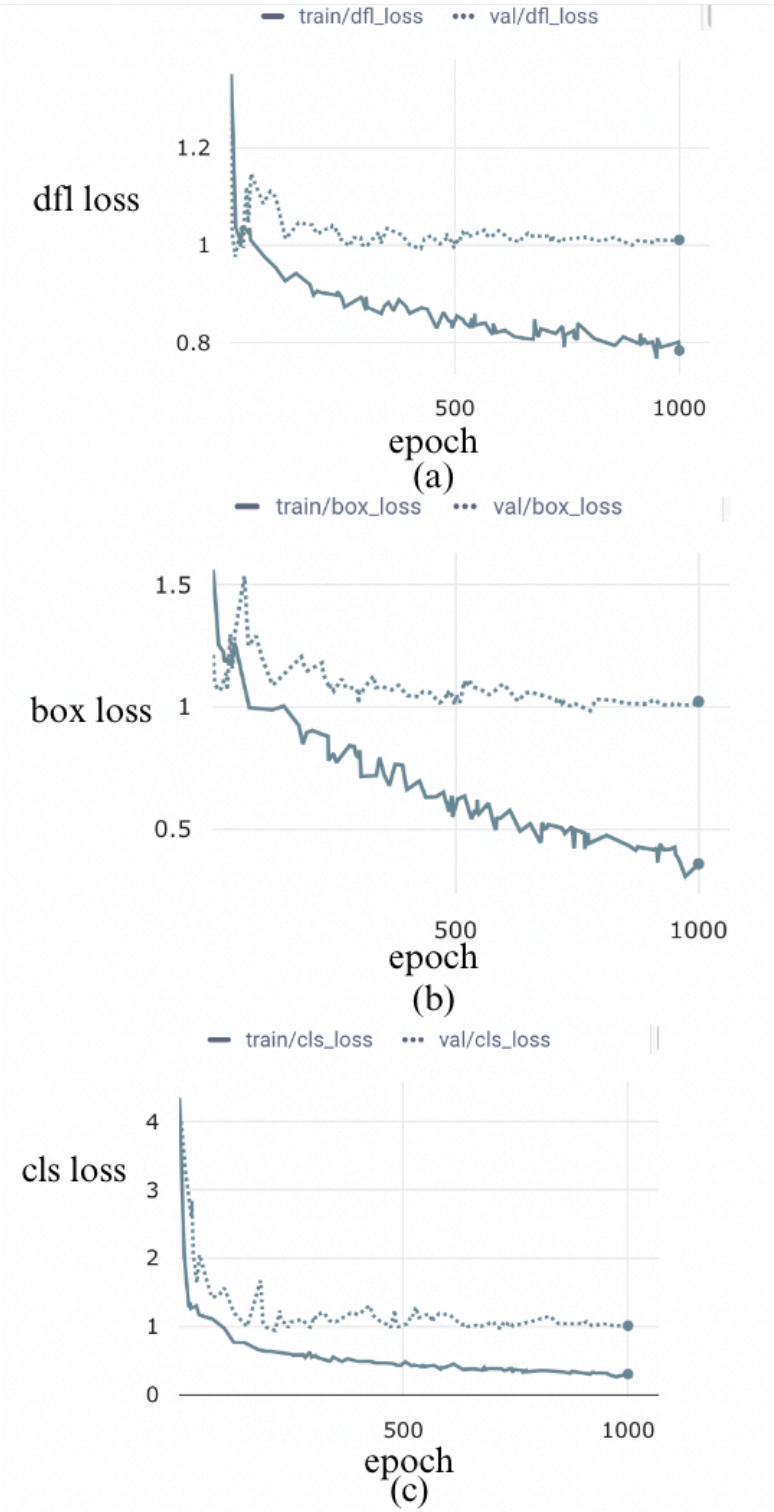
Class	**Box(P)**	**R**
all	0.917	0.707
ripe cocoa	0.947	0.743
unripe cocoa	0.887	0.671

Dalam hasil pelatihan model YOLOV8N selama 700 epoch, terjadi peningkatan yang signifikan dalam kinerja model. Secara keseluruhan, model mencapai tingkat akurasi deteksi yang tinggi sebesar 0,917 dengan nilai presisi sebesar 0,707. Ini menunjukkan bahwa model berhasil meningkatkan kemampuannya dalam mendeteksi objek secara umum. Hasil yang lebih baik ini dapat memberikan kepercayaan lebih dalam penggunaan model untuk mendeteksi berbagai objek di dalam gambar.

Ketika fokus pada kategori "ripe cocoa", model menunjukkan peningkatan yang konsisten dengan tingkat akurasi sebesar 0,947 dan presisi sebesar 0,743. Meskipun recall masih dapat ditingkatkan, peningkatan ini menunjukkan bahwa model semakin mampu mengklasifikasikan cokelat matang dengan akurasi yang lebih tinggi. Namun, untuk kategori "unripe cocoa", meskipun terjadi peningkatan, model masih menghadapi beberapa tantangan dalam mendeteksi dan mengklasifikasikan cokelat yang belum matang. Dengan tingkat akurasi sebesar 0,887 dan recall sebesar 0,671, masih ada ruang untuk perbaikan lebih lanjut.

4.3.5. Model YOLOV8n dengan 1000 epoch

Dilakukan pelatihan model hingga epoch 100 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.28.



Gambar 4.29 dfl loss, box loss, cls loss model YOLOV8n dengan 1000 epoch (a) dfl loss, (b) box loss dan (c) cls

Seperti yang dapat dilihat pada ketiga gambar 4.28 terlihat bahwa model berhasil meningkatkan akurasi dengan baik. Nilai loss terus menurun seiring

berjalannya pelatihan model. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-400 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil Pelatihan YOLOV8n pada epoch 1000

Class	**Box(P)**	**R**
all	0.869	0.767
ripe cocoa	0.775	0.917
unripe cocoa	0.963	0.617

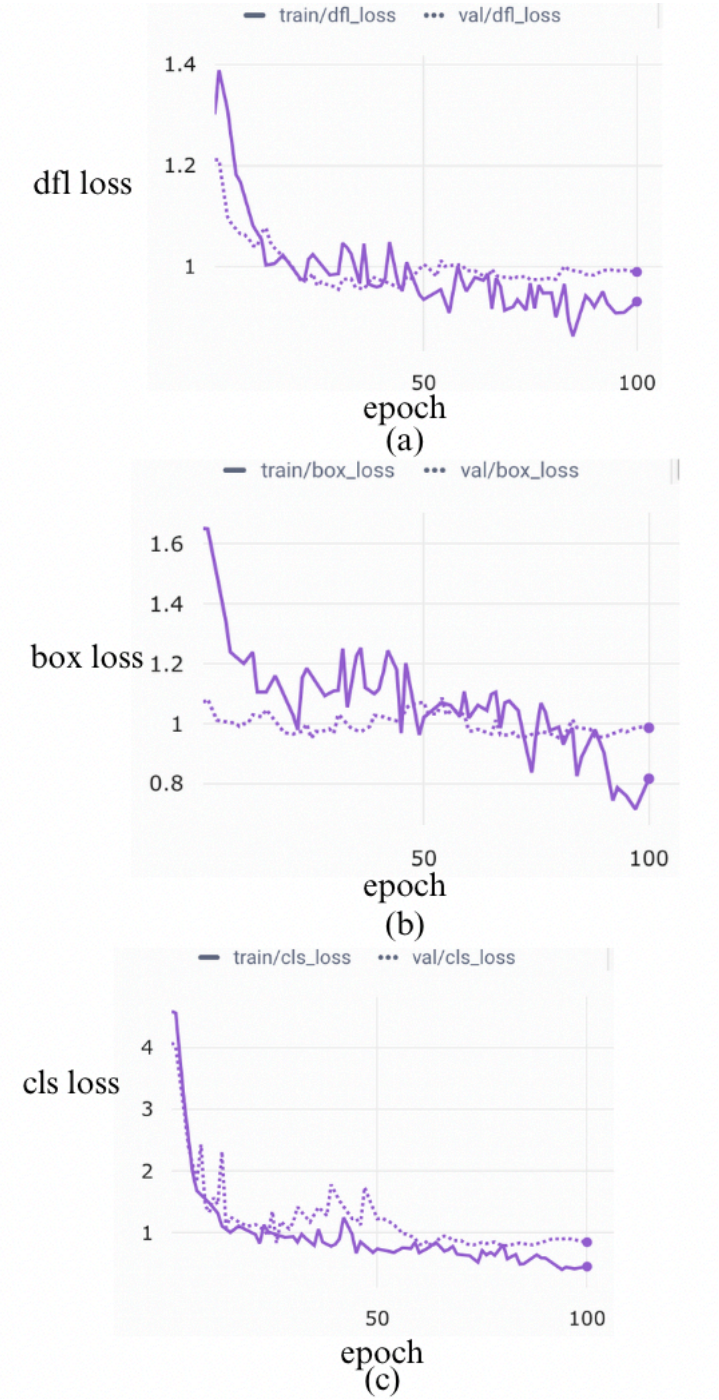
Dalam pelatihan model YOLOV8N, terdapat hasil yang menarik untuk setiap kategori. Secara keseluruhan, model mencapai tingkat akurasi deteksi sebesar 0,869 dengan presisi sebesar 0,767. Meskipun tingkat presisi yang tinggi menunjukkan kemampuan model dalam mengenali objek secara spesifik, recall yang sebesar 0,767 menandakan adanya ruang untuk perbaikan dalam mencakup semua objek yang ada dalam gambar secara lebih lengkap.

