PENGARUH *BATCH SIZE* PADA *TRAINING* AKURASI ALOGIRTMA CNN (STUDI KASUS SISTEM PAKAR TANAMAN KENTANG)

SKRIPSI

Diajukan sebagai Salah Satu Syarat Untuk Mendapatkan
Gelar Sarjana Komputer (S.Kom) Program Studi Informatika

RESKY UTAMI
105841107319

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAKASSAR
2023

PENGARUH BATCH SIZE PADA TRAINING AKURASI ALGORITMA CNN (STUDI KASUS SISTEM PAKAR TANAMAN KENTANG)

Diajukan Untuk Memenuhi Salah Satu Syarat Guna Memperoleh
Gelar Sarjana Komputer Prodi Informatika Fakultas Teknik
Universitas Muhammadiyah Makassar

Disusun Dan Diajukan Oleh:

RESKY UTAMI
105841107319

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAKASSAR
2023

UNIVERSITAS MUHAMMADIYAH MAKASSAR

FAKULTAS TEKNIK

GEDUNG MENARA IQRA LT. 3

Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221

Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com

Website: http://teknik.unismuh.makassar.ac.id

بسم الله الرحمن الرحيم

PENGESAHAN

Skripsi atas nama **Resky Utami** dengan nomor induk Mahasiswa **105841107319**, dinyatakan diterima dan disahkan oleh Panitia Ujian Tugas Akhir/Skripsi sesuai dengan Surat Keputusan Dekan Fakultas Teknik Universitas Muhammadiyah Makassar Nomor : 405/05/A.5-II/VIII/45/2023, sebagai salah satu syarat guna memperoleh gelar Sarjana Komputer pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar pada hari Kamis tanggal 31 Agustus 2023.

Makassar, 16 Safar 1444 H / 02 September 2023 M

Panitia Ujian :

1. Pengawas Umum
   a. Rektor Universitas Muhammadiyah Makassar
      Prof. Dr. H. Ambo Asse, M.Ag. : .....
   b. Dekan Fakultas Teknik Universitas Hasanuddin
      Prof. Dr. Eng. Muhammad Isran Ramli, ST., MT. : .....
2. Penguji
   a. Ketua : Dr. Ir. Zahir Zainuddin, M.Sc. : .....
   b. Sekertaris : Lukman Anas, S.Kom., MT. : .....
3. Anggota : 1. Muhyiddin A.M Hayat, S.Kom., MT. : .....
   2. Rizki Yusliana Bakti, ST., MT. : .....
   3. Lukman, S.Kom., M.T. : .....

Mengetahui :

Pembimbing I	Pembimbing II
Titin Wahyuni, S.Pd., MT.	Fahrim Irhamna Rahman, S.Kom., MT.

Dekan Fakultas Teknik

Dr. Ir. Hj. Nurnawaty, ST., MT., IPM.

NBM : 795 108

UNIVERSITAS MUHAMMADIYAH MAKASSAR

FAKULTAS TEKNIK

GEDUNG MENARA IQRA LT. 3

Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: http://teknik.unismuh.makassar.ac.id

بسم الله الرحمن الرحيم

HALAMAN PERSETUJUAN

Tugas Akhir ini diajukan untuk memenuhi syarat ujian guna memperoleh gelar Sarjana Komputer (S.Kom) Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar.

Judul Skripsi : **PENGARUH BATCH SIZE PADA TRAINING AKURASI ALGORITMA CNN (STUDI KASUS SISTEM PAKAR TANAMAN KENTANG)**

Nama : **Resky Utami**

Stambuk : 10584 11073 19

Makassar, 02 September 2023

Telah Diperiksa dan Disetujui
Oleh Dosen Pembimbing;

Pembimbing I	Pembimbing II
Titin Wahyuni, S.Pd.,MT.	**Fahrim Irhamna Rahman, S.Kom., MT.**

Mengetahui,
Ketua Program Studi Informatika

Muhyidin A.M Hayat,S.kom,M.T
NBM : -

MOTTO DAN PERSEMBAHAN

" Hatiku tenang karena mengetahui bahwa apa yang melewatkanku tidak akan pernah menjadi takdirku, dan apa yang ditakdirkan untukku tidak akan pernah melewatkanku."

-Umar bin Khattab-

Skripsi ini ku persembahkan untuk

Orang tuaku tersayang <u>Ayahanda Azis dan Ibunda Nurmala</u>, yang tak pernah lelah meneteskan keringat, air mata, dan merangkai do'a untukku. Keikhlasan dan kesabaran yang telah mendidik dan membimbing ananda dari kecil hingga dewasa, dan kepada beliau semoga Allah SubhanahuWaTa'Ala meridhoi segala amal ibadah dan dilipat gandakan-Nya.

ABSTRAK

Penelitian ini bertujuan untuk mengidentifikasi penyakit pada tanaman kentang dengan memanfaatkan data gambar untuk mengidentifikasi penyakit pada tanaman kentang menggunakan metode *Convolutional Neural Network* (CNN). Pengambilan data dilakukan di lahan pertanian Balai Penyuluhan Pertanian (BPP) Desa Kanreapia di Kecamatan Tombolo Pao, Kabupaten Gowa. Daerah ini merupakan daerah agraris dengan mayoritas penduduknya bercocok tanam, terutama tanaman kentang. Penelitian ini berfokus pada efek dari variasi *batch size* dalam proses pelatihan model CNN terhadap akurasi identifikasi penyakit pada tanaman kentang dengan Penggunaan teknologi *image processing* atau biasa disebut pengolahan citra digital. Penelitian ini memperhatikan faktor-faktor seperti ukuran *batch* pada proses pelatihan model, serta pembagian data menjadi bagian pelatihan dan pengujian. Melalui percobaan dengan variasi *batch size* dan *split* data, penelitian ini bertujuan untuk menemukan konfigurasi optimal yang dapat menghasilkan akurasi terbaik dalam identifikasi penyakit pada tanaman kentang. Berdasarkan hasil yang diperoleh setelah melakukan beberapa percobaan *batch size* dan *split* data, *batch size* 32 dan *split* data 80:10:10 menunjukkan nilai akurasi terbaik yaitu sekitar 98,50%.

Kata Kunci :*Batch Size, Convolutional Neural Network, Image Processing,* Penyakit Kentang*, Split Data*

ABSTRACT

This research aims to identify diseases in potato plants by utilizing image data for disease identification using the Convolutional Neural Network (CNN) method. Data collection was conducted in the agricultural land of Balai Penyuluhan Pertanian (BPP) in Kanreapia Village, Tombolo Pao District, Gowa Regency. This region is an agrarian area with the majority of the population engaged in farming activities, particularly potato cultivation. The study focuses on the effect of varying batch sizes in the CNN model training process on the accuracy of disease identification in potato plants, employing image processing technology. The research takes into consideration factors such as batch size in the model training process, as well as the division of data into training and testing sets. Through experiments involving variations in batch size and data split, the study aims to determine the optimal configuration that can yield the best accuracy in disease identification in potato plants. Based on the results obtained after conducting several experiments with different batch sizes and data splits, a batch size of 32 and a data split of 80:10:10 demonstrate the highest accuracy value, approximately 98.50%.

***Keywords**: Batch Size, Convolutional Neural Network, Image Processing, Potato Disease, Data Split*

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيم

Puji dan syukur Alhamdulilah penulis panjatkan kehadirat Allah Subhanahu Wa Ta'ala atas segala limpahan Rahmat dan Magfirah-Nya, sehingga penulis dapat menyelesaikan penyususnan skripsi yang berjudul " Pengaruh *Batch size* Pada *Training* Akurasi Algoritma CNN (Studi Kasus Sistem Pakar Tanaman Kentang) ". Tak lupa pula shalawat serta salam penulis haturkan kepada Rasulullah Nabi Muhammad Sallallahu 'Alaihi Wasallam yang senantiasa kita tunggu-tunggu syafaatnya kelak di hari akhir nanti. Skripsi yang penulis buat ini bertujuan untuk memenuhi syarat dalam menyelesaikan Program Sarjana (S1) pada Fakultas Teknik Universitas Muhammadiyah Makassar.

Dalam proses penulisan sampai dengan terselesaikannya skripsi ini, tentunya banyak sekali pihak yang berkontribusi di dalamnya. Penghargaan yang setinggi-tingginya dan terimakasih banyak penulis haturkan dengan hormat kepada:

1. Kepada cinta pertama dan panutanku, Ayahanda Azis dan Pintu Surgaku, Ibunda Nurmala yang penulis sayangi, dengan setulus hati dan Ikhlas telah mendidik dan memberikan pengorbanan yang tak ternilai, dorongan moril dan materil serta do'a dan cinta yang selama ini diberikan kepada penulis hingga mampu menyelesaikan studi sampai sarjana.
2. Bapak Prof. Dr. H. Ambo Asse, M.Ag., sebagai Rektor Perguruan Tinggi Universitas Muhammadiyah Makassar.
3. Ibu Dr.Ir.Hj. Nurnawaty, ST., M.T., IPM., sebagai Dekan Fakultas Teknik Universitas Muhammadiyah Makassar.
4. Bapak Muhyiddin AM Hayat, S.Kom., M.T., selaku Ketua Prodi Informatika, Fakultas Teknik Universitas Muhammadiyah Makassar.
5. Ibu Titin Wahyuni, S.Pd., MT selaku Dosen Pembimbing I dan Bapak Fahrim Irhamna Rachman, S.Kom., MT., selaku Dosen Pembimbing II yang

senantiasa meluangkan waktu dan pikirannya untuk membimbing dan mengarahkan penulis dalam penyusunan skripsi ini.

6. Ibu Sahriah, SP. selaku Ka. UPT BPP Kanreapia Kab. Gowa yang bersedia untuk mengizinkan penulis melakukan penelitian.
7. Bapak/Ibu Dosen dan Staff Administrasi Prodi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar.
8. Untuk kedua saudari penulis, Hasniati dan Magfirah Nur Rohmah yang selalu memberi dukungan dan mendo'akan penulis.
9. Untuk Mumu kucing peliharaan yang penulis sayangi, yang selalu menemani dan mewarnai hari-hari penulis dan menjadi bukti nyata bahwa obat tidak selalu berbentuk pil. Semoga senang di tempat barumu.
10. Untuk teman terbaik penulis, pemilik nim 105841100919 dan 105841110319 yang selalu menemani dan membantu hingga skripsi ini dapat penulis selesaikan.
11. Saudara/Saudari kami di Fakultas Teknik Koordinat 2019 yang selalu belajar dan berjuang bersama banyak membantu serta memberi dukungan dalam menyelesaikan tugas skripsi ini.

Peneliti berharap semoga skripsi ini bisa bermanfaat dan memberikan tambahan pengetahuan dan wawasan yang semakin luas kepada pembaca.

"Billahi Fii Sabilil Haq Fastabiqul Khaerat"

Makassar, 18 Agustus 2023

Resky Utami

DAFTAR ISI

HALAMAN PENGESAHAN iii
HALAMAN PERSETUJUAN iv
HALAMAN PERSEMBAHAN v
ABSTRAK vi
ABSTRACT vii
KATA PENGANTAR viii
DAFTAR ISI x
DAFTAR GAMBAR xii
DAFTAR TABEL xiii
DAFTAR LAMPIRAN xiv
BAB I PENDAHULUAN 1
A. Latar Belakang Masalah 1
B. Rumusan Masalah 2
C. Tujuan Penelitian 2
D. Manfaat Penelitian 3
E. Ruang Lingkup Penelitian 3
F. Sistematika Penulisan 3
BAB II TINJAUAN PUSTAKA 5
A. Landasan Teori 5
1. Sistem pakar 5
2. Pendeteksi Citra 6
3. Tanaman Kentang 7
4. Penyakit Tanaman Kentang 7
5. Data 11
6. *Python* 12
7. *Convolutional Neural Network* (CNN) 12
8. *Library Python* 14
9. *Splitting Data* 14
10. *Batch size* 15
11. *Epoch* 17
B. Penelitian Terkait 17
C. Kerangka Pikir 20
BAB III METODE PENELITIAN 21
A. Tempat Dan Waktu Penelitian 21
B. Alat Dan Bahan 21

C. Perancangan Sistem 22
D. Teknik Pengujian 26
1. Pengujian *Confusion Matrix* 26
E. Teknik Analisis Data 27
BAB IV HASIL DAN PEMBAHASAN 30
A. Pengumpulan data 30
B. Memuat Dataset 34
C. Splitting Data 36
D. Model Convolutional Neural Network (CNN) 38
E. Pelatihan model dengan metode *Convolutional Neural Network* (CNN) .. 42
F. Hasil Pelatihan Model 47
G. Pengujian menggunakan *Confusion Matrix* 54
BAB V PENUTUP 56
A. Kesimpulan 56
B. Saran 56
DAFTAR PUSTAKA 57
LAMPIRAN 60

DAFTAR GAMBAR

Gambar 1.Daun *Late blight* (Rakhmawati et al., 2018). 8
Gambar 2. Daun *Early Blight* (Rakhmawati et al., 2018). 8
Gambar 3. Gejala busuk cincin bakteri yang disebabkan oleh *Clavibacter sepedonicus* pada umbi kentang (Aisyah et al., 2022). 9
Gambar 4. Umbi kentang yang terserang penyakit *Bacterial soft rot* (Margareth et al., 2018). 10
Gambar 5. Umbi kentang yang terserang penyakit kudis bakteri (a) Ringan, (b) Sedang dan (c) Parah (Margareth et al., 2018) 10
Gambar 6. *Convolutional Neural Network* 13
Gambar 7. Kerangka Pikir 20
Gambar 8. *Flowchart system* 22
Gambar 9. *Confusion matrix* (Rabbani et al., 2021) 26
Gambar 10. Hasil pengumpulan Dataset 31
Gambar 11. Hasil run fungsi *classes* 35
Gambar 12. Hasil Source code untuk menampilkan dataset 36
Gambar 13. Hasil arsitektur model CNN 41
Gambar 14. Hasil dari proses training selama 10 *epoch* 48
Gambar 15. Grafik hasil training dataset 49
Gambar 16. Hasil pengujian *Confusion Matrix* 54

DAFTAR TABEL

Table 1. *Persentase* penyakit tanaman kentang 10
Table 2. Dataset penyakit tanaman kentang. 31
Table 3. Jumlah dataset masing-masing penyakit tanaman kentang. 34
Table 4. Hasil percobaan perbandingan. 45
Table 5. Uji coba prediksi gambar 49

DAFTAR LAMPIRAN

Lampiran 1. Nama penyakit tanaman kentang..... 61
Lampiran 2. Jumlah dataset setiap penyaki..... 63
Lampiran 3. Hasil Pengumpulan Dataset..... 64
Lampiran 4. Hasil tampilan visual dari *batch* gambar. 65
Lampiran 5. Hasil ringkasan arsitektur model CNN..... 66
Lampiran 6. Hasil prediksi dan *confidence* kelas..... 67
Lampiran 7. Hasil pengujian *batch size* 16 *split* data 80:10:10 67
Lampiran 8. Grafik akurasi pengujian *batch size* 16 *split* data 80:10:10..... 68
Lampiran 9. Hasil pengujian *batch size* 16 *split* data 70:15:15 68
Lampiran 10. Grafik akurasi pengujian *batch size* 16 *split* data 70:15:15..... 68
Lampiran 11. *Source code*..... 73
Lampiran 12. Surat Keterangan Bebas Plagiasi..... 78
Lampiran 13. Hasil Scan Plagiasi Per Bab..... 79

BAB I
PENDAHULUAN

A. Latar Belakang Masalah

Kentang merupakan salah satu tanaman dikotil dari keluarga *Solanaceae*. Tanaman kentang adalah tanaman yang dapat berkembang biak secara vegetatif melalui umbi. Tanaman kentang akan tumbuh subur pada daerah yang beriklim dingin (Fuadi & Suharso, 2022). Tanaman kentang juga menjadi salah satu tanaman yang rentan terkena penyakit. Penyakit pada tanaman ketang dapat menyerang bagian daun kentang, batang, dan umbi kentang yang disebabkan oleh hama, virus, maupun bakteri. Kendala yang sering kali di jumpai adalah tidak semua petani memiki pengetahun tentang cara pengenalan gejala penyakit ini secara menyeluruh.