Dalam kategori "ripe cocoa", model menunjukkan tingkat presisi sebesar 0,775 yang cukup baik. Namun, recall yang rendah sebesar 0,917 mengindikasikan bahwa model masih melewatkan beberapa cokelat matang yang seharusnya terdeteksi. Sementara itu, dalam kategori "unripe cocoa", model berhasil mencapai tingkat presisi yang tinggi sebesar 0,963. Namun, recall yang rendah sebesar 0,617 menunjukkan bahwa model masih menghadapi kesulitan dalam mendeteksi sebagian besar cokelat yang belum matang.

Secara keseluruhan, model YOLOV8N telah menunjukkan kemajuan yang baik dalam deteksi objek, namun masih ada aspek-aspek yang perlu ditingkatkan. Peningkatan pada recall dalam kedua kategori "ripe cocoa" dan "unripe cocoa" akan menjadi prioritas dalam pelatihan selanjutnya. Dengan demikian, model dapat mengenali dan mengklasifikasikan objek dengan lebih akurat dan menyeluruh, menghasilkan hasil yang lebih baik dalam aplikasi deteksi objek yang berkaitan dengan cokelat..

4.3.6. Model YOLOV8m dengan 100 epoch

Dilakukan pelatihan model hingga epoch 100 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.29.



Gambar 4.30 loss model YOLOV8n dengan 100 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.29 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-100. Pergerakan nilai pelatihan *(train)* dan pengujian *(test)* tidak menunjukkan indikasi adanya *overfitting* atau *underfitting*. Nilai *loss* terus menurun seiring berjalannya pelatihan hingga epoch 100. Hasil akhir pelatihan dapat dilihat pada tabel 4.8.

Tabel 4.8 Hasil Pelatihan YOLOV8M pada epoch 100

Class	**Box(P)**	**R**
all	0.834	0.820
ripe cocoa	0.880	0.917
unripe cocoa	0.788	0.723

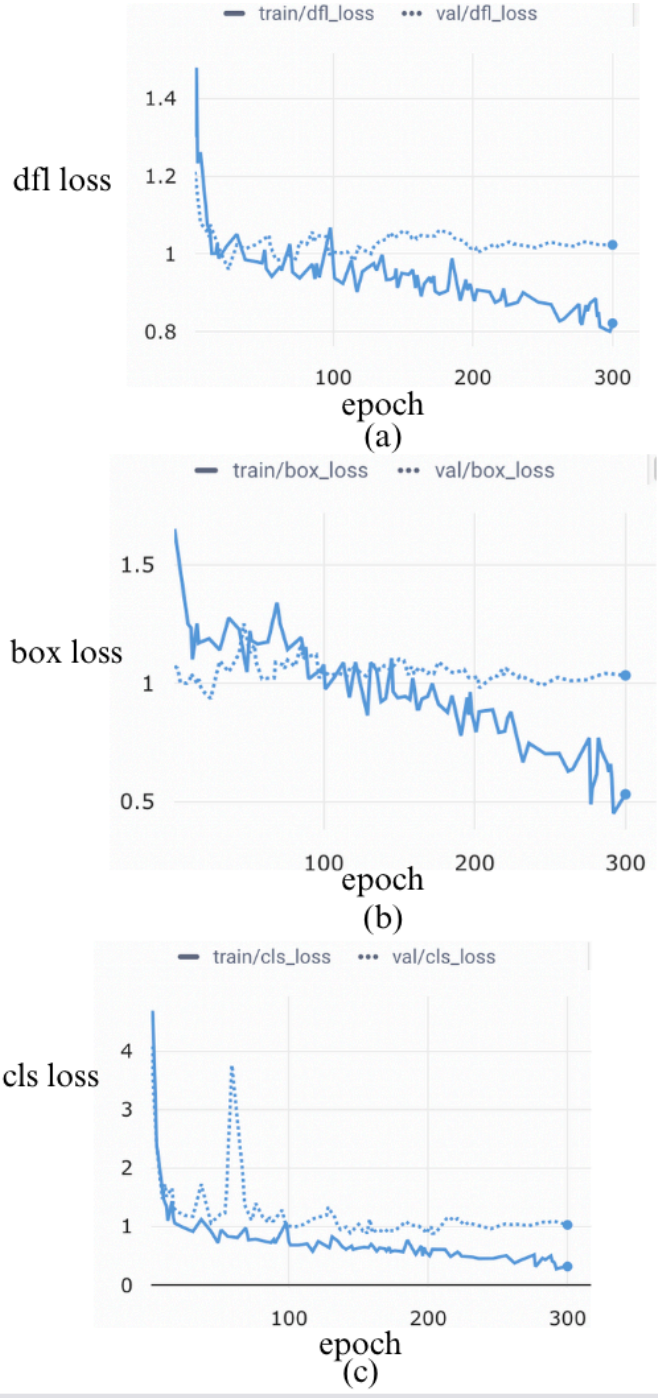
Model YOLOV8M, setelah melalui pelatihan selama 100 epoch, menunjukkan hasil yang cukup baik. Dalam kategori "all", model mencapai tingkat akurasi deteksi sebesar 0,834 dengan presisi sebesar 0,82. Hasil ini menunjukkan kemampuan model dalam mendeteksi objek secara umum dalam dataset yang digunakan. Tingkat akurasi yang tinggi ini memberikan kepercayaan bahwa model dapat mengenali objek dengan baik.

Ketika berfokus pada kategori "ripe cocoa", model YOLOV8M mencapai tingkat akurasi yang lebih tinggi sebesar 0,88 dengan recall sebesar 0,917. Hal ini menunjukkan kemampuan model dalam mengklasifikasikan cokelat matang dengan akurasi yang baik dan mampu mendeteksi sebagian besar objek yang ada. Tingkat recall yang tinggi juga menandakan bahwa model dapat mengenali sebagian besar cokelat matang yang ada dalam dataset.

Namun, dalam kategori "unripe cocoa", model menghadapi beberapa tantangan dengan tingkat akurasi sebesar 0,788 dan recall sebesar 0,723. Hal ini mengindikasikan bahwa model masih perlu ditingkatkan dalam mendeteksi dan mengklasifikasikan cokelat yang belum matang. Secara keseluruhan, model YOLOV8M menunjukkan kemampuan yang baik dalam mendeteksi objek secara umum dan khususnya dalam kategori "ripe cocoa".

4.3.7. Model YOLOV8m dengan 300 epoch

Dilakukan pelatihan model hingga epoch 300 menggunakan arsitektur YOLOV8m. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.30.



Gambar 4.31 dfl loss, box loss, cls loss model YOLOV8n dengan 300 epoch (a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.30 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-200. Nilai loss terus menurun seiring berjalannya pelatihan. Namun pada metrik dfl loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.8.

Tabel 4.9 Hasil Pelatihan YOLOV8m pada epoch 300

Class	**Box(P)**	**R**
all	0.820	0.725
ripe cocoa	0.903	0.875
unripe cocoa	0.737	0.574

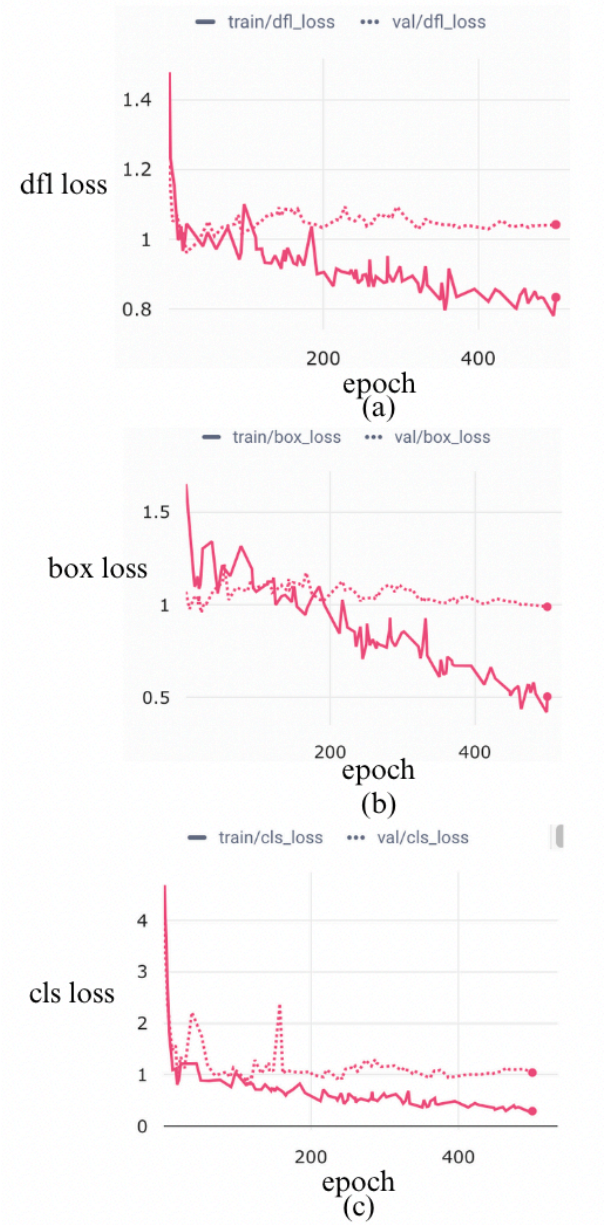
Setelah melalui pelatihan selama 300 epoch, model YOLOV8M menunjukkan hasil yang menarik. Secara keseluruhan, model ini mencapai tingkat akurasi deteksi sebesar 0,82 dengan presisi sebesar 0,725 dalam kategori "all". Meskipun tingkat akurasi yang cukup baik, recall yang sebesar 0,725 menunjukkan bahwa model mungkin masih melewatkan sebagian objek yang ada dalam gambar. Dalam kategori "ripe cocoa", model menunjukkan peningkatan performa dengan tingkat akurasi sebesar 0,903 dan recall sebesar 0,875. Hasil ini mengindikasikan kemampuan model dalam mengklasifikasikan cokelat matang dengan akurasi yang tinggi dan mampu mendeteksi sebagian besar objek yang ada. Peningkatan tersebut menunjukkan adanya kemajuan dalam pelatihan model.

Namun, dalam kategori "unripe cocoa", model masih mengalami beberapa kendala dengan tingkat akurasi sebesar 0,737 dan recall sebesar 0,574. Hal ini menandakan bahwa model masih menghadapi kesulitan dalam mendeteksi dan mengklasifikasikan cokelat yang belum matang dengan akurasi dan kelengkapan yang lebih baik. Secara keseluruhan, model YOLOV8M telah menunjukkan kemajuan dalam pelatihan selama 300 epoch. Meskipun tingkat akurasi dan presisi

dalam kategori "all" cukup baik, recall masih perlu ditingkatkan. Peningkatan signifikan terlihat dalam kategori "ripe cocoa" dengan akurasi dan recall yang lebih baik.

4.3.8. Model YOLOV8m dengan 500 epoch

Dilakukan pelatihan model hingga epoch 500 menggunakan arsitektur YOLOV8n. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.31.



Gambar 4.32 dfl loss, box loss, cls loss model YOLOV8n dengan 500 epoch (a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.25 terlihat bahwa model berhasil meningkatkan akurasi dengan baik sampai pada epoch ke-300. Nilai loss terus menurun seiring berjalannya pelatihan hingga epoch 300. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.9.

Tabel 4.10 Hasil Pelatihan YOLOV8m pada epoch 500

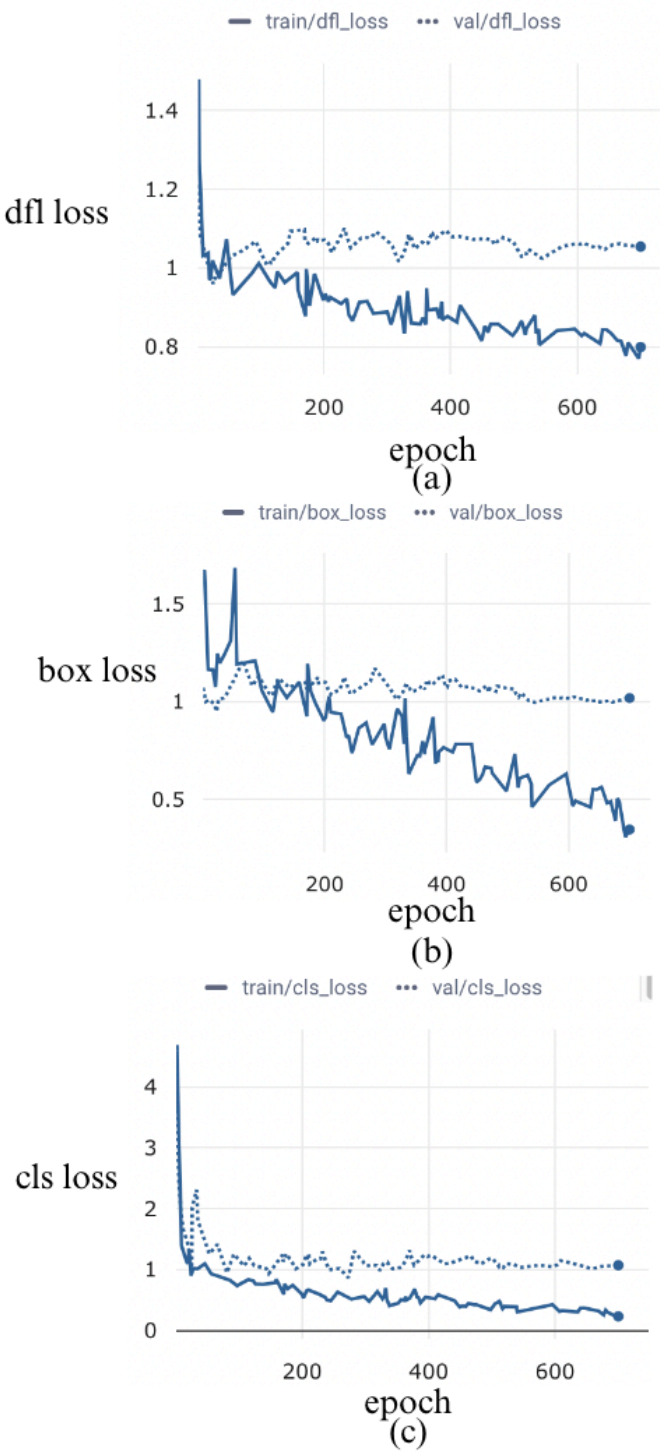
Class	**Box(P)**	**R**
all	0.910	0.704
ripe cocoa	0.959	0.792
unripe cocoa	0.861	0.617

Setelah melalui pelatihan selama 500 epoch, model YOLOV8M menunjukkan hasil yang menggembirakan. Secara keseluruhan, model ini mencapai tingkat akurasi deteksi yang tinggi sebesar 0,91 dengan presisi sebesar 0,704 dalam kategori "all". Meskipun tingkat akurasi yang tinggi, recall yang sebesar 0,704 menunjukkan bahwa model mungkin masih melewatkan sebagian objek yang ada dalam gambar.

Dalam kategori "ripe cocoa", model menunjukkan performa yang sangat baik dengan tingkat akurasi sebesar 0,959 dan recall sebesar 0,792. Hasil ini menunjukkan kemampuan model dalam mengklasifikasikan cokelat matang dengan akurasi yang tinggi dan mampu mendeteksi sebagian besar objek yang ada. Namun, dalam kategori "unripe cocoa", model masih menghadapi beberapa kendala dengan tingkat akurasi sebesar 0,861 dan recall sebesar 0,617. Hal ini menandakan bahwa model masih mengalami kesulitan dalam mendeteksi dan mengklasifikasikan cokelat yang belum matang dengan akurasi dan kelengkapan yang lebih baik.

4.3.9. Model YOLOV8m dengan 700 epoch

Dilakukan pelatihan model hingga epoch 700 menggunakan arsitektur YOLOV8m. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.32.



Gambar 4.33 loss model YOLOV8n dengan 700 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.32 terlihat bahwa model berhasil meningkatkan akurasi dengan baik. Nilai loss terus menurun seiring berjalannya pelatihan hingga epoch 300. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai loss train dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.10.

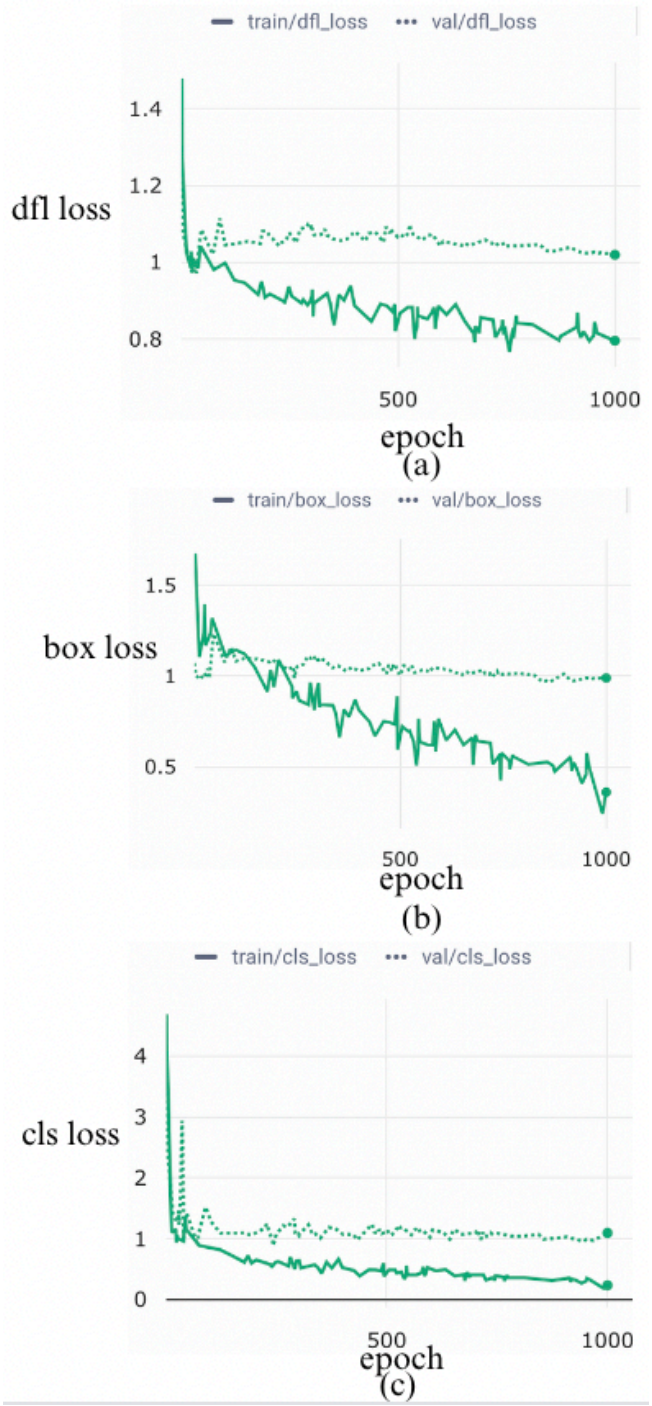
Tabel 4.11 Hasil Pelatihan YOLOV8m pada epoch 700

Class	**Box(P)**	**R**
all	0.748	0.808
ripe cocoa	0.610	0.958
unripe cocoa	0.885	0.658

Model YOLOV8M dilatih selama 700 epoch dan mencapai akurasi deteksi sebesar 0,748 dan presisi 0,808 dalam kategori "semua". Namun, recall 0,808 menunjukkan ruang untuk meningkatkan deteksi objek dalam gambar. Dalam kategori "kakao matang", akurasi 0,61 dengan recall tinggi 0,958 menunjukkan kemampuan model mengklasifikasikan kakao matang, meskipun dengan positif palsu. Dalam kategori "kakao mentah", akurasi 0,885 dengan recall 0,658 menunjukkan model masih kesulitan mendeteksi kakao mentah. Model perlu meningkatkan recall dalam kategori ini. Secara keseluruhan, YOLOV8M menunjukkan kemajuan selama 700 epoch. Dalam kategori "semua", recall perlu ditingkatkan untuk mendeteksi lebih banyak objek. Dalam kategori "kakao matang", recall tinggi menunjukkan kemampuan model, tetapi perlu penanganan positif palsu. Dalam kategori "kakao mentah", meningkatkan recall menjadi fokus utama untuk meningkatkan deteksi kakao mentah dengan akurasi yang lebih tinggi.