Dalam era teknologi sekarang ini, pemanfaatan teknologi informasi sudah banyak dijumpai di segala bidang, bidang pendidikan, bidang pemerintahan, bidang industri, bidang kesehatan dan termasuk bidang pertanian. Salah satu pemanfaatan bidang informatika dalam mengidentifikasi penyakit yang ada pada tanaman kentang yaitu dengan menggunakan *image processing* atau biasa disebut pengolahan citra digital. *Image processing* melakukan identifikasi yang dapat membantu para petani untuk memberikan penanganan secara efektif dan efisien pada tanaman yang terserang penyakit atau tidak normal. Dalam perkembangan teknologi saat ini telah banyak penelitian yang mengembangkan pengolahan citra digital dalam bidang pertanian (Rozaqi et al., 2021).

Desa Kanreapia yang berada di Kecamatan Tombolo Pao Kabupaten Gowa merupakan daerah agraris karena sebagian besar penduduk bermata pencaharian sebagai petani atau bercocok tanam seperti sayur mayur, umbi-umbian dan lain sebagainya (T. S. Dewi et al., 2020). Salah satu tanaman yang banyak dijumpai di Desa Kanreapia adalah tanaman kentang.

Lahan pertanian kentang organik milik Balai Penyuluhan Pertanian (BPP) Desa Kanreapia. Dengan luas 2 hekatare merupakan pusat pengembangan dan uji

coba produk pertanian, dengan melakukan pengujian pupuk *Biota Plus* pada tanaman kentang. Meskipun menjadi pusat pengembangan dan uji coba produk pertanian, tanaman kentang yang ada di lahan BPP Kanreapia belum maksimal dalam pencegahan terhadap penyakit dan hama. Saat ini karna cuaca yang tidak menentu tanaman kentang rentan terkena penyakit busuk daun jika terjadi hujan lebat yang terjadi secara terus-menerus. Sebaliknya jika musim kemarau berlangsung lama, tanaman kentang akan mengalami kerusakan akibat serangan hama, jika dibiarkan kerusakan dapat menular ke tangkai dan umbi yang dapat menyebabkan kerugian para petani saat masa panen.

Melihat dari kondisi yang terjadi di lahan pertanian BPP Kanreapia, menjadi acuan peneliti untuk membuat penelitian berdasarkan studi kasus pada tanaman kentang. Penelitian ini bertujuan untuk mengidentifikasi penyakit pada tanaman kentang dengan memanfaatkan data gambar tanaman kentang berdasarkan gejala-gejala yang muncul yang nantinya akan didapatkan akurasi dari penyakit tanaman kentang, untuk menentukan mana akurasi yang baik maka dilakukan beberapa percobaan *batch size* dan *split* data. Pada penelitian ini akan menggunakan algoritma *Convolutional Neural Network* (CNN) yang merupakan salah satu dari metode *Deep Learning*. Oleh karena itu peneliti mengambil judul " PENGARUH *BATCH SIZE* PADA *TRAINING* AKURASI ALOGIRTMA CNN (STUDI KASUS SISTEM PAKAR TANAMAN KENTANG) ".

B. Rumusan Masalah

Berdasarkan identifikasi masalah di atas, maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana akurasi yang didapatkan dari *batch size* 16, 32, 64, 128 pada studi kasus tanaman kentang?
2. Diantara *splitting* data 80:10:10 dan 70:15:15, *splitting* data mana yang mendapat akurasi paling baik dalam studi kasus tanaman kentang?

C. Tujuan Penelitian

Adapun tujuan penelitian berdasarkan rumusan masalah yang sudah ditetapkan, maka tujuan dari penelitian ini adalah:

1. Mengatahu bagaimana akurasi yang didapatkan dari *batch size* 16, 32, 64, 128 pada studi kasus tanaman kentang.
2. Mengetahui *splitting* data 80:10:10 dan 70:15:15 mana yang mendapat akurasi paling baik dalam studi kasus tanaman kentang.

D. Manfaat Penelitian

Dapat menambah pengetahuan dan wawasan penulis mengenai pentingnya pemilihan *batch size* dan *splitting* data untuk model CNN yang dapat menghasilkan akurasi yang baik, serta menjadi salah satu syarat kelulusan di Universitas Muhammadiyah Makassar.

E. Ruang Lingkup Penelitian

Agar pembahasan pada penelitian tidak meluas maka penulis membatasi permasalahan yang disesuaikan dengan kemampuan yang ada, diantaranya:

1. Data yang diambil berdasar dari hasil interview dan pengamatan yang di lakukan dengan kepala cabang dinas pertanian Tombolo Pao dan petani yang bekerja di lahan BPP Kanreapia.
2. Jenis citra yang diidentifikasi adalah citra daun, tangkai, dan umbi kentang.
3. Penelitian berfokus pada klasifikasi penyakit yang dilihat berdasarkan gejala pada tanaman kentang.
4. Penelitian berfokus pada *batch size* dan *splitting data*.
5. *Batch size* yang digunakan pada penelitian ini yaitu 16,32,64, dan 128. Untuk proses *splitting data* yaitu 80:10:10 dan 70:15:15.

F. Sistematika Penulisan

Sistematika penulisan ini dibuat untuk mempermudah dalam penyusunan proposal penelitian ini maka dari itu perlu ditentukan sistematika penulisan proposal yang baik. Sistematika penulisannya adalah sebagai berikut:

BAB I **: PENDAHULUAN,** Bab ini mendeskripsikan tentang latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan pada penelitian yang dilakukan.

BAB II **: TINJAUAN PUSTAKA,** Bab ini berisi mengenai pembahasan tentang penelitian yang terkait, landasan teori, serta kerangka fikir.

BAB III **: METODE PENELITIAN,** Bab ini memberikan gambaran tentang penelitian yang dilakukan dan penjelasan tentang apa yang dilakukan dalam penelitian ini.

BAB IV **: HASIL DAN PEMBAHASAN,** Bab ini berisi tentang penguraian dari hasil penelitian.

BAB V **: PENUTUP,** Bab ini berisi kesimpulan dan saran dari akhir penulisan sebuah skripsi.

BAB II
TINJAUAN PUSTAKA

A. Landasan Teori

1. Sistem pakar

Sistem pakar adalah program kecerdasan buatan yang menggabungkan pangkalan pengetahuan *base* dengan sistem inferensi untuk menirukan seorang pakar. Banyak penelitian yang dilakukan dengan memanfaatkan sistem pakar, karena seperti yang sudah kita ketahui, teknologi infomasi sudah masuk ke dalam semua bidang tidak hanya pada bidang komputer. Pada dasarnya Sistem pakar ini dibangun dimaksudkan untuk menggantikan peran dari seorang pakar (Santi & Andari, 2019).

Menurut R. I. Borman dalam Ridho Handoko & Neneng, Sistem pakar merupakan cabang dari kecerdasan buatan (*Artiicial Intelligence*) yang cukup tua karena sistem ini mulai dikembangkan pada pertengahan 1960. Sistem ini bekerja untuk mengadopsi pengetahuan manusia ke komputer yang menggabungkan dasar pengetahuan untuk menggantikan seorang pakar dalam menyelasaikan suatu masalah (Ridho Handoko & Neneng, 2021). Sistem pakar dirancang untuk menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli dalam menjawab pertanyaan dan memecahkan suatu masalah. Dengan adanya siistem pakar juga memberikan kemudahan kepada seorang pakar dalam aktivitasnya sebagai asisten yang sangat berpengalaman. Komponen-komponen pada sistem pakar yaitu:

1) *Knowledge Base*, adalah kumpulan informasi dan pengetahuan yang diperoleh dari ahli atau sumber-sumber lainnya yang berisi aturan-aturan atau *knowledge representation* (representasi pengetahuan) yang digunakan oleh sistem pakar untuk melakukan penalaran dan memberikan solusi atau rekomendasi untuk permasalahan yang dihadapi.
2) *Inference engine*, melakukan proses penalaran seperti yang dilakukan oleh seorang ahli pakar. *Inference engin*e mengambil keputusan atau memberikan solusi berdasarkan pengetahuan yang terdapat pada

knowledge base. Hasil yang diberikan oleh *inference engine* dapat berupa jawaban, rekomendasi, atau diagnosa.

3) *User interface,* bagian yang berhubungan langsung dengan pengguna atau user. *User interface* dibagi menjadi dua jenis, yaitu *text-based interface* dimana *user* yang menggunakan teks atau karakater seabagai *input* dan *output* seperti *command-line interface* atau *console.* Dan *graphical user interface* (GUI) antarmuka yang menggunakan elemen-elemen *grafis* seperti tombol, menu, dan ikon sebagai *input* dan *output*.

2. Pendeteksi Citra

Deteksi adalah proses identifikasi atau pengenalan suatu objek atau fenomena tertentu, biasanya melibatkan pengumpulan data atau informasi. Proses memeriksa terhadap sesuatu dengan menggunakan cara dan teknik tertentu. Deteksi dapat digunakan untuk berbagai masalah, misalnya dalam sistem pendeteksi suatu penyakit, dimana sistem mengidentifikasi masalah-masalah yang berhubungan dengan penyakit yang biasa disebut gejala. Tujuan dari deteksi adalah memecahkan suatu masalah dengan berbagai cara tergantung metode yang diterapkan sehingga menghasilkan sebuah solusi (Maulana et al., 2018). Pengolahan citra (*image processing*) mempunyai keterikatan yang erat dengan disiplin ilmu yang jika sebuah disiplin ilmu dinyatakan dalam bentuk proses suatu *input* menjadikan *output*, maka pengolahan citra memiliki *input* berupa citra serta *output* berupa citra (Rilo Pambudi et al., 2020).

Pendeteksi citra atau *image detector* adalah perangkat atau program yang digunakan untuk menemukan objek, pola, atau fitur pada citra pada digital/ pendeteksi citra dapat digunakan dalam berbagai aplikasi seperti pengenalan wajah, pengenalan tanda tangan, deteksi gerakan, dan pengenalan karakter Tulisan tangan. Pendeteksi citra menggunakan teknik pengolahan citra dan pembelajaran mesin dalam melakukan tugasnya. Teknik pengolahan citra dapat digunakan untuk memperbaiki kualitas citra, menghilangkan noise, meningkatkan kontras, dan lain sebagainya.

3. Tanaman Kentang

Tanaman sayuran semusim, berumur pendek kurang lebih hanya 90–180 hari dan berbentuk perdu atau semak. Kentang memiliki kandungan karbohidrat yang tinggi serta mengandung serat, vitamin C, vitamin B6, kalium, dan zat besi. Kentang dapat tumbuh di daerah dengan iklim sedang hingga dingin, dengan suhu tanah yang ideal antara 14 – 18 derajat *celcius*. Tanaman kentang membutuhkan tanah yang lembab dan gembur dengan Ph antara 5-7. Kentang dapat ditanam di lahan pertanian, di pekarangan rumah, atau didalam pot dengan ukuran yang sesuai. Tanaman kentang juga dapat tumbuh dengan baik di daerah dataran tinggi atau pegunungan dengan ketinggian antara 800-2.500 meter di atas permukaan laut.

4. Penyakit Tanaman Kentang

Penyakit pada tanaman kentang beberapa disebabkan oleh bakteri, virus, maupun cendawan. Namun, penyakit tanaman kentang umum terjadi akibat beberapa patogen yang menyerang tanaman kentang, faktor seperti iklim dan cuaca pun bisa menjadi salah satu penyebab munculnya penyakit pada tanaman kentang. Jika tidak dicegah dan ditangani dengan baik maka bisa akan berpengaruh berpengaruh pada penurunan hasil produksi (Lesmana et al., 2022).

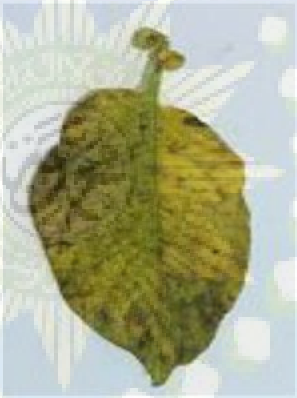
Beberapa penyakit yang sering menyerang tanaman kentang mulai dari daun, batang hingga umbi yaitu:

Penyakit pada daun kentang:

1) Busuk daun (*Late Blight*) Penyakit busuk daun disebabkan oleh *Phytophthora infestans* yang merupakan penyakit utama pada tanaman kentang. Gejala awalnya yaitu berupa bercak kebasah-basahan yang terdapat pada bagian tepi atau tengah daun. Kemudian bercak tersebut akan semakin melebar dan terbentuk daerah nekrotik yang berwarna coklat (Ningsih, 2021).

Gambar 1.Daun *Late blight* (Rakhmawati et al., 2018).

2) Bercak kering daun (*Early blight*) penyakit yang disebabkan oleh jamur *Alternaria solani*. Penyakit ini dapat mempengaruhi daun, tangkai, batang, dan umbi kentang. Gejala dari pennyakit ini adalah munculnya bercak kecil berwarna coklat pada daun kemudian daun menguning.



Gambar 2. Daun *Early Blight* (Rakhmawati et al., 2018).

Penyakit pada batang kentang:

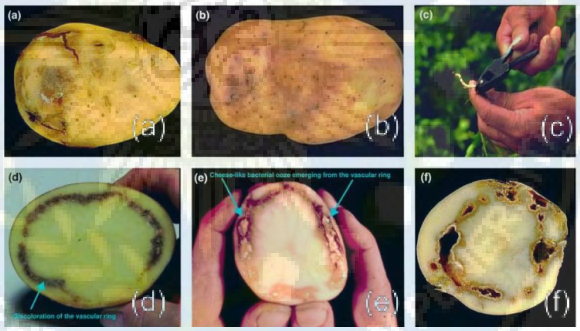
1) Layu fusarium (*Fusarium wilt*) Penyakit yang disebabkan oleh jamur *fusarium oxysporum* pada tanaman kentang dengan gejala daun mulai menguning, layu kemudian menyebar ke seluruh tanaman, batang pada bagian bawah terlihat lembek dan warnanya menjadi lebih gelap jika batang di potong maka terlihat adanya bekas busuk, akar tanaman akan membusuk dan menjadi lebih kecil, kemudian tanaman yang terinfeksi *fusarium wilt* biasanya menghasilkan umbi yang sedikit. Penyakit ini biasanya menyerang bagian akar dan batang tanaman, menyebabkan daun layu dan mati. Jamur ini dapat hidup ditanah

selama bertahun-tahun dan menyebar melalui bibit atau sisa tanaman yang terinfeksi.

2) Karat batang (*Stem rust*) Penyakit ini disebabkan oleh jamur *Puccinia graminis* yang menyebabkan kerusakan pada batang dan daun kentang. Gejalanya meliputi bitnik-bintik merah kecoklatan pada batang dan daun, daun mungkin akan menguning kemudian rontok.

Penyakit pada umbi kentang:

1) Busuk cincin bakteri (*Bacterial ring rot*) Penyakit menular yang disebabkan oleh bakteri *Clavibacter michiganensis*. Gejalanya meliputi pada permukaan luar kentang akan terlihat adanya retakan (**Gambar 3a**) dan nampak lebih berwarna coklat kemerahan dibandingkan dengan umbi kentang yang sehat (**Gambar 3b**). Ketika umbi kentang dipotong membujur akan terlihat lingkaran berwarna coklat, berlubang dan ketika diperas akan mengeluarkan cairan seperti seperti keju (**Gambar 3d dan 3e**). (Aisyah et al., 2022).