4.3.10. Model YOLOV8m dengan 1000 epoch

Dilakukan pelatihan model hingga epoch 1000 menggunakan arsitektur YOLOV8m. Pada akhir proses training, nilai akurasi dan loss pada training dan test bisa dilihat pada gambar 4.33.



Gambar 4.34 model YOLOV8n dengan 1000 epoch

(a) dfl loss, (b) box loss dan (c) cls loss

Seperti yang dapat dilihat pada ketiga gambar 4.32 terlihat bahwa model berhasil meningkatkan akurasi dengan baik. Nilai loss terus menurun seiring berjalannya pelatihan hingga epoch 300. Namun pada metrik dfl loss serta box loss ada indikasi terjadinya overfitting. Pada epoch ke-200 nilai *loss train* dan *test* mengalami pergerakan ke arah yang berbeda. Dimana nilai loss val meningkat namun nilai loss train menurun. Hasil akhir pelatihan model ini dapat dilihat pada tabel 4.11.

Tabel 4.12 Hasil Pelatihan YOLOV8m pada epoch 1000

Class	**Box(P)**	**R**
all	0.816	0.768
ripe cocoa	0.746	0.856
unripe cocoa	0.886	0.681

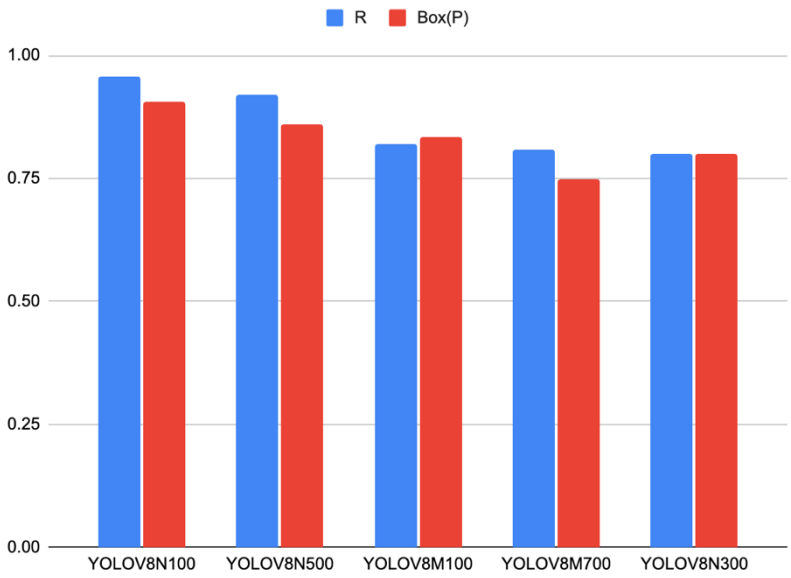
Model YOLOV8M telah dilatih selama 1000 epoch dan menunjukkan hasil yang menarik. Dalam kategori "semua", model ini mencapai presisi sebesar 0,816 dan recall sebesar 0,768. Ini menunjukkan kemampuan model dalam mengenali dan mengklasifikasikan objek dengan akurasi yang cukup tinggi. Dalam kategori "kakao matang", model mencapai presisi sebesar 0,746 dengan recall yang lebih tinggi, yaitu 0,856. Ini menunjukkan kemampuan model dalam mendeteksi dan mengklasifikasikan kakao matang dengan baik, meskipun masih terdapat ruang untuk meningkatkan presisi. Di sisi lain, dalam kategori "kakao mentah", model mencapai presisi sebesar 0,886 dengan recall sebesar 0,681.

Meskipun akurasi cukup baik, recall yang rendah menunjukkan bahwa model masih kesulitan dalam mendeteksi sebagian besar kakao mentah. Meningkatkan recall dalam kategori ini menjadi fokus utama untuk meningkatkan kemampuan model dalam mengklasifikasikan kakao mentah dengan akurasi yang lebih tinggi. Secara keseluruhan, YOLOV8M telah menunjukkan kemajuan dalam 1000 epoch pelatihan. Meskipun memiliki akurasi yang baik dalam kategori "semua" dan "kakao matang", meningkatkan recall akan membantu model dalam menangkap lebih banyak objek secara akurat. Untuk kategori "kakao mentah",

meningkatkan recall menjadi hal yang penting untuk meningkatkan kemampuan model dalam mendeteksi kakao mentah dengan lebih baik.

4.3.11. Evaluasi 5 Model YOLOV8 dengan performa terbaik

Untuk menentukan model YOLOV8 yang memiliki akurasi paling baik maka berikut akan ditampilkan 5 model dengan nilai R dan Box(P) (box loss) yang paling tinggi diantara model-model lainnya pada gambar 4.35.



Gambar 4.35 Grafik 5 Model Dengan Nilai Recall Tertinggi

Dari model dengan arsitektur YOLOV8 diatas maka disimpulkan model terbaik adalah model dengan arsitektur YOLOV8 nano dengan 100 epoch.

4.4. Sistem Prediksi Jarak Objek pada Citra

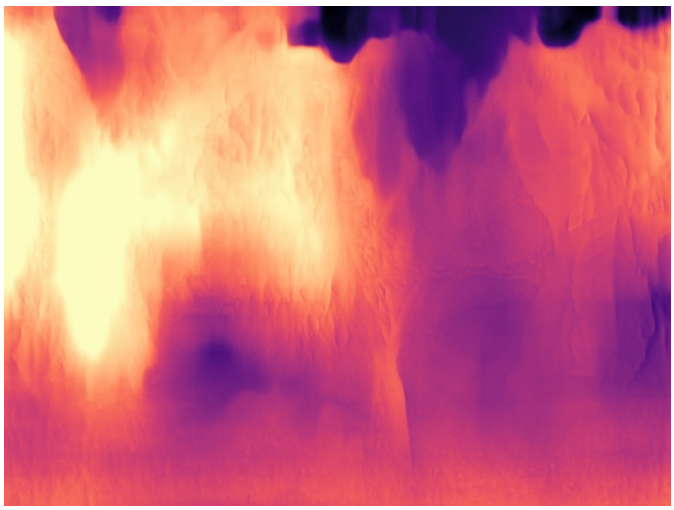
Sistem ini akan melakukan Prediksi Jarak Objek terhadap kamera. Prediksi dilakukan menggunakan nilai *rgb* sebagai input. Proses dimulai dengan memilih titik koordinat salah satu piksel yang berada pada area objek sebagai input proses segmentasi. Kemudian dilakukan segmentasi pada citra untuk mendapatkan area objek. Kemudian dilakukan prediksi kedalaman citra menggunakan Model CNN *Monocular Depth Estimation.* Area yang telah didapatkan digunakan untuk mengambil warna pada citra hasil prediksi kedalaman pada area yang diinginkan.

Kemudian dari kumpulan warna tersebut dicari nilai mediannya sehingga didapatkan nilai median rgb untuk digunakan pada input model ANN prediksi jarak. Sehingga kemudian didapatkan nilai jarak objek terhadap kamera.

4.4.1. Mengunggah Gambar dan Penggunaan Model CNN *Monocular Depth Estimation*

Proses awal yang harus dilakukan yakni mengunggah gambar tanaman kakao yang akan digunakan. Setelah itu dilakukan proses prediksi kedalaman menggunakan Model CNN *Monocular Depth Estimation.* Sehingga dihasilkan citra yang merepresentasikan kedalaman atau dapat disebut sebagai citra kedalaman. Citra inilah yang nantinya akan digunakan nilai rgbnya sebagai input untuk memprediksi jarak objek terhadap kamera.

Gambar 4.36 Citra Tanaman Kakao Asli



Gambar 4.37 Citra Kedalaman Tanaman Kakao

4.4.2. Menentukan Titik Piksel pada Objek

Karena sistem yang dikembangkan belum dapat mengidentifikasi bagian batang tanaman kakao secara otomatis, maka pada penelitian ini masih diperlukan *input manual* untuk menentukan titik piksel yang merupakan bagian dari tanaman kakao. Proses ini dilakukan dengan melihat terlebih dahulu gambar serta *axis* untuk melakukan taksiran lokasi x,y piksel yang akan digunakan.

Gambar 4.38 Tampilan awal untuk penentuan titik pada objek

Setelah melakukan taksiran maka pada gambar ini akan digunakan titik pada koordinat 2750, 3000. Nilai x,y tersebut akan ditampilkan dengan simbol bintang seperti pada gambar berikut.

Gambar 4.39 Tampilan gambar dengan titik piksel yang telah dipilih

Setelah menemukan titik piksel yang akan digunakan, maka pada proses selanjutnya titik tersebut akan digunakan sebagai input pada proses segmentasi.

4.4.3. Proses Segmentasi dengan Model CNN *Segment Anything*

Pada proses ini akan dilakukan segmentasi untuk mendapatkan area objek yang akan digunakan (Tanaman Kakao). Titik koordinat yang telah ditentukan menjadi input/acuan model untuk memprediksi area sekitarnya yang masih merupakan bagian dari objek. Hal ini dilakukan untuk mempermudah pemilihan area objek.

Gambar 4.40 Hasil Segmentasi Area Objek Tanaman Kakao

Area/*mask* tersebut kemudian akan digunakan untuk mengambil nilai rgb pada citra kedalaman.

4.4.4. Pengambilan nilai median RGB dan Prediksi Jarak

Setelah mendapatkan area objek tanaman kakao, maka nilai rgb pada citra kedalaman yang beririsan dengan area objek tanaman kakao akan diambil. Dari nilai-nilai tersebut kemudian akan didapatkan nilai median RGB. Nilai median

RGB kemudian digunakan untuk memprediksi jarak. Dengan menggunakan model ANN yang telah dikembangkan, dilakukan prediksi jarak menggunakan nilai rgb sebagai input. Sehingga kemudian didapatkan nilai jarak objek tanaman kakao terhadap kamera dalam satuan meter.

4.4.5. Evaluasi Sistem Prediksi Jarak Objek Pada Citra

Penggunaan model CNN *Segment Anything* belum bisa optimal untuk mengambil bagian tanaman saja karena adanya noise pada background objek tersebut. Hal ini dipengaruhi oleh kondisi lingkungan perkebunan kakao yang padat dengan tanaman kakao.

4.5. Sistem Prediksi Koordinat Tanaman Kakao pada Citra

Pada bagian ini akan dilakukan prediksi nilai koordinat tanaman kakao. Pada tahap ini diperlukan beberapa variabel sebagai input yaitu nilai koordinat longitude dan latitude tanaman kakao, nilai derajat arah hadap kamera, serta jarak objek terhadap kamera. Pada proses ini akan digunakan rumus yang bernama *Vincenty Formula.* Hasil dari kalkulasi menggunakan rumus tersebut yakni titik koordinat longitude serta latitude objek tanaman kakao. Untuk menguji akurasi dari rumus tersebut, pada bagian selanjutnya akan dilakukan percobaan kalkulasi pada 5 titik lokasi.

4.5.1. Evaluasi Akurasi *Vincenty Formula*

Pada bagian ini akan dilakukan pengujian akurasi dari hasil kalkulasi *Vinceny Formula.* Pengujian akan dilakukan pada 5 lokasi pada tabel 4.13.

Tabel 4.13 Ground Truth Titik Koordinat Pengujian Vincety Formula

No	Building		Destination	
	Name	Coordinate	Name	Coordinate
1	Alun-Alun Tugu Malang	-7.97692544511029, 112.634055029002	Stasiun Malang Kota Baru	-7.97720169566579, 112.637112747216

Tabel 4. 14 Ground Truth Titik Koordinat Pengujian Vincety Formula (lanjutan)

No	Building		Destination	
	Name	Coordinate	Name	Coordinate
2	Alun-Alun Tugu Malang	-7.97692544511029, 112.634055029002	Bulan Photocopy & Print	-7.97419601839356, 112.634292801417
3	Gerbang UB Soekarno hatta	-7.94984, 112.615411	Soekarno Hatta Bridge	-7.9496079959561, 112.615839888356
4	Gerbang UB Soekarno hatta	-7.94984, 112.615411	Kober Mie Setan	-7.9481629265072, 112.61676260122
5	Gerbang UB Soekarno hatta	-7.94984, 112.615411	Mixue Suhat Malang	-7.94637779324508, 112.618050061491

Tabel 4. 15 Arah dan Jarak

No	Heading	Distance to Desstination (km)
1	96,607°	0.28
2	4,124	0.3
3	45,456°	0.069
4	43,014°	0.22
5	-40,579°	0.45

Tabel 4. 16 Hasil Prediksi dan Selisih

No	Predicted Coordinate	Differences
1	-7.97721516624731, 112.636580818575	-1.347×10^{-5}, -5.319×10^{-4}
2	-7.97423446585066, 112.634250919231	-3.845×10^{-5}, -4.188×10^{-5}
3	-7.94936783429258, 112.615817547406	-3.845×10^{-5}, -4.188×10^{-5}

Tabel 4. 17 Hasil Prediksi dan Selisih (lanjutan)

No	Predicted Coordinate	Differences
4	-7.94839333872107, 112.616773784662	-2.304×10^{-4}, 1.118×10^{-5}
5	-7.9467662955678, 112.618069047894	-3.885×10^{-4}, 1.899×10^{-5}

Berdasarkan lima percobaan yang telah dilakukan, dapat disimpulkan bahwa rumus yang digunakan memiliki akurasi yang cukup baik dan kesalahan yang relatif kecil. Perbedaan rata-rata antara hasil yang diperoleh dari rumus dan titik koordinat sebenarnya adalah sangat kecil, yaitu sebesar -0,00009966528437. Selain itu, perbedaan maksimum antara hasil yang diperoleh dari rumus dan nilai sebenarnya juga cukup kecil, hanya sekitar 0,0002401616635. Sehingga dapat disimpulkan rumus *Vincenty Formula* tersebut dapat digunakan untuk melakukan kalkulasi titik koordinat suatu objek berdasarkan titik koordinat asal, derajat arah serta jarak terhadap objek.

4.6. Sistem Estimasi Jumlah Buah Kakao

Sistem ini melakukan deteksi buah kakao. Buah kakao yang terdeteksi dalam bentuk bounding box akan dihitung jumlahnya. Setelah terdeteksi maka setiap buah dalam setiap bounding box tersebut diklasifikasi untuk memprediksi buah tersebut telah memasuki usia matang atau belum. Proses pada sistem ini cukup sederhana, pengguna cukup mengunggah citra tanaman kakao. Lalu sistem akan melakukan deteksi buah kakao, serta mengkalkulasi bounding box yang muncul. Sehingga akan ditampilkan kepada pengguna berapa jumlah buah kakao yang ada pada tanaman kakao tersebut. Adapun deteksi ini dilakukan dengan menggunakan confidence threshold sebesar 0.25.

4.6.1. Evaluasi Sistem Estimasi Jumlah Buah Kakao

Pada tahap ini akan dilakukan evaluasi pada 5 model CNN YOLOV8 terbaik yang digunakan untuk melakukan deteksi buah kakao. Selain metriks pelatihan

seperti box loss, dfl loss dan cls loss akan dilakukan pula pengujian manual secara visual. Model akan dijalankan untuk melakukan proses deteksi, lalu hasil deteksi akan dibandingkan dengan hasil penghitungan buah kakao manual secara visual. Adapun berikut 2 citra yang akan digunakan untuk melakukan evaluasi sistem estimasi jumlah buah kakao yaitu gambar 4.41 dan 4.42.

Gambar 4. 41 Citra Tanaman Kakao dengan Buah muda

Dapat dilihat pada gambar 4.40 terdapat 19 buah muda. Buah kakao yang masih muda akan nampak berwarna hijau. Lama-kelamaan akan muncul titik titik kecoklatan pada buah kakao. Beberapa buah kakao pada gambar 4.40 berada pada posisi yang sulit untuk dideteksi seperti buah yang tertutupi oleh buah lainnya, serta buah yang berada dibalik batang. Buah pada posisi yang sulit dijangkau ini akan sulit untuk dideteksi oleh model CNN YOLO.

Gambar 4. 42 Citra Tanaman Kakao dengan Buah Matang

Dapat dilihat pada gambar 4.41 terdapat 20 buah matang. Pada kedua citra diatas yaitu gambar 4.40 dan gambar 4.41 akan dilakukan deteksi menggunakan model YOLOV8n dengan 100 epoch. Hasil deteksi dapat dilihat pada gambar 4.43 dan 4.44.

Gambar 4. 43 Hasil Deteksi Buah Kakao menggunakan model YOLOV8n dengan 100 epoch

Pada gambar 4.43 terdapat 8 bounding box. Artinya terdapat 8 buah kakao matang yang terdeteksi. Kedelapan bounding box tersebut juga memiliki confidece threshold yang tinggi, yaitu pada nilai diatas 0.9 yang artinya model cukup yakin bahwa prediksinya akurat. Namun masih terdapat 10 buah yang tidak terdeteksi. Buah yang tidak terdeteksi kebanyakan memiliki posisi yang susah untuk dideteksi seperti berada dibalik batang, dibalik buah lainnya, serta ukurannya sangat kecil. Deteksi dilakukan pula pada tanaman kakao dengan buah muda. Hasil deteksi dapat dilihat pada gambar 4.44.

Gambar 4. 44 Hasil Deteksi Buah Kakao menggunakan model YOLOV8n dengan 100 epoch

Seperti yang dapat dilihat pada gambar 4.44, terdapat 9 bounding box/buah yang berhasil terdeteksi. Namun pada ground truth terdapat 19 buah, sehingga masih terdapat 10 buah. Deteksi dilakukan menggunakan kelima model pada 2 gambar tersebut. Hasil evaluasi pada kelima model dapat dilihat pada tabel 4.15.

Tabel 4. 18 Evaluasi 5 model terbaik pada 2 contoh gambar

model	filename	prediction result	
		num of ripe cocoa	num of unripe cocoa
YOLOV8n 100 epoch	IMG_20230621_160151.jpg		9
	IMG_20230621_160747.jpg	8	

Tabel 4. 19 Evaluasi 5 model terbaik pada 2 contoh gambar (lanjutan)

model	filename	prediction result	
		num of ripe cocoa	num of unripe cocoa
YOLOV8n 500 epoch	IMG_20230621_160151.jpg		12
	IMG_20230621_160747.jpg	10	
YOLOV8m 100 epoch	IMG_20230621_160151.jpg	2	14
	IMG_20230621_160747.jpg	17	
YOLOV8m 700 epoch	IMG_20230621_160151.jpg	3	11
	IMG_20230621_160747.jpg	11	
YOLOV8n 300 epoch	IMG_20230621_160151.jpg		10
	IMG_20230621_160747.jpg	11	

Secara keseluruhan model belum berhasil mendeteksi semua buah kakao yang ada pada tanaman kakao. Hal ini disebabkan karena banyak posisi buah kakao yang tertutupi oleh objek lain seperti batang tanaman kakao, buah kakao, serta ukurannya yang kecil. Hal ini dapat dikembangkan dengan cara melengkapi dataset yang lebih bervariatif seperti menambahkan anotasi bounding box pada buah kakao yang tertutupi. Namun hal tersebut juga perlu diimbangi dengan adanya metode yang dapat melokalisasi hasil deteksi karena apabila sebuah kakao terletak dibalik batang sehingga seolah-olah nampak terdapat dua buah kakao maka menjadi akan bias bagi model.