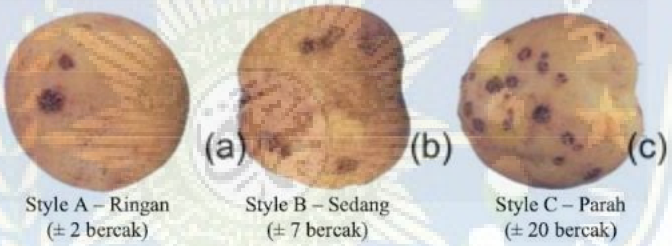


Gambar 3. Gejala busuk cincin bakteri yang disebabkan oleh *Clavibacter sepedonicus* pada umbi kentang (Aisyah et al., 2022).

2) Busuk lunak bakteri (*Bacterial soft rot*) Penyakit yang disebabkan oleh bakteri *Erwinia carotovora* yang menyerang tanaman kentang. Menurut Javandira et al., dalam Aisyah et al., penyakit ini mampu tersebar akibat bakteri *Erwinia carotovora* terbawa oleh tanah dan sulit untuk dikendalikan secara kimiawi dan penyebaran daro bakteri ini berlangsung cepat. Faktor utama dalam penyebaran penyakit ini adalah kelembaban udara yang tinggi (Aisyah et al., 2022).

Gambar 4. Umbi kentang yang terserang penyakit *Bacterial soft rot* (Margareth et al., 2018).

3) Kudis bakteri (*Common scab*) Penyakit yang disebabkan oleh bakteri *Streptomyces scabies*. Gejala yang ditimbulkan dari penyakit ini adalah umbi kentang akan memiliki permukaan yang bertekstur seperti menonjol dan bersisik akibat lentisel yang lebih sedikit dan keras, serta mengalami pembengkakan yang menimbulkan bisul bergabus.



Gambar 5. Umbi kentang yang terserang penyakit kudis bakteri (a) Ringan, (b) Sedang dan (c) Parah (Margareth et al., 2018)

Sebelumnya, peneliti telah melakukan beberapa survey mengenai penyakit tanaman kentang yang paling sering dialami dengan para petani, pada 3 lokasi lahan pertanian. Tabel berikut menunjukkan penyakit yang sering dijumpai pada 3 lahan pertanian tersebut.

Table 1. *Persentase* penyakit tanaman kentang

Nama penyakit	**Persentase penyakit**		
	Lahan 1	**Lahan 2**	**Lahan 3**
Early blight	75 %	30-45%	30-45%
Late blight	30-45 %	35%	40%
Stem rust	20%	20%	20%

Fusarium wilt	20%	20%	20%
Common scab	30%	75%	75%
Bacterial ring rot	30%	50%	50%
Bacterial soft rot	25%	20%	25%

Dari tabel di atas, pada ketiga lahan pertanian tersebut penyakit *early blight* dan *common scab* adalah penyakit kentang yang paling sering dialami oleh para petani. Solusi untuk penanggulangan penyakit ini adalah:

a) *Early blight*: untuk mencegah penyebaran saat mulai timbul gejala, di lakukan dengan penyemprotan dengan fungisida. Ada dua jenis bahan fungisida yang digunakan untuk pencegahan penyakit *early blight* yaitu fungisida kontak dan fungisisda sistemik. Merek fungisida kontak yang bisa digunakan petani adalah *platoon, victory* dan untuk sistemik yaitu merek sirkus, *acrobat*. Cara penyemprotan dengan mencampur salah satu dari masing-masing merek tersebut dengan takaran untuk kontak yaitu 10ml/10L air dan untuk sistemik yaitu 5ml/10L air, waktu penyemprotan yang bagus adalah dini hari sebelum matahari terbit dan saat embun masih ada pada daun kentang atau saat sore hari sebelum daun terkena embun. karna embun yang mengering di daun dapat menyebabkan pembusukan.
b) *Common scab:* tergantung dari ph tanah, cara penanggulangannya bisa dengan melakukan penanaman selang seling dengan tanaman jenis lain selain tomat dan kentang, mengurangi penggunaan pupuk kimia dan diganti dengan pupuk kompos, atau tanah diistirahatkan selama 1 sampai 2 tahun.

5. Data

Data merupakan kumpulan fakta, informasi, atau nilai numerik yang disimpan dalam format berupa teks, gambar, suara, video. Menurut Sutabri dalam Abdurahman, data adalah fakta mengenai objek. (Abdurahman, 2018). Pengelolaan data adalah segala macam pengelolaan terhadap data atau kombinasi-

kombinasi dari berbagai macam pengelolaan terhadap data untuk membuat data itu berguna sesuai dengan hasil yang diinginkan dapat segera dipakai (Hidayatulloh et al., 2020).

Data biasanaya digunakan untuk tujuan analisis, pengambilan keputusan, atau untuk membuat model dan prediksi. Dataset sering digunakan dalam ilmu data, pembelajaran mesin, dan kecerdasan buatan untuk melatih algoritma dan model untuk mengenali pola dan membuat prediksi berdasarkan data tersebut.

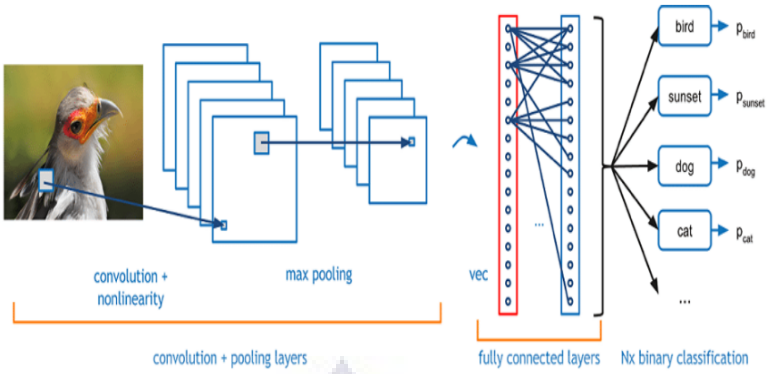
6. *Python*

Menurut Ljubomir Perkovic dalam Dewi, *Python* merupakan bahasa pemrograman dengan tujuan umum yang dikembangkan secara khusus untuk membuat *source code* mudah dibaca. *Python* juga memiliki library yang lengkap sehingga memungkinkan programmer untuk membuat aplikasi yang mutakhir dengan menggunakan *source code* yang tampak sederhana (S. R. Dewi, 2018).

Python memiliki Pustaka atau library yang dapat digunakan untuk memperluas fungsionalitas bahasa ini. Pustaka tersebut seperti *numPy, Pandas, Matplotlib, Scikit-learn,* dan *Tensorflow* yang dapat membantu pengguna untuk melakukan bebagai tugas seperti pemrosesan data, visualisasi data, dan pembelajaran mesin.

7. *Convolutional Neural Network* (CNN)

Convolutional Neural Network merupakan salah satu jenis algoritma supervised yang cara kerjanya adalah menerima input berupa gambar. *Convolutional Neural Network* (CNN) merupakan algoritma *deep learning* karena kedalaman jaringannya yang tinggi. Sedangkan Algoritma CNN merupakan hasil dari pengembangan *Multilayer Perceptron* (MLP) yang memungkinkan untuk melakukan pengolahan data dua dimensi sehingga algoritma CNN dapat digunakan dalam pengelolahan data citra atau suara.



Gambar 6. *Convolutional Neural Network*

Beberapa komponen kunci dari CNN termasuk:

a) *Convolutional Layer*, adalah lapisan yang berisi sejumlah filter atau kernel yang digeser (dikonvolusi) melintasi gambar. Setiap filter dapat mengidentifikasi fitur-fitur seperti tepi, sudut, dan tekstur.

b) *Pooling Layer*, lapisan ini berfungsi untuk mengurangi dimensi peta fitur dan meringkas informasi penting. Pooling umumnya melibatkan operasi seperti maksimum (*max-pooling*) atau rata-rata (*average-pooling*) pada area tertentu dari peta fitur.

c) *Activation Function*, setelah setiap operasi konvolusi dan pooling, fungsi aktivasi seperti ReLU (Rectified Linear Unit) biasanya diterapkan untuk memperkenalkan non-linearitas pada model.

d) *Fully Connected Layer*, ini adalah lapisan yang mirip dengan jaringan saraf biasa, di mana setiap *neuron* dihubungkan ke setiap *neuron* di lapisan sebelumnya. Lapisan ini digunakan untuk menggabungkan fitur-fitur yang telah dipelajari oleh lapisan sebelumnya.

e) *Output Layer*, lapisan terakhir yang menghasilkan hasil prediksi. Jumlah neuron di lapisan ini bergantung pada jumlah kelas yang ingin diprediksi.

CNN dipelajari melalui tahap pelatihan dengan menggunakan algoritma pembelajaran seperti *stochastic gradient descent* (SGD) untuk mengoptimalkan bobot dan parameter lainnya. Saat pelatihan, CNN belajar mengenali pola dan fitur-fitur yang relevan dalam data pelatihan, yang memungkinkannya untuk

mengenali objek, wajah, atau pola lain dalam data baru yang belum pernah dilihat sebelumnya.

8. *Library Python*

Library python adalah Kumpulan modul, fungsi, dan kode yang telah dikembangkan sebelumnya oleh komunitas atau individu untuk memudahkan tugas-tugas pemrograman dalam berbagai bidang. Beberapa *library python* yang bisa digunakan dalam pembelajaran mesin dan *deep learning* pada pembuatan sistem pakar, seperti *numpy, keras, tensorflow, sklearn, seaborn* dan *matplotlib*.

NumPy merupakan library pada bahasa pemrograman *Python* yang lebih menekankan pada komputiasi ilmiah. *NumPy* dapat melakukan pembentukan objek N-dimensional array yang hampir sama dengan *list* di *python*. Namun *NumPy* memiliki kelebihan penggunaan memori yang lebih kecil dan durasi runtime yang lebih singkat dari pada *list* di *pyhton.* (Burch dan Grudnitski dalam (Fauzi, 2019). *Keras* adalah sebuah *library* yang digunakan untuk membangun, melatih, dan mengevaluasi model *neural networks* dalam bahasa pemrograman *python*. *Tensorflow* merupakan perpustakaan perangkat lunak yang dikembangkan

oleh Tim *Google Brain* dalam organisasi penelitian Mesin Cerdas Google, untuk tujuan melakukan pembelajaran mesin dan penelitian jaringan syaraf dalam. Tensorflow menggabungkan aljabar komputasi teknik pengoptimalan kompilasi, mempermudah penghitungan banyak ekspresi matematis dimana masalahnya adalah waktu yang dibutuhkan untuk melakukan perhitungan (S. R. Dewi, 2018). *Sklearn* menyediakan berbagai algoritma pembelajaran mesin dan alat untuk memproses serta menganalisis data. *Matplotlib* merupakan sebuah *library* yang digunakan untuk visualisasi data dalam bahasa pemrograman dengan membuat berbagai jenis grafik dan visualisasi data untuk membantu menganalisis dan memahami data.

9. *Splitting Data*

Splitting data adalah proses membagi dataset menjadi dua atau lebih subset yang berbeda untuk digunakan dalam pelatihan, validasi, dan pengujian model dalam *machine learning* atau analisis data. Tujuan dari *splitting* data adalah untuk

mengukur performa model secara objektif dengan menggunakan data yang tidak terlihat sebelumnya (data train atau valid). Biasanya, dataset dibagi menjadi tiga subset utama:

a. *Training Set*, digunakan untuk melatih model. Model belajar dari pola dalam data ini sehingga dapat melakukan prediksi pada data yang belum pernah dilihat sebelumnya.

b. *Validation Set*, digunakan untuk mengatur parameter model atau untuk memilih model terbaik dari beberapa opsi yang berbeda. Validasi membantu mencegah *overfitting* (model terlalu sesuai dengan data pelatihan) karena dapat memantau performa model di atas data yang tidak digunakan dalam pelatihan.

c. *Testing Set*, digunakan untuk mengukur kinerja akhir model secara objektif. Data ini seharusnya tidak dilihat oleh model selama proses pelatihan atau validasi, sehingga memberikan gambaran yang lebih akurat tentang seberapa baik model dapat generalisasi pada data baru.

Pemisahan ini penting untuk memastikan bahwa model tidak hanya menghafal data pelatihan (*overfitting*), tetapi juga mampu melakukan prediksi yang baik pada data yang belum pernah dilihat sebelumnya. Pemisahan data dapat dilakukan dengan berbagai cara, seperti pemisahan acak (*random*) atau pemisahan berdasarkan waktu. Terkadang, metode validasi silang (*cross-validation*) juga digunakan untuk memaksimalkan penggunaan data dan mengurangi bias dalam pengujian.

10. *Batch size*

Batch size adalah parameter yang digunakan dalam pelatihan model dalam berbagai tugas pembelajaran mesin, terutama dalam pelatihan jaringan saraf *neural networks*. Ini mengacu pada jumlah contoh yang diberikan kepada model dalam satu iterasi pelatihan sebelum bobot model diperbarui. Dalam istilah praktis, *batch size* mempengaruhi seberapa banyak memori dan sumber daya

komputasi yang diperlukan dalam setiap iterasi pelatihan. Berikut adalah beberapa poin penting untuk memahami tentang *batch size*:

a. . *Stochastic Gradient Descent (SGD)*: Saat model diperbarui dengan setiap batch data, pendekatan pelatihan yang umum digunakan adalah *Stochastic Gradient Descent* (SGD). Dalam SGD, bobot model diperbarui berdasarkan *gradien* dari fungsi kerugian yang dihitung atas batch data tersebut.
b. Ukuran *Batch*: *Batch size* menentukan berapa banyak contoh yang diberikan kepada model dalam satu iterasi. *Batch size* dapat berupa angka yang lebih kecil (misalnya 16, 32, 64, atau 128) atau lebih besar tergantung pada kompleksitas model dan ketersediaan sumber daya.
c. Keuntungan Batch Size Kecil: *Batch size* yang lebih kecil cenderung memberikan pembaharuan bobot yang lebih sering, yang dapat membantu dalam konvergensi lebih cepat dan mungkin menghasilkan generalisasi yang lebih baik jika data pelatihan memiliki variasi yang tinggi.
d. Keuntungan *Batch Size* Besar: *Batch size* yang lebih besar dapat memanfaatkan efisiensi komputasi modern, seperti pemrosesan paralel *GPU*, untuk menghitung gradien secara efisien. Ini dapat mempercepat pelatihan.
e. *Overfitting*: *Batch size* juga dapat mempengaruhi risiko *overfitting*. *Batch size* yang lebih kecil bisa membantu mencegah *overfitting* karena model lebih sering diperbarui, tetapi risiko dari *noise* yang lebih tinggi di dalam gradien juga bisa menjadi masalah.
f. *Trade-off*: Pemilihan *batch size* melibatkan *trade-off* antara efisiensi pelatihan, konvergensi, dan penggunaan memori. Penting untuk mencoba beberapa ukuran *batch* dan mengawasi dampaknya terhadap pelatihan dan kinerja validasi.

Dalam banyak kasus, *batch size* adalah salah satu parameter yang harus dieksperimenkan dan disesuaikan selama proses pelatihan. Ada juga teknik seperti

"*mini-batch gradient descent"* yang membagi dataset pelatihan menjadi *batch* yang lebih kecil dan lebih sesuai dengan sumber daya yang tersedia.

11. *Epoch*

Epoch adalah satu putaran penuh dari pelatihan suatu model pembelajaran mesin atau algoritma. Selama satu *epoch*, model akan melihat dan memproses seluruh data pelatihan yang tersedia. Pada akhir setiap *epoch*, model akan dievaluasi berdasarkan performanya terhadap data validasi yang terpisah dari data pelatihan. Konsep *epoch* sangat penting dalam pembelajaran mesin terutama saat melibatkan algoritma pembelajaran berulang, seperti dalam jaringan saraf tiruan (*neural networks*) untuk tugas seperti pengenalan gambar. Dalam setiap *epoch*, parameter-model diperbarui berdasarkan hasil kesalahan yang dihitung selama proses pelatihan. Proses ini berulang-ulang sampai model mencapai tingkat kinerja yang diinginkan atau konvergensi tercapai.

Jumlah *epoch* yang diperlukan untuk melatih model dengan baik dapat bervariasi tergantung pada kompleksitas tugas, volume data, arsitektur model, dan parameter pelatihan lainnya. Terkadang, melatih model terlalu banyak *epoch* dapat menyebabkan *overfitting* atau ketika model terlalu baik menyesuaikan data pelatihan tetapi buruk dalam generalisasi ke data baru, sementara melatih terlalu sedikit *epoch* dapat menyebabkan *underfitting* atau ketika model gagal menangkap pola yang kompleks dalam data.

B. Penelitian Terkait

Penelitian terkait adalah bagian yang membahas dan menganalisis penelitian sebelumnya yang berkaitan dengan masalah yang diteliti. Penelitian terkait dimaksudkan sebagai acuan. Berikut ringkasan hasil penelitian sebelumnya yang berkaitan dengan pebelitian yang dilakukan.

Penelitian terkait dengan pendeteksi citra oleh (Nauval & Lestari, 2022) dengan judul " Implementasi Deteksi Objek Daun Kentang Dengan Metode *Convolutional Neural Network*". Hasil penelitian menunjukkan bahwa, Dari hasil pengujian model tersebut didapatkan nilai hasil akurasi sebesar 94%. Selanjutnya peneliti melakukan ekstraksi fitur dengan metode VGG16 untuk meningkatkan

hasil akurasi. Namun ternyata tidak sesuai harapan karena hasil akurasinya lebih rendah yaitu 93,3%. Sehingga dihasilkan kesimpulan bahwa deteksi citra daun tanaman kentang menggunakan metode CNN sesuai harapan. Adapun hasil confusion matrix dari pengujian yang dilakukan menunjukkan jumlah citra daun tanaman kentang dengan label late blight yang terdeteksi benar adalah sebanyak 88 gambar. Sedangkan jumlah citra daun tanaman kentang dengan label early blight yang terdeteksi benar adalah sebanyak 97 gambar. Selanjutnya jumlah citra daun tanaman kentang dengan label healthy yang terdeteksi benar adalah sebanyak 97 gambar.

Penelitian serupa juga dilakukan oleh (Ningsih, 2021) Progrma Studi Teknik Informatika Universitas Yudharta Pasuruan dengan judul skripsi "Klasifikasi Jenis Penyakit Daun Kentang Menggunakan *Convolutional Neural Network* (CNN) Model Arsitektur *GoogleNet*". Hasil penelitian menunjukkan bahwa, Akurasi model dari data train dan data testing jika dilihat dari epoch 1 sampai 10 dominan naik, namun juga ada penurunan pada epoch tertentu, kenaikan dan penurunan akurasi model terjadi karena data yang diprediksi benar setiap iterasi selalu berbeda atau naik turun. Waktu pada komputasi model dominan cepat, namun juga ada beberapa yang lambat dalam komputasi pada beberapa *batch size* dan *dropout*.

Selain itu, penelitian yang dilakukan oleh (Rozaqi et al., 2021) dengan judul "*Detection of Potato Diseases Using Image Segmentation and Multiclass Support Vector Machine*" menggunakan metode *Support Vector Machine* (SVM) untuk mendeteksi penyakit pada daun kentang dengan jumlah data gambar daun yang digunakan 200 data daun yang sakit dan 100 daun yang normal atau sehat. Proses pada penelitian ini adalah dengan melakukan segmentasi citra untuk menampilkan penyakit pada daun saja tanpa menampilkan background dan daun yang normal dalam artian warna hijau daun, kemudian gambar daun akan di ekstrak tekstur nya menggunakan *Gray Level Co-occurrence Matrix* (GLCM). Hasil dari penelitian ini meiliki akurasi sebesar 95%.

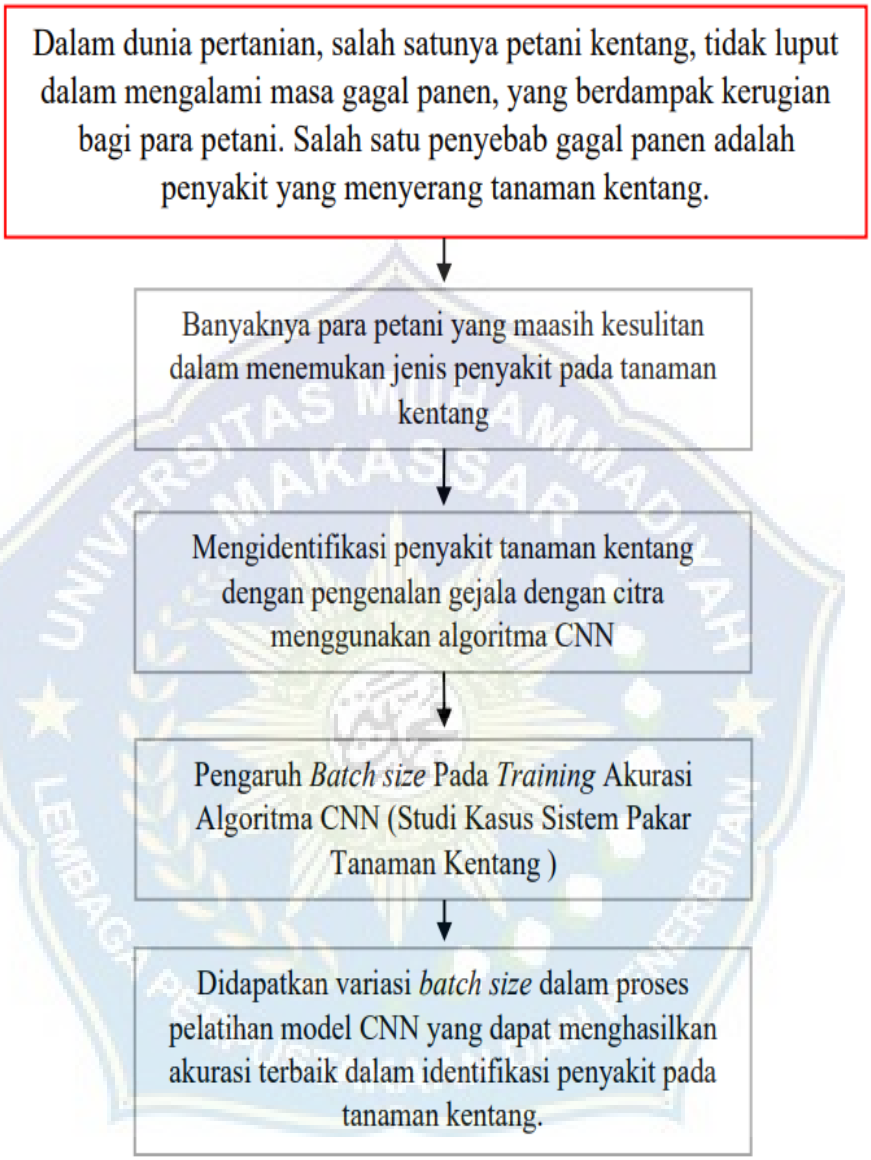
Selanjutnya, penelitian dengan sistem pakar mendeteksi penyakit tanaman kentang juga dilakukan oleh (T. S. Dewi et al., 2020) Program Studi Sistem

Informasi, STMIK Triguna Dharma dengan judul "Sistem Pakar Mendiagnosa Penyakit Busuk Daun Pada Tanaman *Solanum Tubersum* (Kentang Berumbi Merah) Menggunakan Metode *Teorema Bayes*". Hasil penelitian menunjukkan bahwa, Berdasarkan pengujian metode *Teorema Bayes*, maka dilakukan inisialisasi gejala, mencari nilai keyakinan untuk mendapatkan hasil diagnosa untuk mengidentifikasi dan manganalisis penyakit tanaman kentang berumbi merah berdasarkan gejala-gejala dengan melakukan riset untuk mendapatkan gejala-gejala dari daun busuk. Berdarasarkan pembangunan aplikasi, maka yang dilakukan adalah merancang dengan menggunakan bahasan pemodelan *Unified Modeling Language* (UML) dan menggunakan bahasa pemograman *visual basic*.

Penelitian pendeteksi citra oleh (Kusumadewa & Supatman, 2018) Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Mercu Buana Yogyakarta dengan judul "Identifikasi Citra Daun Teh Menggunakan Metode Histogram untuk Deteksi Dini Serangan Awal Hama Empoasca". Hasil penelitian menunjukkan bahwa Identifikasi Citra Daun Teh Menggunakan Metode Histogram Untuk Deteksi Dini Serangan Awal Hama Empoasca dengan 44 data uji memperoleh unjuk kerja sebesar 95,45% pada parameter *alfa* 0,1 dan *dec alfa* 0,5.

C. Kerangka Pikir

Kerangka pikir dalam penelitian ini bisa digambarkan sebagai berikut:



Gambar 7. Kerangka Pikir.

BAB III
METODE PENELITIAN

A. Tempat Dan Waktu Penelitian

1. Tempat Penelitian

   Penelitian ini dilakukan di lahan pertanian BPP Kanreapia yang dalam penelitian ini pusat datanya adalah Tanaman kentang.

2. Waktu Penelitian

   Kegiatan penelitian akan dilakukan mulai Mei 2023, kegiatan penelitian ini akan dilakukan hingga seluruh prosedur pengumpulan data selesai.

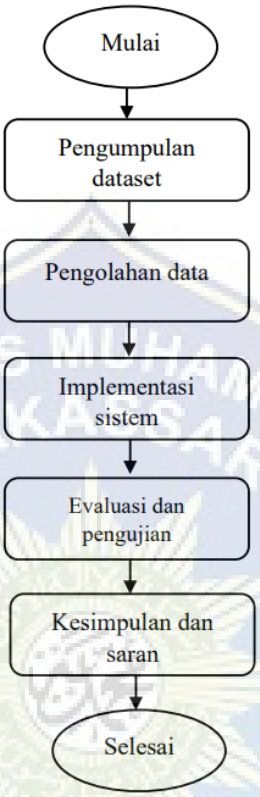
B. Alat Dan Bahan

Untuk pembuatan sistem pakar pendeteksi citra pada penelitian ini alat dan bahan yang perlukan yaitu:

1. Kebutuhan *Hardware* (Perangkat Keras)
   a) *Smartphone android* oppo A17
   b) Laptop Asus Vivobook m415da dengan spesifikasi:
      - Besar memori RAM 8GB
      - Kapasistas *storage* 512GB
      - Monitor 14.0″ dengan resolusi 1920x1080 pixel
      - *Processor* AMD Ryzen 3
      - Windows 11
2. Kebutuhan *Software* (Perangkat Lunak)
   a) *TensorFlow*
   b) *Google colaboratory*
   c) *Keras*
   d) *Numpy*
   e) Sistem operasi Windows 11

C. Perancangan Sistem

Perancangan sistem menggunakan *flowchart*.



Gambar 8. *Flowchart system*

1. Pengumpulan Dataset

Berdasarkan sumber yang ada data dibagi menjadi dua, yakni data primer yaitu pengambilan data secara langsung dari sumber data, seperti angket, observasi, dan wawancara yang dilakukan peneliti secara langsung dan data skunder yaitu pengambilan data yang dilakukan oleh penelitian lain. Dalam penelitian ini, peneliti menggunakan kedua metode di tersebut untuk mengumpulkan data.

2. Pengolahan Data

Pada tahap ini dilakukan resize ukuran dataset kemudian membagi kelas sesuai dengan klasifikasi penyakit untuk melakukan proses penelitian, menentukan *batch size* dan *splitting* data sehingga mecapai tujuan penelitian dengan metode penelitian yang telah di tentukan. Pada tahap ini merupakan proses yang dilakukan untuk deteksi penyakit daun kentang menggunakan *deep learning* klasifikasi data citra.

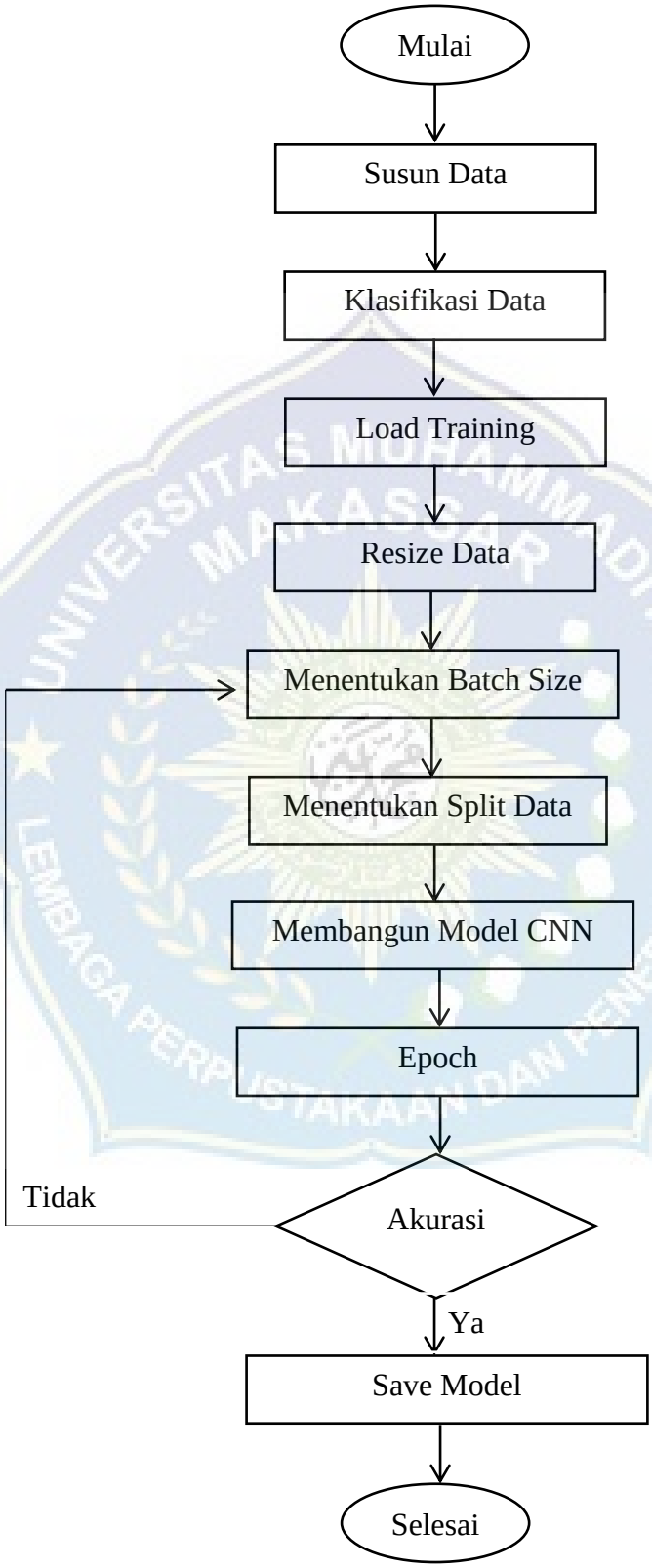
3. Implementasi Sistem

Implementasi merupakan pelaksanaan atau penerapan. Yang dilaksanakan dan diterapkan yaitu pokok-pokok yang telah dirancang atau didesain sebelumnya, kemudian dijalankan sepenuhnya. Pada tahapan ini dilakukan implemenatasi sistem berupa peng-codean menggunakan Bahasa pemrograman *Phyton.*

4. Evaluasi dan pengujian sistem

Evaluasi dan pengujian sistem merupakan sistem yang dirancang untuk memastikan bahwa sistem yang dibuat berfungsi sesuai dengan yang di harapkan.