Apabila ketiga sistem diatas digabungkan, sistem dapat digunakan sebagai alat untuk melakukan monitoring perkebunan kakao dengan metode geotagging. Pada

penelitian ini sistem hanya mampu untuk melakukan prediksi titik koordinat tanaman kakao serta deteksi buah kakao. Sehingga sistem dapat menghasilkan *output* berupa titik koordinat serta jumlah buah kakao. Sistem ini merupakan alternatif murah untuk mengetahui perkembangan dari setiap tanaman kakao dari waktu ke waktu.

BAB V SIMPULAN DAN SARAN

5.1. Simpulan

Berikut kesimpulan pada penelitian ini.

1. Pada penelitian ini telah berhasil dikembangkan metode geotagging dengan beberapa sistem didalamnya diantaranya model CNN estimasi kedalaman, *Segment Anything,* ANN prediksi jarak, serta rumus *Vincenty Formula.* Dimana hasil eksperimen ANN menghasilkan model ANN dengan nilai loss mae sebesar 0.333776. Dimana model tersebut menggunakan optimizer *adamax, batch size* 7 pada epoch 1000.
2. Pada penelitian ini berhasil dikembangkan metode untuk melakukan kuantifikasi otomatis yaitu dengan menggunakan model CNN YOLOV8. Dari hasil eksperimen, didapatkan model CNN YOLOV8 dengan nilai pricision 0.907 serta recall 0.958.

5.2. Saran

Adapun saran-saran dan masukan yang dapat digunakan sebagai referensi penelitian selanjutnya dengan topik yang sama yakni terkait geotagging.

1. Akurasi model ANN dapat diperbaiki apabila model tersebut dilatih ulang
2. Terdapat metode pengujian akurasi prediksi jarak yang lebih komperehensif
3. Dapat dilakukan penyempurnaan sistem yakni dengan menambahkan model CNN untuk melakukan deteksi objek. Sehingga pengguna tidak lagi perlu untuk menentukan salah satu titik pada objek.
4. Menambahkan anotasi bounding box pada buah yang tertutupi buah/objek lain.
5. Sistem ini dapat diterapkan sebagai *back end* pada antarmuka web dimana pengguna dapat melakukan pengambilan citra, pemilihan titik, memprediksi jarak serta mendeteksi buah kakao.

DAFTAR PUSTAKA

(n.d.). Retrieved from Federal Aviation Administration of United States Department of Transportation: https://www.faa.gov/uas

Abdoellah, D. S. (2021). *Analisis Kinerja dan Prospek Komoditas Kakao.* Jember, Jawa Timur: Pusat Penelitian Kopi dan Kakao Indonesia, PT Riset Perkebunan Nusantara.

Adam Paszke, S. G. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Machine Learning (cs.LG); Mathematical Software (cs.MS); Machine Learning (stat.ML)*.

Alexander Kirillov, E. M.-Y. (2023). Segment Anything. *Computer Vision and Pattern Recognition (cs.CV); Artificial Intelligence (cs.AI); Machine Learning (cs.LG)*.

Barima Yao Sadaiou Sabas, K. G. (2020). Cocoa Production and Forest Dynamics in Ivory Coast from 1985 to 2019. *Land 2020*.

Bradford W. Parkinson, J. J. (1966). *Global Positioning System: Theory and Applications.* Washington DC: American Institute of Aeronautics and Astronautics, Inc.

Chatterjee, S. (2018). Time Series Crop Monitoring of an Agricultural Scheme on a Plot Basis Using Complementary Remote Sensing .

Cle´ment Godard, O. M. (2019). Digging Into Self-Supervised Monocular Depth Estimation.

Dedi Atunggal, N. H. (2018). Developing Android Application for Precise Geotagging Using RTK GPS Module. *2018 4th International Conference on Science and Technology (ICST);2018;1; ;* . Yogyakarta.

Faisal Khan, S. S. (2020). Deep Learning-Based Monocular Depth Estimation Methods—A State-of-the-Art Review. *Sensors*.

Indonesia, D. J. (2021). *Laporan Tahunan 2021 Direktorat Jenderal Perkebunan Kementrian Pertanian Pengembangan Perkebunan 2021.*

Indonesia, D. J. (2021). *Statistik Perkebunan Unggulan Nasional 2020-2022.*

Krista Merry, P. B. (2019). Smartphone GPS accuracy study in an urban environment. *PLOS ONE*.

Meng Guo, J. L. (2017). A Review of Wetland Remote Sensing. *Special Issue "Understanding Land Surface Processes and Ecosystem Changes with Optical and Laser Remote Sensing"*.

Olaf Ronneberger, P. F. (n.d.). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany*.

Paul A. Longley, M. F. (2015). *Geographic Information Science and Systems.* Wiley.

R Neswati, L. A. (2019). Land Suitability for Cocoa Development in South Sulawesi: An Analysis using GIS and Parametric Approach. *The 4th International Conference of Indonesian Society for Remote Sensing.* IOP Conference Series: Earth and Environmental Science.

Rachita Byahatti, D. S. (2021). Object Detection and Classification using YOLOv3. *International Journal of Engineering Research & Technology (IJERT)*.

Roboflow, I. (2023). *roboflow*. Retrieved from https://roboflow.com

Saad Ashfaq, M. A. (2022). Accelerating Deep Learning Model Inference on Arm CPUs with Ultra-Low Bit Quantization and Runtime. *Machine Learning (cs.LG); Artificial Intelligence (cs.AI)*.

Sugiarto, A. C. (n.d.). Manajemen Produksi Cokelat Vicco Dark Premium 70 Gram Di Pusat Penelitian Kopi Dan Kakao Indonesia.

Vladimir A. Krylov, E. K. (2018). Automatic Discovery and Geotagging of Objects from Street View Imagery. *remote sensing MDPI*.

Wahyudi, T. P. (2008). *Panduan lengkap kakao manajemen agribisnis dari hulu hingga hilir* . Jakarta: Penebar Swadaya.

Xiaoming Fu, A. L. (2022). A Dynamic Detection Method for Phenotyping Pods in a Soybean Population Based on an Improved YOLO-v5 Network. *MDPI*.

Xiaoming Fu, A. L. (2022). A Dynamic Detection Method for Phenotyping Pods in a Soybean Population Based on an Improved YOLO-v5 Network. *MDPI Agronomy*.

LAMPIRAN

```
### Environment Setup & Architecture Config
import pandas as pd
from keras.models import Sequential, load_model
from keras.layers import Dense
import matplotlib.pyplot as plt
import datetime
from sklearn.model_selection import train_test_split
# Load the dataset from the CSV file
#   data   =   pd.read_csv('/Volumes/Sandisk   SSD/All
Skripsi/Skripsi/Dataset.csv')
# print(len(data))

# Split the dataset into input (X) and output (y)
variables
# X = data[['r', 'g', 'b']]
# y = data['distance']



# Create the neural network model
model = Sequential()
model.add(Dense(1000, input_dim=3, activation='relu'))
model.add(Dense(750, activation='sigmoid'))
model.add(Dense(500, activation='relu'))
model.add(Dense(300, activation='sigmoid'))
model.add(Dense(200, activation='relu'))
model.add(Dense(150, activation='sigmoid'))
model.add(Dense(100, activation='relu'))
model.add(Dense(50, activation='sigmoid'))
model.add(Dense(1, activation='linear'))
from tensorflow.keras.utils import plot_model
```

```
# Visualize the model
plot_model(model,                    to_file='model.png',
show_shapes=True, show_layer_names=True)
from PIL import Image
import matplotlib.pyplot as plt

# Load and display the image
image = Image.open('model.png')
plt.imshow(image)
plt.axis('off')
plt.show()

X.count
data
# TTS
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.04, random_state=20)
# Options
optimizer_selected = 'adamax'
bathsize_selected = 1
epoch_selected = 1000

model.compile(loss='mean_absolute_error',
optimizer=optimizer_selected,
metrics='mean_absolute_error')
# optimizer = adam, sgd, adamax, nadam
#      train      =      model.fit(X_train,      y_train,
epochs=epoch_selected,    batch_size=bathsize_selected,
validation_data=(X_test, y_test))
# Plot Result
plt.figure(figsize=(8,8))
plt.plot(train.history['loss'], linewidth=1.0)
plt.plot(train.history['val_loss'], linewidth=1.0)
```

```
plt.ylabel('Cross Entropy')


plt.ylabel('loss')
plt.xlabel('epoch')

plt.legend(['train','test'], loc='upper right')

current_time = datetime.datetime.now().strftime("%Y-%m-
%d-%H_%M_%S")
last_MAE                                                =
train.history['mean_absolute_error'][len(train.history[
'mean_absolute_error'])-1]
filename       =       f"/Volumes/Sandisk       SSD/All
Skripsi/Pengembangan     Model     ANN     RGB     to
Jarak/Models/local_{optimizer_selected}_{bathsize_selec
ted}_{current_time}_Last MAE={last_MAE}_epoch 5000"

plt.title('Model Loss')
description     =     f"Optimizer:{optimizer_selected},
Batchsize:{bathsize_selected},     Time:{current_time},
\nLast MAE={last_MAE}_epoch 5000"
plt.text(0.1,            0.98,            description,
transform=plt.gca().transAxes,            fontsize=10,
verticalalignment='top',   bbox=dict(facecolor='white',
edgecolor='gray', boxstyle='round,pad=0.5'))

# !!!!!!!!!!!!!!!!!!!
# !!!!!!!!!!!!!!!!!!!
# !!!!!!!!!!!!!!!!!!!
model.save(filename+".h5")
plt.savefig(filename+".png")
```

```
filename = "/Volumes/Sandisk SSD/All Skripsi/Pengembangan Model ANN RGB to Jarak/Models/local_sgd_15_2023-07-05-11_39_58_Last MAE=0.35471969842910767_epoch 3000"
loaded_model = load_model(filename+".h5")
optimizer_selected = 'sgd'
bathsize_selected = 15
epoch_selected = 2000
loaded_model.compile(loss='mean_absolute_error', optimizer=optimizer_selected, metrics='mean_absolute_error')
train = loaded_model.fit(X_train, y_train, epochs=epoch_selected, batch_size=bathsize_selected, validation_data=(X_test, y_test))
```

Lampiran 1 Kode Pelatihan Model ANN prediksi jarak menggunakan nilai RGB

```
# -*- coding: utf-8 -*-
"""train-yolov8-object-detection-on-custom-
dataset.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1N9sR5hrgWq-
M603xucAvXrMtvkbcu9Rk

Tutorial
https://docs.ultralytics.com/yolov5/tutorials/tips_for_
best_training_results/

dYf7lq3b9OrPADmDhQhcwSDpS

## Prep
"""

!pip install comet_ml
import comet_ml

comet_ml.init(project_name="yolov8")

from comet_ml import Experiment
from comet_ml.integration.pytorch import log_model

experiment = Experiment(
  api_key = "dYf7lq3b9OrPADmDhQhcwSDpS",
  project_name = "yolov8",
  workspace="agunggg"
)
```

```
"""## Before you start"""

!nvidia-smi

import os
HOME = os.getcwd()
print(HOME)
# Pip install method (recommended)

!pip install ultralytics==8.0.20

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()

from ultralytics import YOLO

from IPython.display import display, Image

!pip install roboflow --quiet
from roboflow import Roboflow

"""## Custom Training"""

# Commented out IPython magic to ensure Python
compatibility.
!mkdir {HOME}/datasets
# %cd {HOME}/datasets

rf = Roboflow(api_key="Dcl5orKoyjBh7DRAKSBU")
```

```
project = rf.workspace("gungs").project("yolo-cocoa")
dataset = project.version(2).download("yolov5")

import os

def count_files(directory):
    file_count = 0

    # Iterate through all the items in the directory
    for item in os.listdir(directory):
        item_path = os.path.join(directory, item)

        # Check if the current item is a file
        if os.path.isfile(item_path):
            file_count += 1

    return file_count

# Provide the directory path you want to count files in
directory_path = '/content/datasets/YOLO-cocoa-2/valid/images/'
file_count = count_files(directory_path)
print("Total files in the directory:", file_count)

"""## TRAIN"""

# Commented out IPython magic to ensure Python compatibility.
# %cd {HOME}

!yolo task=detect mode=train model= yolov8m.pt data={dataset.location}/data.yaml epochs=700 imgsz=800 plots=True patience=0
```

```
"""### train results"""

result_folder = '/content/runs/detect/train/'

!ls {HOME}/runs/detect/train/

# Commented out IPython magic to ensure Python
compatibility.
# %cd {HOME}
Image(filename=f"{result_folder}"+"/confusion_matrix.pn
g", width=600)

# Commented out IPython magic to ensure Python
compatibility.
# %cd {HOME}
Image(filename=f"{result_folder}"+"/results.png",
width=600)

# Commented out IPython magic to ensure Python
compatibility.
# %cd {HOME}
Image(filename=f"{result_folder}"+"/val_batch0_pred.jpg
", width=600)

"""## Inference with Custom Model"""

result_folder

# Commented out IPython magic to ensure Python
compatibility.
# %cd {HOME}
```

```
test = !yolo task=detect mode=predict
model={HOME}/runs/detect/train/weights/best.pt
conf=0.25 source=/content/test save=True
test

"""**NOTE:** Let's take a look at few results."""

import glob
from IPython.display import Image, display

for image_path in
glob.glob(f'{HOME}/runs/detect/predict/*.jpg'):
      display(Image(filename=image_path, width=600))
      print("\n")

"""### Download Zipped model"""

from google.colab import files
import shutil
import os

folder_path = '/content/runs/'

# Create a zip file of the folder
shutil.make_archive(folder_path, 'zip', folder_path)

# Download the zip file
files.download(folder_path+'.zip')

files.download('/content/yolov8m.pt')

from ultralytics import YOLO
```

```
# Load a model
# model = YOLO('yolov8n.pt')  # load an official model
model                                                    =
YOLO('/content/runs/detect/train/weights/best.pt')    #
load a custom model

# Predict with the model
img          =          '/content/datasets/YOLO-cocoa-
2/valid/images/IMG_20230621_125352_jpg.rf.a01376e8cc4b2
0ef124622299f340d11.jpg'
results = model(img)  # predict on an image

from google.colab.patches import cv2_imshow

res = model(img)
res_plotted = results[0]
cv2_imshow(res_plotted)
```