Percangan sistem *Training* adalah sebagai berikut :



Flowchart perancangan sistem

Flowchart diatas dijelaskan dengan:

1. **Susun Data**, mengumpulkan dan mengatur dataset citra yang berisi berbagai jenis citra dan diklasifikasikan. Setiap citra memiliki label yang menunjukkan kategori atau kelas.

2. **Klasifikasi Data**, mengelompokkan citra dalam dataset ke dalam kelas-kelas yang sesuai. Setiap citra akan diberi label yang mewakili kategori di mana citra tersebut termasuk.

3. ***Load Training***, Dataset yang telah diberi label kemudian dimuat ke dalam model CNN dengan memproses dan melatih model CNN menggunakan data tersebut.

4. ***Resize* Data**, gambar dalam dataset memiliki ukuran yang berbeda. Agar data cocok untuk dimasukkan ke dalam model CNN, perlu menyesuaikan ukuran gambar. Dengan melakukan resize gambar menjadi 150x150.

5. **Menentukan *Batch Size***, Saat melatih model CNN, data tidak diberikan sekaligus, tetapi dalam batch-batch kecil. Batch size adalah jumlah citra yang diberikan pada setiap iterasi selama pelatihan. Dalam penelitian ini menggunakan uji coba *batch size* 16,32,64,128.

6. **Menentukan *Split Data***, Dataset dibagi menjadi set pelatihan (*training set*) dan set validasi (*validation set*) untuk mengukur kinerja model selama pelatihan. Set pelatihan digunakan untuk melatih model, sedangkan set validasi digunakan untuk mengukur seberapa baik model menggeneralisasi pada data yang belum pernah dilihat sebelumnya. Dalam penelitian ini menggunakan *split* 80:10:10 dan 70:15:15

7. **Membangun Model CNN**, merancang arsitektur model CNN. Arsitektur ini terdiri dari berbagai lapisan konvolusi, penggabungan, dan lain-lain yang berfungsi untuk mengekstraksi fitur-fitur dari citra dan membuat prediksi klasifikasi.

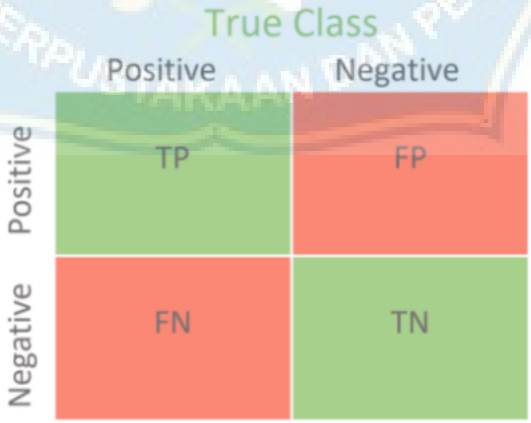
8. ***Epoch***, Satu epoch adalah satu kali melatih model dengan seluruh data pelatihan. Proses pelatihan dilakukan dalam 10 *epoch*, di mana model diperbarui dan ditingkatkan setiap kali melalui data pelatihan.

9. **Akurasi didapatkan**, akurasi dihitung dengan membandingkan prediksi yang dibuat oleh model dengan label sebenarnya dari data yang memberi gambaran tentang seberapa baik model bekerja dalam mengklasifikasikan dataset

D. Teknik Pengujian

1. Pengujian *Confusion Matrix*

Confusion matrix merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Dalam *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya. Terdapat 4 (empat) istilah pada kinerja *confusion matrix* sebagai representasi hasil proses klasifikasi, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). *Nilai True Negative* (TN) merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan *False Positive* (FP) merupakan data negatif namun terdeteksi sebagai data positif. (Karsito & Susanti, 2019)



Gambar 9. *Confusion matrix* (Rabbani et al., 2021)

$$\text{Accuracy} = \frac{tp+tn}{tp+fn+fp+tn} \quad (1)$$

Akurasi *(Accuracy)*, untuk mengukur sejauh mana model memprediksi dengan benar.

$$\text{Precision} = \frac{tp}{tp+fp} \quad (2)$$

Presisi *(Precision)*, untuk mengukur berapa persen dari prediksi positif yang sebenarnya benar.

$$\text{Recall} = \frac{tp}{tp+fn} \quad (3)$$

Recall (Sensitivitas atau *True Positive Rate*): Mengukur berapa persen dari kasus positif yang diidentifikasi dengan benar oleh model.

Confusion matrix dalam konteks klasifikasi sering disajikan dalam bentuk visual yang memiliki variasi warna sebagai upaya untuk membuatnya lebih mudah dibaca dan diinterpretasikan. Warna dapat membantu membedakan berbagai elemen dalam confusion matrix dengan lebih jelas, sehingga informasi yang terkandung di dalamnya lebih mudah dimengerti. Biasanya, dalam sebuah confusion matrix warna-warni, elemen-elemen berbeda seperti *true positive*, *false positive*, *true negative*, dan *false negative* dapat diwakili oleh warna yang berbeda. Misalnya, warna yang lebih terang atau hijau mungkin digunakan untuk mewakili nilai yang lebih baik atau hasil yang lebih positif, sementara warna yang lebih gelap atau merah mungkin digunakan untuk nilai yang lebih rendah atau hasil yang kurang positif.

E. Teknik Analisis Data

Menurut Milles and Huberman, analisis data tertata dalam situs ditegaskan bahwa kolom pada sebuah matriks tata waktu disusun dengan jangka waktu, dalam susunan tahapan, sehingga dapat di lihat kapan gejala tertentu terjadi.

Selain menggunakan metode pengumpulan dan analisis data secara sekunder, analisis data dalam penelitian ini juga menggunakan teknik analisis data dalam situs yang dikembangkan oleh Miles Huberman. Data yang sudah terkumpul di buat dalam matriks. Dalam matriks akan disajikan penggalan-

penggalan data deskriptif sekitar peristiwa atau pengalaman tertentu yang membagi data sebelum dan sesudahnya. Setelah data dimasukkan kedalam matriks selanjutnya di buat daftar cek.

Analisis data dalam penelitian ini dilaksanakan pada saat pengumpulan data dalam periode tertentu. Pada saat wawancara, peneliti sudah melakukan analisis terhadap jawaban yang diwawancarai. Apabila jawaban yang disampaikan oleh orang yang diwawancarai atau informan setelah dianalisis dirasa kurang memuaskan, maka peneliti akan melanjutkan pertanyaan lagi, sampai tahap tertentu sehingga diperoleh data atau informasi yang lebih kredibel (Iii, 2019).

1. Pengumpulan data

Pada analisis model pertama dilakukan pengumpulan data hasil wawancara, hasil observasi, dan berbagai dokumen berdasarkan kategorisasi terkait masalah yang terjadi di lahan BPP Kanreapia yang sesuai dengan masalah penelitian yang kemudian dikembangkan penajaman data melalui pencarian data selanjutnya.

2. Reduksi data

(Miles dan Huberman) Reduksi data adalah suatu bentuk analisis yang menajamkan, menggolongan, mengarahkan, membuang data yang tidak perlu dan mengorganisasi data dengan cara sedemikian rupa sehingga simpulan final dapat ditarik dan diverifikasi.

3. Penyajian data

Penyajian data bertujuan untuk menemukan kesimpulan riset yang dilakukan serta memberikan kemungkinan adanya penarikan simpulan terkait penyakit apa saja yang menyerang tanaman kentang yang di lokasi penelitian.

4. Penarikan kesimpulan

Berdasarkan penyajian data yang telah diperoleh dapat dirumuskan kesimpulan sementara. Hasil yang diperoleh berupa pencatatan, pola-pola, pernyataan-pernyataan, konfigurasi, arahan sebab akibat dan proposisi lain. Kesimpulan sementara ini akan terus berkembang sejalan dengan penemuan data baru dan pemahaman baru, sehingga didapatkan suatu kesimpulan yang benar -

benar sesuai dengan penelitian ini. Kemudian menghasilkan data yang lengkap sehingga dapat dirumuskan kesimpulan akhir.

BAB IV
HASIL DAN PEMBAHASAN

A. Pengumpulan data

Data yang digunakan dalam penelitian ini merupakan data citra daun, batang, dan umbi kentang yang diperoleh dari lahan pertanian BPP Kanreapia sebanyak 5.160 data, dataset kemudian dibagi menjadi data *testing*, data *training*, dan data *validation*. Data *validation* akan menggunakan beberapa scenario pembagian data yaitu 80:10:10 dan 70:15:15. Penelitian ini menggunakan 10 kelas, yaitu kelas daun sehat, *early blight*, *late blight*, batang sehat, *fusarium wilt*, *Stem rust*, umbi sehat, *bacterial ring rot*, *bacterial soft rot*, dan *common scab*. Hasil pengumpulan data di lahan BPP Kanreapia ditunjukkan pada gambar di bawah.

Gambar 10. Hasil pengumpulan Dataset

Sampel data citra pada penyakit tanaman kentang ditunjukkan pada table berikut.

Table 2. Dataset penyakit tanaman kentang.

No.	**Nama Penyakit Kentang**	**Gambar Penyakit**
1.	Daun sehat	

2.	*Early blight*	
3.	*Late blight*	
4.	Batang sehat	
5.	*Stem rust*	
6.	*Fusarium wilt*	

7.	Umbi sehat	
8.	*Common scab*	
9.	*Bacterial ring rot*	
10.	*Bacterial soft rot*	

Jumlah dataset penyakit tanaman kentang yang dikumpulkan dengan jumlah yang berbeda setiap kelasnya. Jumlah dataset setiap kelas depat dilihat pada tabel berikut.

Table 3. Jumlah dataset masing-masing penyakit tanaman kentang.

No.	Nama Penyakit	Jumlah Data
1.	Daun Sehat	500
2.	*Early Blight*	600
3.	*Late Blight*	600
4.	Batang Sehat	500
5.	*Stem Rust*	600
6.	*Fusarium wilt*	600
7.	Umbi Sehat	240
8.	*Common Scab*	500
9.	*Bacterial Ring Rot*	500
10.	*Bacterial Soft Rot*	500
	Jumlah	**5.160**

B. Memuat Dataset

1. Pada tahap ini menghubungkan *google colab* dengan *drive* untuk pemuatan dataset, dan mendefinisikan dari mana sumber data berasal (*dictionary*). Folder untuk dataset ini yaitu '*Potato—Diseases*' dengan 5.160 dataset dan 10 kelas. Berikut kode pemuatan dataset.

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

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/Potato--Diseases",
    shuffle = True,
    image_size = (IMAGE_SIZE, IMAGE_SIZE),
    batch_size = BATCH_SIZE
)
#menampilkan nama kelas
classes  = dataset.class_names
classes
```

Kode di atas untuk memuat dataset ke dalam sistem dan menampilkan masing-masing kelas yang ada dalam dataset memggunakan fungsi *dataset = tf.keras.preprocessing.image_dataset_from_directory()*. Hasil dari masing-masing kelas yang ada dalam dataset adalah:

```
['Bacterial--Ring-Rot',
 'Bacterial--Soft--Rot',
 'Batang--Healthy',
 'Common--Scab',
 'Daun--Healthy',
 'Early--Blight',
 'Fusarium--wilt',
 'Late--Blight',
 'Stem--Rust',
 'Umbi--Healthy']
```

Gambar 11. Hasil run fungsi *classes*

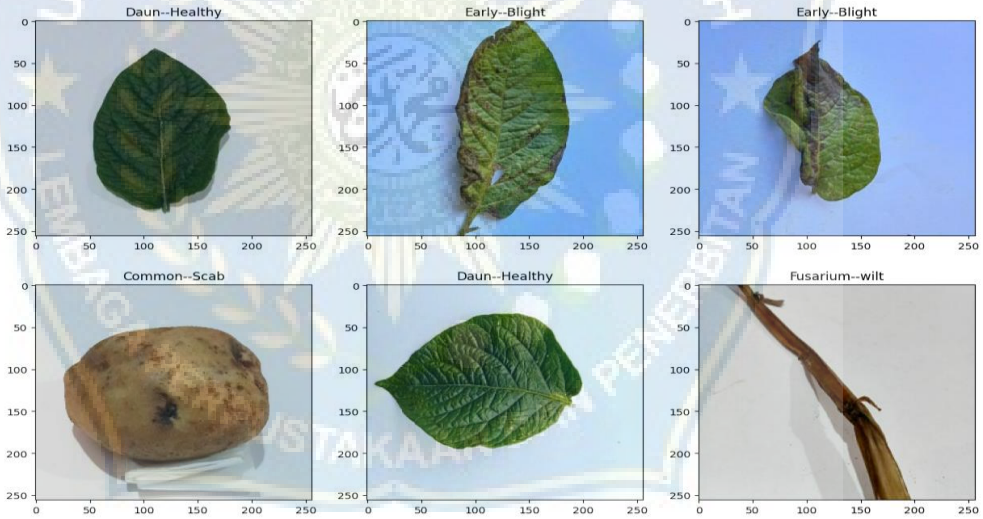
2. Kemudian menentukan karakteristik dataset dengan menggunakan fungsi *image size* yaitu sebagai ukuran yang akan digunakan untuk mengubah ukuran gambar sebelum diproses oleh model. Pada sistem ini, ukuran gambar akan diubah menjadi 150 *pixel* (*pixsel x pixsel*) dalam dimensi tinggi dan lebar. Selanjutnya *batch size* untuk menentukan berapa banyak gambar yang akan diproses dalam satu iterasi selama pelatihan model, dan *channels* untuk menunjukkan jumlah saluran warna dalam gambar. Nilai 3 merujuk pada model warna RGB (*Red, Green, Blue*). Dalam model warna RGB, setiap *pixel* dalam gambar direpresentasikan oleh tiga nilai yang mewakili intensitas warna merah, hijau, dan biru. Dalam penelitian ini terdapat 4 pembagian *batch size* yaitu 16, 32, 64, 128. Setelah melakukan percobaan dengan keempat *batch size,* didapatkan hasil untuk *batch size* 32 mampu meningkatkan efisiensi dan mengurangi waktu pelatihan serta memiliki nilai akurasi yang baik. Berikut adalah kode untuk menentukan karakteristik dataset.

```
IMAGE_SIZE = 150
BATCH_SIZE = 32
CHANNELS = 3
```

3. Untuk menampilkan beberapa gambar dari jumlah dataset yang ada, bertujuan untuk memvisualisasikan citra dataset sesuai dengan masing-masing kelas yang telah ditentukan dapat dilihat menggunakan kode berikut.

```
plt.figure(figsize=(10, 10))

for image_batch, label_batch in dataset.take(1):
  for i in range(6):
    ax = plt.subplot(2, 3, i + 1)
    plt.imshow(image_batch[i].numpy().astype("uint8"))
    plt.title(classes[label_batch[i]])
```

Pada kode di atas, menampilkan 6 gambar dengan label kelas yang sesuai. Menggunakan *ax = plt.subplot* untuk membuat subplot di posisi tertentu, *plt.imshow* untuk menampilkan gambar, dan *plt.title* untuk menampilkan label kelas pada gambar. Berikut hasil run dari kode tersebut:



Gambar 12. Hasil Source code untuk menampilkan dataset

C. Splitting Data

Sebelum pembuatan model CNN perlu dilakukan proses pemilihan dan pembagian data yang digunakan untuk proses *training*, *validation*, *dan testing*. Pada penelitian ini terdapat skenario pembagian data (*spliting data*) yang antara lain 80:10:10 dan 70:15:15. Artinya data akan dibagi menjadi 80% untuk set data *training*, 10 data *validation*, dan 10 data *testing*. Pada penelitian ini akan

memecah data menjadi 80:10:10 karena setelah mencoba skenario pembagian data 80:10:10 dan 70:15:15, pembagian data 80% memiliki akurasi yang lebih baik. Pembagian dataset ini penting untuk menghasilkan model yang dapat diuji dan dievaluasi dengan benar. Berikut kode untuk melakukan *splitting* data.