Lampiran 2 Kode pelatihan YOLOV8 untuk deteksi Buah Kakao

```
# -*- coding: utf-8 -*-
"""Final_System refactored.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1GorMCVpZ8aPCDyns5Ja73SH0KpiWCVDq

# Prep all packages
"""

!git clone https://github.com/danielyoga/Models_Skripsi_2023.git

import os
HOME = os.getcwd()
print(HOME)
# Pip install method (recommended)

!pip install ultralytics==8.0.20

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()

from ultralytics import YOLO

from IPython.display import display, Image
```

```
!pip install roboflow --quiet
from roboflow import Roboflow

!git clone https://github.com/nianticlabs/monodepth2.git
!python -m pip install opencv-python matplotlib onnx onnxruntime
!python -m pip install 'git+https://github.com/facebookresearch/segment-anything.git'
!wget https://dl.fbaipublicfiles.com/segment_anything/sam_vit_h_4b8939.pth

import cv2, numpy as np, glob, subprocess, os, matplotlib.pyplot as plt, math, torch, onnxruntime
from google.colab import drive, files
from PIL import Image
from PIL import Image

from onnxruntime.quantization import QuantType
from onnxruntime.quantization.quantize import quantize_dynamic
from segment_anything import sam_model_registry, SamPredictor
from segment_anything.utils.onnx import SamOnnxModel

from keras.models import load_model
loaded_model = load_model("/content/Models_Skripsi_2023/local_adamax_1_2023-06-26-10_18_29_Last MAE=0.24479460716247559_epoch 5000.h5")
```

```
onnx_model_path =
"/content/Models_Skripsi_2023/sam_onnx_quantized_exampl
e.onnx"
checkpoint = "sam_vit_h_4b8939.pth"
model_type = "vit_h"

sam =
sam_model_registry[model_type](checkpoint=checkpoint)

def show_mask(mask, ax):
    color = np.array([30/255, 144/255, 255/255, 0.6])
    h, w = mask.shape[-2:]
    mask_image = mask.reshape(h, w, 1) *
color.reshape(1, 1, -1)
    ax.imshow(mask_image)

def show_points(coords, labels, ax, marker_size=375):
    pos_points = coords[labels==1]
    neg_points = coords[labels==0]
    ax.scatter(pos_points[:, 0], pos_points[:, 1],
color='green', marker='*', s=marker_size,
edgecolor='white', linewidth=1.25)
    ax.scatter(neg_points[:, 0], neg_points[:, 1],
color='red', marker='*', s=marker_size,
edgecolor='white', linewidth=1.25)

"""# 1. Prediksi Jarak

## Depth Estimation
"""

# Commented out IPython magic to ensure Python
compatibility.
```

```
# Execute Process

# Upload File
uploaded_file = files.upload()
filename = next(iter(uploaded_file))
file_content = uploaded_file[filename]

with open(filename, 'wb') as f:
    f.write(file_content)
    print(f"File '{filename}' uploaded and saved.")

# Estimate Depth
#    %run    monodepth2/test_simple.py    --image_path
"/content/"{filename} --model_name mono_1024x320

"""## Pilih Titik di Objek"""

# Show image with axis
image = cv2.imread(filename)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(10,10))
plt.imshow(image)
plt.axis('on')
plt.show()

image_disp    =    cv2.imread(filename.replace(".jpg",
"_disp.jpeg"))

plt.figure(figsize=(10,10))
plt.imshow(image_disp)
plt.axis('on')
plt.show()
```

```
# Check titik yang dipilih
input_point = np.array([[2700, 2900]])
input_label = np.array([1])

plt.figure(figsize=(10,10))
plt.imshow(image)
show_points(input_point, input_label, plt.gca())
plt.axis('on')
plt.show()

"""## Segment Object Area"""

ort_session                                                    =
onnxruntime.InferenceSession(onnx_model_path)
sam.to(device='cuda')
predictor = SamPredictor(sam)

predictor.set_image(image)
image_embedding                                                =
predictor.get_image_embedding().cpu().numpy()

onnx_coord = np.concatenate([input_point, [[0.0, 0.0]]],
axis=0)[None, :, :]
onnx_label    =    np.concatenate([input_label,    [-1]],
axis=0)[None, :].astype(np.float32)

onnx_coord                                                     =
predictor.transform.apply_coords(onnx_coord,
image.shape[:2]).astype(np.float32)

ort_inputs = {
    "image_embeddings": image_embedding,
```

```
    "point_coords": onnx_coord,
    "point_labels": onnx_label,
    "mask_input":    np.zeros((1,    1,    256,    256),
dtype=np.float32),
    "has_mask_input": np.zeros(1, dtype=np.float32),
    "orig_im_size":           np.array(image.shape[:2],
dtype=np.float32)
}

masks,   _,   low_res_logits   =   ort_session.run(None,
ort_inputs)
masks = masks > predictor.model.mask_threshold

plt.figure(figsize=(10,10))
plt.imshow(image)
show_mask(masks, plt.gca())
show_points(input_point, input_label, plt.gca())
plt.axis('on')
plt.show()

"""## get median pada segment area on disp image"""

converted_array = masks.astype(int)
positions = np.where(converted_array == 1)
x_coords = positions[2]
y_coords = positions[3]
pixel_positions = np.column_stack((x_coords, y_coords))

data   =   np.array(cv2.imread(filename.replace(".jpg",
"_disp.jpeg")))
#  data  =  np.array(Image.open(filename.replace(".jpg",
"_disp.jpeg")))
```

```
# Get pixel values at the specified positions
pixel_values = data[pixel_positions[:, 0],
pixel_positions[:, 1]]

median = np.median(pixel_values, axis=0)

print("Median:", median)

"""## median to jarak"""

# Execute process
# predict jarak using median rgb
median = median.reshape(1, -1)
predictions = loaded_model.predict(median)
predictions = predictions[0][0].tolist()
predictions

predicted_distance = predictions
predicted_distance

"""# 2. Perhitungan Koordinat

## Get Camera Long Lat
"""

image = Image.open(filename)
exif_data = image._getexif()

gps_info = exif_data.get(34853)

direction = gps_info[17]

latitude = gps_info.get(2)
```

```
longitude = gps_info.get(4)

latitude_decimal = float(latitude[0] + latitude[1] / 60
+ latitude[2] / 3600)
longitude_decimal = float(longitude[0] + longitude[1] /
60 + longitude[2] / 3600)

print(latitude_decimal, longitude_decimal, direction)

lat1 = latitude
lon1 = longitude

"""## Count Coordinate"""

# Constants
R = 6371.0  # Earth's radius in kilometers

# Starting point
latlong = [latitude_decimal, longitude_decimal]
lat1 = math.radians(latlong[0])
lon1 = math.radians(latlong[1])

# Heading in degrees
heading = math.radians(direction)

# Distance in kilometers
distance = predictions/1000

# Calculate destination point
lat2 = math.asin(math.sin(lat1) * math.cos(distance/R)
+
                 math.cos(lat1) * math.sin(distance/R)
* math.cos(heading))
```

```
lon2 = lon1 + math.atan2(math.sin(heading) *
math.sin(distance/R) * math.cos(lat1),
                         math.cos(distance/R)         -
math.sin(lat1) * math.sin(lat2))

# Convert back to degrees
lat2 = math.degrees(lat2)
lon2 = math.degrees(lon2)

# Display the destination coordinates
print(f"Destination: {lat2}, {lon2}")

"""# 3. Deteksi Buah Kakao"""

# If not found, create predict and predict result folder
import os
import shutil

if not os.path.exists("/content/predict"):
    os.makedirs("/content/predict")
else :
    file_list = os.listdir("/content/predict")
    for file_name in file_list:
      file_path   =   os.path.join("/content/predict",
file_name)

      if os.path.isfile(file_path):
          os.remove(file_path)

# Add the current image to the "predict" folder
image_name = os.path.basename(filename)
predict_image_path  =  os.path.join("/content/predict",
image_name)
```

```
shutil.copy(image_name, predict_image_path)

# Commented out IPython magic to ensure Python
compatibility.
# %cd {HOME}
test = !yolo task=detect mode=predict
model='/content/Models_Skripsi_2023/best.pt' conf=0.25
source=/content/predict/ save=True
test

import re

for i in test:
  match = re.search(r'(?<=to\s).*', i)
  if match:
      result = match.group()
      clean_string = re.sub("\x1b\[.*?m", "", result)
      print(clean_string)

num_unripecocoas = num_ripecocoas = 0

for element in test:
    if "unripecocoas" in element:
        ripecocoas_index =
element.index("unripecocoas")
        num_unripecocoas = int(element[ripecocoas_index
- 3:ripecocoas_index].strip())
        print("Terdeteksi ",num_ripecocoas," Buah Kakao
Belum Matang")

    if "ripecocoas" in element :
        ripecocoas_index = element.index("ripecocoas")
```

```
        num_ripecocoas = int(element[ripecocoas_index -
3:ripecocoas_index].strip())
        print("Terdeteksi ",num_ripecocoas," Buah Kakao
Matang")

# Show image with axis
image     =     cv2.imread(str(os.path.join("/content/",
clean_string, filename)))
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(10,10))
plt.imshow(image)
plt.axis('on')
plt.show()

"""### Initiate Dataframe"""

import pandas as pd

# Column names for the new DataFrame
column_names = ['predicted_distance',
                'lat1',
                'lon1',
                'lat2',
                'lon2',
                'num_unripecocoas',
                'num_ripecocoas']

# Creating an empty DataFrame with the specified column
names
geotagging_database                                    =
pd.DataFrame(columns=column_names)
```

```
# Displaying the newly created DataFrame
print(geotagging_database)

"""### Append to dataframe"""

data_to_append = [[predicted_distance,
                   lat1,
                   lon1,
                   lat2,
                   lon2,
                   num_unripecocoas,
                   num_ripecocoas]]

geotagging_database.append(pd.DataFrame(data_to_append,
columns=column_names), ignore_index=True)

geotagging_database                                    =
geotagging_database.append(pd.DataFrame(data_to_append,
columns=column_names), ignore_index=True)

geotagging_database

"""# Check GPU Available Resources"""

if torch.cuda.is_available():
    gpu_stats = torch.cuda.memory_stats()
    total_memory                                       =
gpu_stats["allocated_bytes.all.current"] / (1024 ** 3)
    max_memory  =  gpu_stats["allocated_bytes.all.peak"]
/ (1024 ** 3)
    available_memory = max_memory - total_memory
    print(f"Available            GPU            memory:
{available_memory:.2f} GB")
```

```
else:
    print("No GPU available.")

"""# Pembuktian Rumus"""

import math
import pandas as pd

# Starting coordinates
def calculate_destination(latlong, heading, distance):
    R = 6371  # Earth's radius in kilometers

    # Starting point
    lat1 = math.radians(latlong[0])
    lon1 = math.radians(latlong[1])

    # Heading in degrees
    heading = math.radians(heading)

    # Distance in kilometers
    # distance = predictions

    # Calculate destination point
    lat2       =       math.asin(math.sin(lat1)       *
math.cos(distance/R) +
                     math.cos(lat1)                    *
math.sin(distance/R) * math.cos(heading))
    lon2   =   lon1   +   math.atan2(math.sin(heading)   *
math.sin(distance/R) * math.cos(lat1),
                               math.cos(distance/R)     -
math.sin(lat1) * math.sin(lat2))

    # Convert back to degrees
```

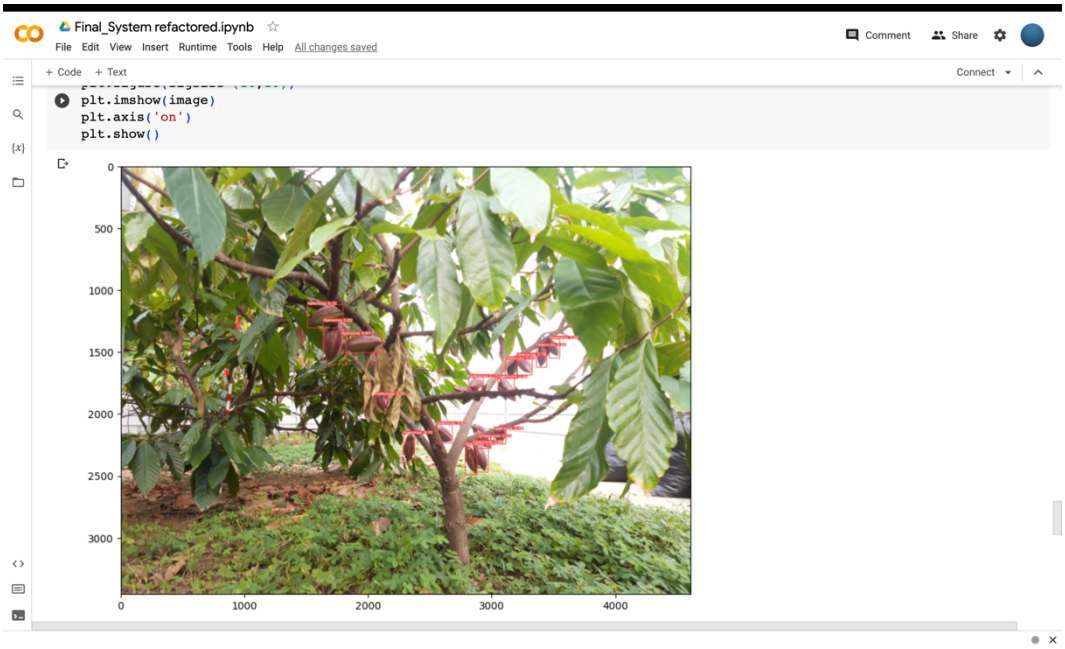
```
    lat2 = math.degrees(lat2)
    lon2 = math.degrees(lon2)

    # Return the destination coordinates
    return lat2, lon2

calculate_destination([-7.94984,   112.615411], 40.456,
0.069)
```

Lampiran 3 Kode Sistem Keseluruhan

Lampiran 4 Dataframe Sebagai Penyimpanan Sementara Data Hasil Prediksi Koordinat



Lampiran 5 Tangkapan Layar Hasil Deteksi Buah Kakao