```
def get_dataset_partition_tf(dataset, train_split = 0.8,
validation_split = 0.1, test_split = 0.1, shuffle = True,
shuffle_size = 10000):
  dataset_size = len(dataset)

  if shuffle:
    dataset = dataset.shuffle(shuffle_size, seed = 18)

  train_size = int(train_split * dataset_size)
  validation_size = int(validation_split * dataset_size)

  train_dataset = dataset.take(train_size)
  validation_dataset =
dataset.skip(train_size).take(validation_size)
  test_dataset =
dataset.skip(train_size).skip(validation_size)

  return train_dataset, validation_dataset, test_dataset
```

Kode di atas bisa dijabarakan sebagai :

a. *'train_split', 'validation_split', 'test_split'* sebagai proporsi dataset yang akan dibagi atau dialokasikan untuk masing-masing bagian.

b. *'shuffle'* untuk menentukan apakah dataset akan diacak sebelum pembagian.

c. *'shuffle_size'* yaitu jumlah sampel yang akan untuk mengacak dataset.

d. Setelah fungsi dieksekusi, akan didapatkan tiga variabel yaitu dataset *training*, dataset *validation*, dan dataset *testing*. Yang masing-masing berisi bagian-bagian yang telah dibagi dari dataset asli. Variabel ini dapat digunakan untuk memvalidasi, melatih, dan menguji model secara terpisah.

D. Model Convolutional Neural Network (CNN)

Tahap ini yaitu pembuatan model CNN pada sistem yang akan dibuat. Sebelum itu, hal yang harus dilakukan adalah mengimport *library* dan input *layer* untuk membuat model CNN.

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
import cv2
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

library diatas berisi:

a. *import tensorflow as tf* yang digunakan untuk mengimpor *TensorFlow*, sebuah *platform* yang digunakan untuk mengembangkan dan melatih model *machine learning* dan *deep learning*.

b. Selanjutnya mengimport modul layer dan layer dari *tensorflow dengan from tensorflow.keras import models, layers*.

c. Proses untuk membuat visualisasi seperti grafik dan plot menggunakan *import matplotlib as plt*.

d. *import cv2* untuk mengimpor pustaka *OpenCV* (*Open Source Computer Vision Library*), yang digunakan untuk berbagai tugas pengolahan citra dan komputer vision.

e. *from sklearn.metrics import confusion_matrix* digunakan untuk menghitung *confusion matrix*, yang menyajikan performa model klasifikasi dalam bentuk matriks.

f. *import seaborn as sns*, dengan mengimpor pustaka Seaborn, yang merupakan pustaka untuk membuat visualisasi statistik yang lebih menarik dan informatif.

Setelah mengimport *library* tang akan digunakan dalam model ini, selanjutnya adalah dengan layer-layer yang akan digunakan. Berikut kode untuk layer model CNN.

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 10

model = models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32, (3, 3), activation = 'relu', input_shape
= input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3, 3), activation =
'relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3, 3), activation =
'relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation = 'relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation = 'relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation = 'relu'),
    layers.Dense(n_classes, activation = 'softmax')
])
model.build(input_shape = input_shape)
```

Source code diatas adalah fungsi yang akan digunakan untuk membuat model CNN dapat diajabarkan sebagai berikut :

1. Menentukan bentuk input yang akan digunakan model, yaitu *input_shape* yang terdiri dari dimensi *batch size*, ukuran citra, dan saluran warna (RGB = 3).
2. Menambahkan *Convolution2D layer*. Digunakan untuk data berupa gambar yang berfungsi untuk mengenali gambar berdasarkan piksel-

piksel yang ada pada gambar dengan menggunakan fungsi activation relu.

3. Selanjutnya dilakukan *Maxpooling2D* untuk mereduksi dimensi gambar. *Maxpooling Layers* merupakan operasi yang digunakan untuk mengambil nilai maksimum dari sekelompok nilai dalam matriks.
4. Proses ini kemudian di ulangi beberapa kali dengan lapisan konvolusi dan *max-pooling* yang berbeda.
5. Pada *layers.Flatten()* akhirnya dilakukan proses perubahan matriks atau array dari *output* lapisan sebelumnya untuk selanjutnya diubah menjadi vektor atau dimensi. Dalam *neural network*, fungsi flatten layer sering digunakan setelah lapisan konvolusi atau *pooling* untuk mengubah *output* dari lapisan tersebut menjadi vektor yang dapat dihubungkan ke lapisan-lapisan selanjutnya seperti lapisan dense (*fully connected*).
6. Kemudian menambahkan *layers.Dense* (*fully connected*) dengan 64 unit dan fungsi aktivasi relu.
7. Yang terakhir, menambahkan lapisan *output* dengan *n_classes* yang merupakan jumlah neuron atau unit dalam lapisan ini dan fungsi *activation='softmax'* yang merupakan fungsi aktivasi yang diterapkan pada setiap neuron dalam lapisan dense untuk klasifikasi.
8. Kemudian *model.build(input_shape=input_shape)* yaitu metode yang digunakan untuk memvalidasi dan mengkonfirmasi bentuk *input* yang sesuai dengan model yang didefinisikan sebelum lanjut ke tahap pelatihan.

Untuk menampilkan ringkasan dari arsitektur model ini yaitu menggunakan fungsi *model.summary()*. Seperti berikut:

```
model.summary()
```

Kode di atas menampilkan struktur pada layer CNN seperti jumlah parameter, dimensi keluaran (*Output Shape*), jumlah parameter per lapisan, tipe lapisan jumlah parameter *trainable* dan *Non-Trainable* yang dapat di lihat pada gambar berikut.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (32, 256, 256, 3)         0

 sequential_1 (Sequential)   (32, 256, 256, 3)         0

 conv2d (Conv2D)             (32, 254, 254, 32)        896

 max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0
 )

 conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

 max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0
 2D)

 conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928

 max_pooling2d_2 (MaxPooling  (32, 30, 30, 64)         0
 2D)

 conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928

 max_pooling2d_3 (MaxPooling  (32, 14, 14, 64)         0
 2D)

 conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928


 max_pooling2d_4 (MaxPooling  (32, 6, 6, 64)           0
 2D)

 flatten (Flatten)           (32, 2304)                0

 dense (Dense)               (32, 64)                  147520

 dense_1 (Dense)             (32, 10)                  650

=================================================================
Total params: 278,346
Trainable params: 278,346
Non-trainable params: 0
_________________________________________________________________
```

Gambar 13. Hasil arsitektur model CNN

Dari hasil ringkasan arsitektur model CNN di atas, *model: "sequential_2"* menunjukkan penggunaan sequential model, *layer(type)* menunjukkan jenis lapisan dalam model, *output shape* menghasilkan bentuk keluaran dari setiap

lapisan, dan *param #* menunjukkan jumlah parameter yang diatur dalam lapisan tersebut.

E. Pelatihan model dengan metode *Convolutional Neural Network* (CNN)

Metode CNN merupakan salah satu metode *deep learning* yang mampu melakukan proses pembelajaran mandiri untuk pengenalan, ekstraksi, dan klasifikasi objek citra. Untuk melatih suatu model dengan menggunakan metode CNN, tahap selanjutnya setelah melakukan pengolahan dataset yang meliputi pembagian kelas dataset, pre-processing dataset, hingga pelabelan dataset, selanjutnya adalah mempersiapkan data untuk mulai pelatihan model. Source code persiapan pelatihan dataset.

```
train_dataset =
train_dataset.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE
)
validation_dataset =
validation_dataset.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUT
OTUNE)
test_dataset =
test_dataset.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

)
```

Dengan kode di atas, sistem akan mengacak data yang ada dalam dataset dengan menggunakan fungsi *shuffle,* dan ukuran buffer 1000 yang mengacu pada seberapa banyaj elemen yang akan diambil. Selanjutnya adalah melakukan preprocessing dataset, menggunakan kode berikut.

```
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE,
IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1./255)
])
```

Pada kode di atas *Layer.experimental.preprocessing* yang terdiri dari 2 lapisan yaitu *resizing* dan *rescaling,* merupakan pendekatan umum untuk mempersiapkan data gambar sebelum melatih atau menguji model. Resizing dan rescaling berfungsi untuk mengubah skala dan ukuran objek, data atau gambar.

Selanjutnya tahap augmentation yang merupakan teknik dalam *machine learning*, yang membantu model mengenali dataset dengan baik dan mengurangi resiko *overfitting*. Berikut source code tahap augmentation:

```
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_a
nd_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2)
])
```

Tahap selanjutnya yaitu konfigurasi untuk model, dengan source code :

```
model.compile(
    optimizer = 'adam',
    loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits =
False),
    metrics = ['accuracy']
)
```

Kode di atas dapat dijabarakan dengan :

1. *Optimizer (optimizer),* algoritma yang digunakan untuk mengoptimalkan parameter-model selama proses pelatihan. menggunakan *optimizer 'adam',* yang merupakan optimizer dalam *machine learning* untuk tugas pelatihan model *neural network.*
2. *Loss Function (loss)* fungsi yang digunakan untuk mengukur seberapa baik dengan data pelatihan dan seberapa jauh prediksi model. *SparseCategoricalCrossentropy* umumnya digunakan jika terjadi masalah dalam klasifikasi.
3. *Metrics (metrics)* adalah metrik yang akan dievaluasi selama pelatihan dan evaluasi model untuk mengukur kinerja model. Dalam akurasi *('accuracy'),* yang merupakan persentase data yang diprediksi dengan benar oleh model dari total data.

Untuk pelatihan model, tahap seselanjutnya dengan melatih model menggunakan *TensorFlow's Keras API* dengan menggunakan data yang telah disiapkan sebelumnya. Source code dalam tahap ini adalah

```
history = model.fit(
    train_dataset,
    epochs = 10,
    batch_size = BATCH_SIZE,
    verbose = 1,
    validation_data = validation_dataset,
)
```

1. *train_dataset*, dataset pelatihan yang akan digunakan untuk melatih model.
2. *Epochs*, merupakan jumlah siklus melalui seluruh dataset pelatihan yang akan digunakan selama pelatihan. Kode diatas mengatur jumlah epoch menjadi 10, yang berarti dataset pelatihan akan digunakan 10 kali selama pelatihan.
3. *batch_size*, yaitu jumlah sampel yang akan digunakan dalam setiap iterasi pelatihan. Proses pelatihan biasanya dilakukan dengan memproses data dalam batch-batch kecil untuk efisiensi komputasi. Dalam model ini *batch size* yang digunakan adalah 64.
4. *Verbose*, berfungsi mengontrol seberapa banyak informasi yang akan ditampilkan selama proses pelatihan. Jika diatur sebagai 1, maka akan dilihat progress bar yang menunjukkan perkembangan pelatihan pada setiap epoch.
5. *validation_data*, dataset validasi akan digunakan untuk mengukur kinerja model selama proses training dan membantu memantau apakah model sedang overfitting atau tidak.

Setelah menjalankan kode di atas, model akan melalui proses training selama 10 epoch dengan menggunakan dataset pelatihan yang diberikan. Pada setiap *epoch*, model akan memproses *batch-batch* data dan mencoba meminimalkan nilai fungsi.

Pada penelitian ini uji coba yang dilakukan yaitu pada *batch size* dan *splitting data* dengan masing-masing 1 kali pengujian untuk mendapatkan model klasifikasi yang terbaik menggunakan 5.160 dataset dalam 10 kelas. Pembagian data yang di uji coba yaitu dapat dilihat pada table berikut.

Table 4. Hasil percobaan perbandingan.

epoch	***Batch size***	***Splitting data***	**Akurasi**
10	16	80:10:10	93,52 %
	16	70:15:15	93, 4%
	32	80:10:10	98,50 %
	32	70:15:15	97,65%
	64	80:10:10	97,34 %
	64	70:15:15	92,72 %
	128	80:10:10	96,48%
	128	70:15:15	96,26%

Pada percobaan diatas dapat dijabarkan:

1. Pada percobaan pertama dengan *batch size* 16 dengan *splitting* data 80:10:10 didapatkan akurasi sebesar 93,52% dengan proses training selama 10 *epoch.* Akurasi 95,36% didapatkan pada *epoch* 5 lalu turun menjadi 85,76 pada *epoch* 6 kemudan naik Kembali pada *epoch* 7 sampai seterusnya hingga di dapatkan hasil akurasi 93,52% pada *epoch* 10.
2. Percobaan kedua dengan dengan *batch size* 16 dengan *splitting* data 70:15:15 dengan proses training selama 10 *epoch*. Akurasi 91,13% didaptkan pada *epoch* 2 namun mengalami penuruan pada *epoch* 3 dengan akurasi 89,86, kemudan naik lagi pada *epoch* 4 dan sterusnya hingga didapatkan akurasi 93,4% pada *epoch* 10.
3. Percobaan ketiga dengan *batch size* 32 dengan *splitting* data 80:10:10, akurasi 97,29% didapatkan pada *epoch* 3 kemudian pada *epoch* 4 sampai

seterusnya tidak stabil karna akurasi naik turun, hingga didaptkan akurasi 98,50% pada *epoch* 10.

4. Percobaan keempat dengan *batch size* 32 dengan *splitting* data 70:15:15 dengan proses training selama 10 *epoch*. Akurasi 91,18% didaptkan pada *epoch* 1 kemudian pada *epoch* 2 sampai seterusnya tidak stabil karna akurasi naik turun, hingga didaptkan akurasi 97,65% pada *epoch* 10.
5. Percobaan kelima dengan *batch size* 64 dengan *splitting* data 80:10:10 didapatkan akurasi sebesar 97,34% dengan proses training selama 10 *epoch.* Akurasi 95,39 didapatkan pada *epoch* 2 lalu turun menjadi 94,29% pada *epoch* 3 kemudan naik Kembali pada *epoch* 4 sampai seterusnya hingga di dapatkan hasil akurasi 97,34% pada *epoch* 10.
6. Percobaan keenam dengan dengan *batch size* 64 dengan *splitting* data 70:15:15 dengan proses training selama 10 *epoch*. Akurasi 94,49% didaptkan pada *epoch* 2 meningkat pada *epoch* 3 dengan akurasi 95,20%, kemudian pada *epoch* 4 sampai seterusnya tidak stabil karna akurasi naik turun, hingga didaptkan akurasi 92,72% pada *epoch* 10.
7. Percobaan ketujuh dengan *batch size* 128 dengan *splitting* data 80:10:10, akurasi 95,14% didapatkan pada *epoch* 4 kemudian pada *epoch* 5 sampai seterusnya tidak stabil karna akurasi naik turun, hingga didaptkan akurasi 96,48% pada *epoch* 10.
8. Percobaan terakhir dengan dengan *batch size* 128 dengan *splitting* data 70:15:15 dengan proses training selama 10 *epoch*. Akurasi 91,24% didaptkan pada *epoch* 2 kemudian pada *epoch* 3 sampai seterusnya tidak stabil karna akurasi naik turun, hingga didaptkan akurasi 96,26% pada *epoch* 10.

Berdasarkan 8 kali percobaan diatas, dapat diketahui akurasi terbaik dari pengujian *batch size* dan *splitting data* yaitu dengan hasil 98,50 % dari pengujian *batch size* 32, dengan *splitting data* 80:10:10 dan training menggunakan 10 *epoch.* Dalam uji coba tersebut dapat di lihat perbandingan akurasi yang paling mendekati adalah pada *batch size* 32 dengan *splitting* data yang berbeda. Pembagian data 80:10:10 cenderung memberikan hasil yang lebih baik daripada

pembagian 70:15:15 pada semua kombinasi ukuran *batch.* Ini menunjukkan bahwa memiliki lebih banyak data pelatihan (80%) dapat membantu model untuk belajar pola dengan lebih baik daripada jumlah data pelatihan yang lebih kecil (70%).

F. Hasil Pelatihan Model

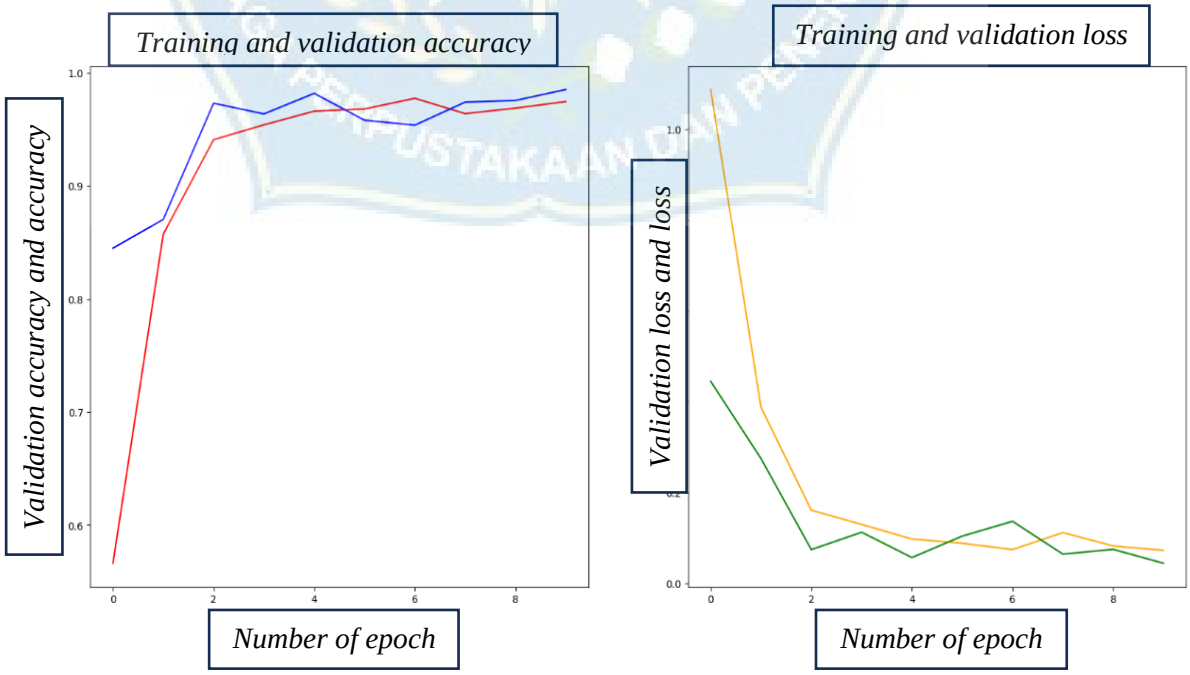
Berdasarkan hasil proses untuk memuat dataset didapatkan jumlah data sebanyak 5.160 data. Setelah itu data akan melalui proses *batch size* dengan menggunakan *batch size* 32. Jadi 5.160 dataset dibagi per *batch* dengan hasil 161,25 *batch*. Kemudian pada proses *splitting data*, data tersebut dibagi menjadi tiga yaitu data *training*, data *validation*, dan data *testing*. Dengan skenario pembagian data 80:10:10 yaitu dengan 80% data training, 10% data validation, dan 10% data testing dengan menggunakan *batch size* 32. Hasil yang diperoleh yaitu, jumlah dataset 5.160/32 *batch size* adalah 161,25 batch dan dibulatkan menjadi 161 *batch*, lalu sisa *batch* 0,25 dibulatkan menjadi 1 tetapi gambarnya tidak cukup 32 gambar dan otomatis sisa *batch* akan masuk ke data testing. Jadi hasil setelah splitting data yaitu data training menjadi 130 *batch*, *validation* 16 *batch* dan *testing* menjadi 17 *batch*. Data tersebut akan melalui proses pelatihan menggunakan 10 *epoch*. *Epoch* merupakan *hyperparameter* yang menentukan berapa kali *algoritma deep learning* bekerja melewati seluruh dataset baik secara forward maupun backward. Dalam istilah yang lebih sederhana, Satu *epoch* tercapai ketika semua *batch* telah berhasil dilewatkan melalui jaringan saraf satu kali. Dalam contoh kasus di bawah ini, 1 *epoch* tercapai ketika 130 *batch* sampel data *training* selesai diproses. Hasil dari proses pelatihan selama 10 *epoch* yang di peroleh adalah:

```
Epoch 1/10
130/130 [==============================] - 764s 2s/step - loss: 1.0877 -
accuracy: 0.5662 - val_loss: 0.4449 - val_accuracy: 0.8448
Epoch 2/10
130/130 [==============================] - 263s 2s/step - loss: 0.3878 -
accuracy: 0.8571 - val_loss: 0.2747 - val_accuracy: 0.8702
Epoch 3/10
130/130 [==============================] - 265s 2s/step - loss: 0.1609 -
accuracy: 0.9407 - val_loss: 0.0741 - val_accuracy: 0.9729
Epoch 4/10
130/130 [==============================] - 291s 2s/step - loss: 0.1295 -
accuracy: 0.9537 - val_loss: 0.1123 - val_accuracy: 0.9634
```

```
Epoch 5/10
130/130 [==============================] - 296s 2s/step - loss: 0.0974 -
accuracy: 0.9659 - val_loss: 0.0567 - val_accuracy: 0.9816
Epoch 6/10
130/130 [==============================] - 262s 2s/step - loss: 0.0881 -
accuracy: 0.9678 - val_loss: 0.1038 - val_accuracy: 0.9579
Epoch 7/10
130/130 [==============================] - 286s 2s/step - loss: 0.0741 -
accuracy: 0.9772 - val_loss: 0.1364 - val_accuracy: 0.9535
Epoch 8/10
130/130 [==============================] - 255s 2s/step - loss: 0.1116 -
accuracy: 0.9637 - val_loss: 0.0642 - val_accuracy: 0.9738
Epoch 9/10
130/130 [==============================] - 255s 2s/step - loss: 0.0821 -
accuracy: 0.9685 - val_loss: 0.0746 - val_accuracy: 0.9753
Epoch 10/10
 130/130 [==============================] - 292s 2s/step - loss: 0.0724 -
        accuracy: 0.9743 - val_loss: 0.0440 - val_accuracy: 0.9850
```

Gambar 14. Hasil dari proses training selama 10 *epoch*

Hasil akurasi yang didapatkan setelah proses training dengan 10 *epoch* adalah 98%. Akurasi model dari data *training* dan data *testing* jika dilihat dari *epoch* 1 sampai 10 dominan mengalami peningkatan, namun ada juga penurunan pada *epoch* tertentu, peningkatan dan penurunan akurasi model terjadi karena data yang diprediksi setiap iterasi berbeda atau naik turun. Untuk melihat bagaimana akurasi model berubah selama pelatihan dan validasi dalam beberapa epoch, ditunjukkan dengan grafik berikut.



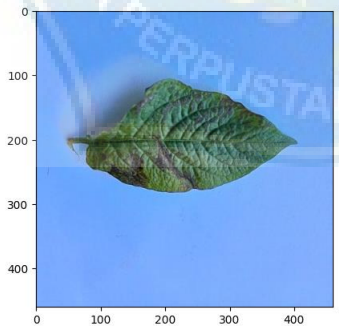
Gambar 15. Grafik hasil training dataset

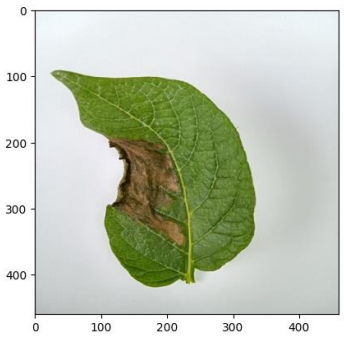
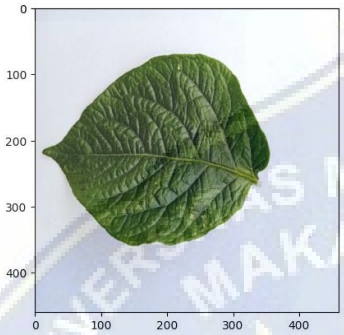
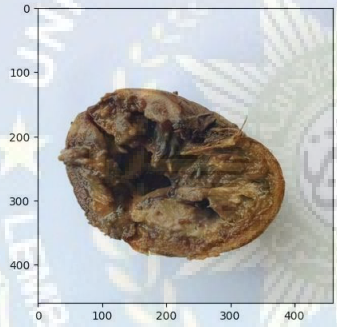
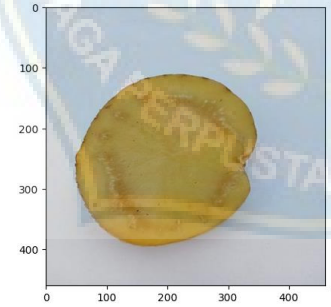
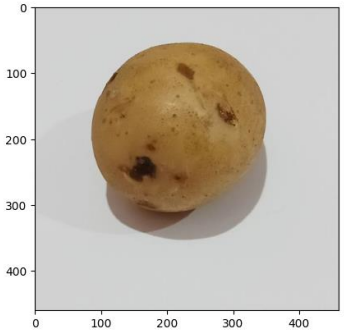
Pada grafik diatas pada bagian sumbu y (*vertikal*) untuk mengetahui nilai atau tingkat akurasi pada proses pelatihan, dan sumbu x (*horizontal*) untuk mengetahui jumlah *epoch* yang digunakan pada proses *training.* Label berwarna merah (-) menunjukkan proses *training accuracy,* label berwarna biru (-) menunjukkan proses *validation accuracy,* label berwarna orange (-) menunjukkan *training loss,* dan label hijau (-) menunjukkan *validation loss.*

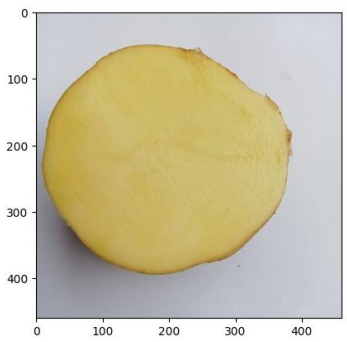
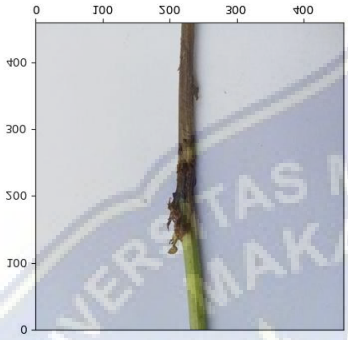
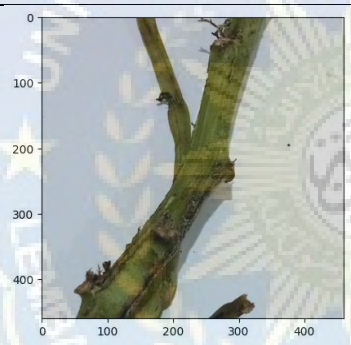
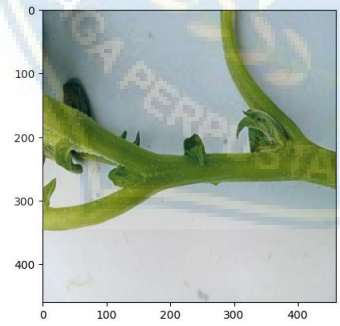
Grafik di hasil training *batch size* 32 dengan *splitting* data 80:10:10 menunjukkan bahwa grafik pertama menunjukkan nilai akurasi yang semakin meningkat dan dari grafik kedua dapat dilihat untuk nilai loss mengalami penurunan, yang berarti model bekerja dengan baik dengan proses training selama 10 *epoch*. Dari grafik diatas juga dapat dilihat akurasi tidak stabil atau naik turun selama proses training.

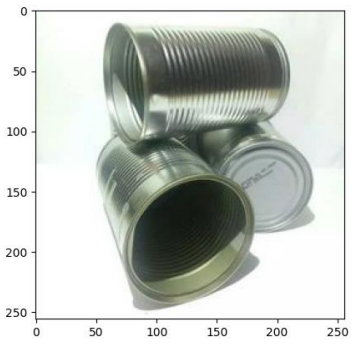
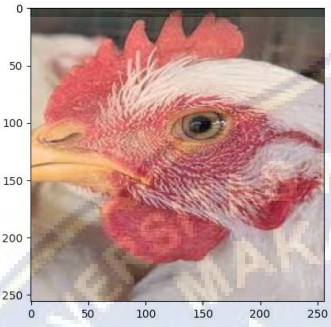
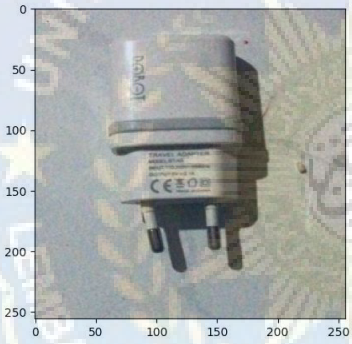
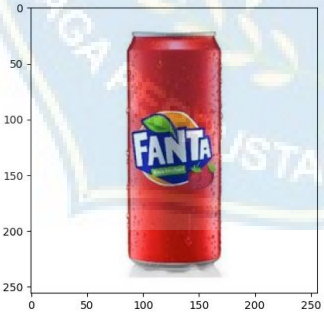
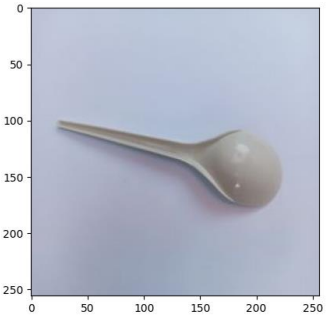
Selanjutnya melakukan percobaan training dengan menggunakan 10 gambar tanaman kentang dan 10 gambar lainnya yang bukan gambar tanaman kentang.

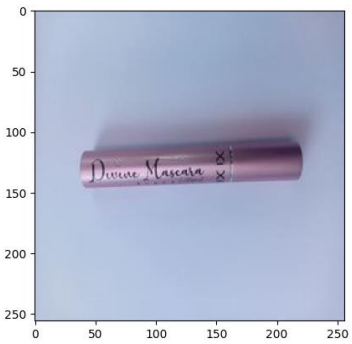
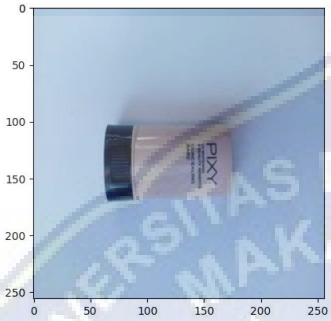
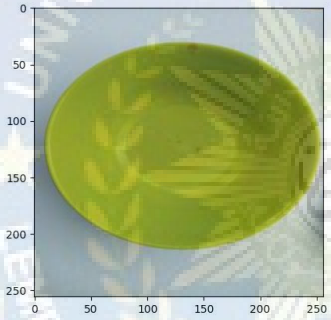
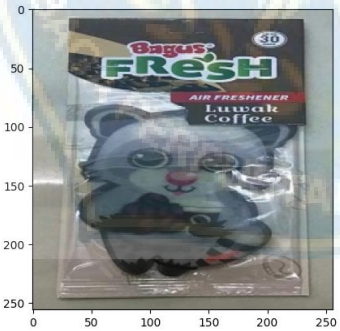
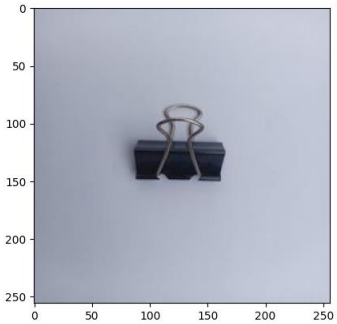
Table 5. Uji coba prediksi gambar

No.	Gambar	Hasil prediksi
1.		1/1 [==========] - 0s 184ms/step 1/1 [==========] - 0s 25ms/step Prdeict class: **Early--Blight**

2.		1/1 [==========] - 0s 32ms/step 1/1 [==========] - 0s 33ms/step Predicted class: **Late--Blight**
3.		1/1 [==========] - 0s 29ms/step 1/1 [==========] - 0s 32ms/step Predicted class: **Daun--Healthy**
4.		1/1 [==========] - 0s 24ms/step 1/1 [==========] - 0s 22ms/step Predicted class: **Bacterial--Soft--Rot**
5.		1/1 [==========] - 0s 19ms/step 1/1 [==========] - 0s 19ms/step Predicted class: **Bacterial--Ring-Rot**
6.		1/1 [==========] - 0s 36ms/step 1/1 [==========] - 0s 30ms/step Predicted class: **Common--Scab**

7.		1/1 [==========] - 0s 21ms/step 1/1 [==========] - 0s 30ms/step Predicted class: **Umbi--Healthy**
8.		1/1 [==========] - 0s 23ms/step 1/1 [==========] - 0s 22ms/step Predicted class: **Fusarium--wilt**
9.		1/1 [==========] - 0s 23ms/step 1/1 [==========] - 0s 22ms/step Predicted class: **Stem--Rust**
10.		1/1 [==========] - 0s 23ms/step 1/1 [==========] - 0s 23ms/step Predicted class: **Batang--Healthy**

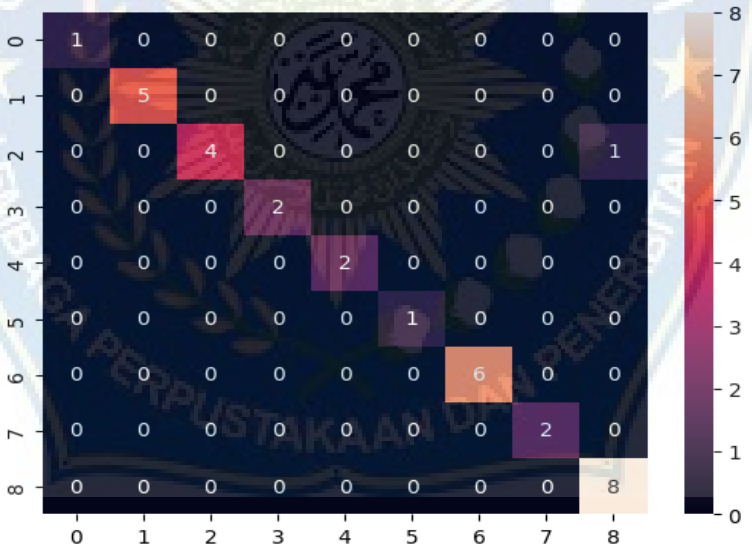
11.		1/1 [==========] - 0s 32ms/step 1/1 [==========] - 0s 32ms/step Predicted class: **Bukan--Penyakit--Kentang**
12.		1/1 [==========] - 0s 22ms/step 1/1 [==========] - 0s 20ms/step Predicted class: **Bukan--Penyakit--Kentang**
13.		1/1 [==========] - 0s 22ms/step 1/1 [==========] - 0s 23ms/step Predicted class: **Bukan--Penyakit--Kentang**
14.		1/1 [==========] - 0s 26ms/step 1/1 [==========] - 0s 21ms/step Predicted class: **Bukan--Penyakit--Kentang**
15.		1/1 [==========] - 0s 23ms/step 1/1 [==========] - 0s 21ms/step Predicted class: **Bukan--Penyakit--Kentang**

16.		1/1 [==========] - 0s 31ms/step 1/1 [==========] - 0s 35ms/step Predicted class: **Bukan--Penyakit--Kentang**
17.		1/1 [==========] - 0s 26ms/step 1/1 [==========] - 0s 25ms/step Predicted class: **Bukan--Penyakit--Kentang**
18.		1/1 [==========] - 0s 24ms/step 1/1 [==========] - 0s 24ms/step Predicted class: **Bukan--Penyakit--Kentang**
19.		1/1 [==========] - 0s 22ms/step 1/1 [==========] - 0s 22ms/step Predicted class: **Bukan--Penyakit--Kentang**
20.		1/1 [==========] - 0s 24ms/step 1/1 [==========] - 0s 20ms/step Predicted class: **Bukan--Penyakit--Kentang**

Setelah dilakukan 20 kali testing, mesin dapat memprediksi 10 data dengan benar dan 10 data bukan penyakit tanaman kentang. Mesin dapat meprediksi benar jika gambar yang dimasukkan sesuai dengan yang di pelajari pada saat *training* yaitu data penyakit kentang, jika ditesting dengan data diluar penyakit kentang maka mesin akan menghasilkan prediksi dengan bukan penyakit kentang.

G. Pengujian menggunakan *Confusion Matrix*

Pada tahap pengujian mengunakan metode *confusion matrix*, hasil akurasi yang telah dihasilkan oleh model dengan metode CNN. Model klasifikasi digunakan karena dapat memperlihatkan bagaimana model yang telah dibuat dapat mengambil suatu keputusan di dunia nyata. Berikut adalah hasil *confusion matrix* dari proses training model.



Gambar 16. Hasil pengujian *Confusion Matrix*

Dari gambar diatas, terdapat 1 FP dan FN, yang terdapat pada *class* 2 dan *class* 8, pada *class* 2 terdapat FN dimana mesin salah menklasifikasikan gambar tidak termasuk kategori *class* 2, padahal aktualnya gambar tesebut masuk ke *class* 2. Sedangkan di class 8 terdapat FP yang dimana mesin salah menklasifikasikan

gambar masuk class 8, padahal aktualnya gamabr tersebut tidak masuk kategori class 8.

Confusion matrix disini digunakan untuk menghitung akurasi dari model. *Accuracy* adalah proporsi data yang diklasifikasikan dengan benar dari keseluruhan data. Berikut contoh perhitungan untuk mendapatkan nilai akurasi dengan pengujian *confusion matrix.*

$$Accuracy = \frac{TN + TP}{Total}$$

$$= \frac{31}{32} = 0.96875$$

$$= 96{,}88\%$$

Jadi hasil pengujian yang dihasilkan dari model CNN menggunakan *batch size* 32 dengan *splitting* data 80:10:10 dan dibagi menjadi 10 kelas mendapatkan akurasi sebesar 96,88%.

BAB V
PENUTUP

A. Kesimpulan

Dari penelitian Sistem pakar diagnosa penyakit tanaman kentang menggunakan metode pengenalan dengan gejala dengan citra, menggunakan model CNN dapat disimpulkan bahwa dalam penelitian ini pada sebagian besar skenario, pembagian data 80:10:10 (80% *pelatihan*, 10% *validasi*, 10% *pengujian*) cenderung memberikan hasil yang lebih baik daripada pembagian 70:15:15. Pada sebagian besar ukuran *batch* yang diuji (16, 32, 64, dan 128), pembagian data 80:10:10 memberikan performa yang lebih baik daripada pembagian 70:15:15. Namun, perbedaan antara keduanya bisa bervariasi tergantung pada ukuran *batch* tertentu. Ukuran *batch* juga memiliki dampak pada performa model. Pada beberapa pengujian, ukuran *batch* yang lebih besar menghasilkan performa yang lebih rendah, sedangkan ukuran batch yang lebih kecil seperti 32 memberikan hasil yang lebih baik. Akurasi tertinggi tercapai pada skenario pembagian data 80:10:10 dengan ukuran batch 32, yaitu sekitar 98,50%. Pada beberapa skenario, pembagian data 70:15:15 memberikan hasil yang lebih rendah, bahkan sampai di bawah 93% pada beberapa pengujian.

B. Saran

1. Pada penelitian ini, hal-hal yang dapat dilakukan untuk penelitian mendatang yaitu dengan inovasi dalam metode, mempertimbangkan inovasi dalam metode atau teknik yang akan digunakan. Apakah ada pendekatan baru yang dapat diterapkan untuk mencapai hasil yang lebih baik atau lebih efisien atau dengan studi perbandingan. Bandingkan beberapa metode atau pendekatan yang berbeda untuk melihat mana yang paling efektif dalam meningkatkan akurasi selain pada *batch size* dan *splitting* data**.**
2. Menggunakan model algoritma serta pengujian yang lebih bervariasi yang diharapkan dapat lebih baik dalam memprediksi gambar dengan hasil akurasi yang lebih memuaskan.

DAFTAR PUSTAKA

Abdurahman, M. (2018). Sistem Informasi Data Pegawai Berbasis Web Pada Kementerian Kelautan Dan Perikanan Kota Ternate. *Jurnal Ilmiah ILKOMINFO - Ilmu Komputer & Informatika*, *1*(2), 70–78. https://doi.org/10.47324/ilkominfo.v1i2.10

Ahmad, N., & Iskandar. (2020). Metode Forward Chaining untuk Deteksi Penyakit Pada Tanaman Kentang. *JINTECH: Journal Of Information Technology*, *1*(2), 7–20. https://doi.org/10.22373/jintech.v1i2.592

Aisyah, R., Salsabila, A. M., Faiqah, J. A., Fadia, A. N., & Fitrah, G. (2022). *Inventarisasi Bakteri Penyebab Penyakit pada Umbi Kentang (Solanum tuberosum L .) dan Cara Penanggulangannya Inventory of Disease Cause by Bacteria in Potato Tuber (Solanum tuberosum L .) and How to Control it*. 44–53.

Burch dan Grudnitski dalam (Fauzi, 2017:19-21). (2019). Bab II Landasan Teori. *Journal of Chemical Information and Modeling*, 4–10.

Dewi, S. R. (2018). Deep Learning Object Detection Pada Video. *Deep Learning Object Detection Pada Video Menggunakan Tensorflow Dan Convolutional Neural Network*, 1–60. https://dspace.uii.ac.id/bitstream/handle/123456789/7762/14611242_Syarifah Rosita Dewi_Statistika.pdf?sequence=1

Dewi, T. S., Purwadi, P., & Prayuda, J. (2020). Sistem Pakar Mendiagnosa Penyakit Busuk Daun Pada Tanaman Solanum Tubersum (Kentang Berumbi Merah) Menggunalan Metode Teorema Bayes. *Jurnal Cyber Tech*, *3*(8), 1332–1341. https://ojs.trigunadharma.ac.id/index.php/jct/article/view/4530

Fuadi, A., & Suharso, A. (2022). Perbandingan Arsitektur Mobilenet Dan Nasnetmobile Untuk Klasifikasi Penyakit Pada Citra Daun Kentang. *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, *7*(3), 701–710. https://doi.org/10.29100/jipi.v7i3.3026

Hidayatulloh, K., MZ, M. K., & Sutanti, A. (2020). Perancangan Aplikasi Pengolahan Data Dana Sehat Pada Rumah Sakit Umum Muhammadiyah Metro. *Jurnal Mahasiswa Ilmu Komputer*, *1*(1), 18–22. https://doi.org/10.24127/.v1i1.122

Iii, B. A. B. (2019). *Analisis Gaya Hidup Imitasi Remaja dalam Komunitas Maranatha Youthteen di Ungaran*. 28–34.

Karsito, & Susanti, S. (2019). Klasifikasi Kelayakan Peserta Pengajuan Kredit Rumah Dengan Algoritma Naïve Bayes Di Perumahan Azzura Residencia. *Jurnal Teknologi Pelita Bangsa*, *9*, 43–48.

Kusumadewa, C. C., & Supatman, S. (2018). Identifikasi Citra Daun Teh

Menggunakan Metode Histogram untuk Deteksi Dini Serangan Awal Hama Empoasca. *JMAI (Jurnal Multimedia & Artificial Intelligence)*, *2*(1), 27–36. https://doi.org/10.26486/jmai.v2i1.71

Lesmana, A. M., Fadhillah, R. P., & Rozikin, C. (2022). Identifikasi Penyakit pada Citra Daun Kentang Menggunakan Convolutional Neural Network (CNN). *Jurnal Sains dan Informatika*, *8*(1), 21–30. https://doi.org/10.34128/jsi.v8i1.377

Margareth, D., Astarini, I. A., & Temaja, I. G. R. M. (2018). DETECTION AND ELIMINATION VIRUS ON POTATO (Solanum tuberosum L.). *International Journal of Biosciences and Biotechnology*, *5*(2), 92. https://doi.org/10.24843/ijbb.2018.v05.i02.p01

Maulana, F. A., Hidayat, N., & Wijoyo, S. H. (2018). Diagnosis Penyakit Pada Bawang Merah Dengan Menggunakan Metode Fuzzy Tsukamoto (Studi Kasus: Uptd. Pembibitan Ternak Dan Hijauan Makanan Ternak Kec …. ... *Teknologi Informasi dan Ilmu …*, *3*(1), 220–226. http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/4071

Nauval, K. I., & Lestari, S. (2022). Implementasi Deteksi Objek Penyakit Daun Kentang dengan Metode Convolutional Neutral Network. *Jurnal Aplikasi Teknologi Informasi dan Manajemen (JATIM)*, *3*(2), 136–149. https://doi.org/10.31102/jatim.v3i2.1576

Ningsih, dahliatul fitriyah. (2021). *KLASIFIKASI JENIS PENYAKIT DAUN KENTANG MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN) MODEL ARSITEKTUR GOOGLENET*. *14*(1), 1–13.

Rabbani, R., Wahidah, I., & Santoso, I. H. (2021). Klasifikasi Data Deteksi Jatuh Menggunakan Machine Learning dengan Algoritma Adaptive Boosting (AdaBoost). *e-Proceeding of Engineering*, *8*(5), 5053–5063.

Rakhmawati, P. U., Pranoto, Y. M., & Setyati, E. (2018). Klasifikasi Penyakit Daun Kentang Berdasarkan Fitur Tekstur dan Fitur Warna Menggunakan Support Vector Machine. *Seminar Nasional Teknologi dan Rekayasa (SENTRA)*, 1–8.

Ridho Handoko, M., & Neneng. (2021). Sistem Pakar Diagnosa Penyakit Selama Kehamilan Menggunakan Metode Naive Bayes Berbasis Web. *Jurnal Teknologi dan Sistem Informasi (JTSI)*, *2*(1), 50–58. http://jim.teknokrat.ac.id/index.php/JTSI

Rilo Pambudi, A., Garno, & Purwantoro. (2020). JIP (Jurnal Informatika Polinema) DETEKSI KEASLIAN UANG KERTAS BERDASARKAN WATERMARK DENGAN PENGOLAHAN CITRA DIGITAL. *Jurnal Informatika Polinema*, *6*(4), 69–74.

Rozaqi, A. J., Sunyoto, A., & Arief, M. rudyanto. (2021). Deteksi Penyakit Pada Daun Kentang Menggunakan Pengolahan Citra dengan Metode

Convolutional Neural Network. *Creative Information Technology Journal*, *8*(1), 22. https://doi.org/10.24076/citec.2021v8i1.263

Santi, I. H., & Andari, B. (2019). Sistem Pakar Untuk Mengidentifikasi Jenis Kulit Wajah dengan Metode Certainty Factor. *INTENSIF: Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi*, *3*(2), 159. https://doi.org/10.29407/intensif.v3i2.12792

L A M P I R A N

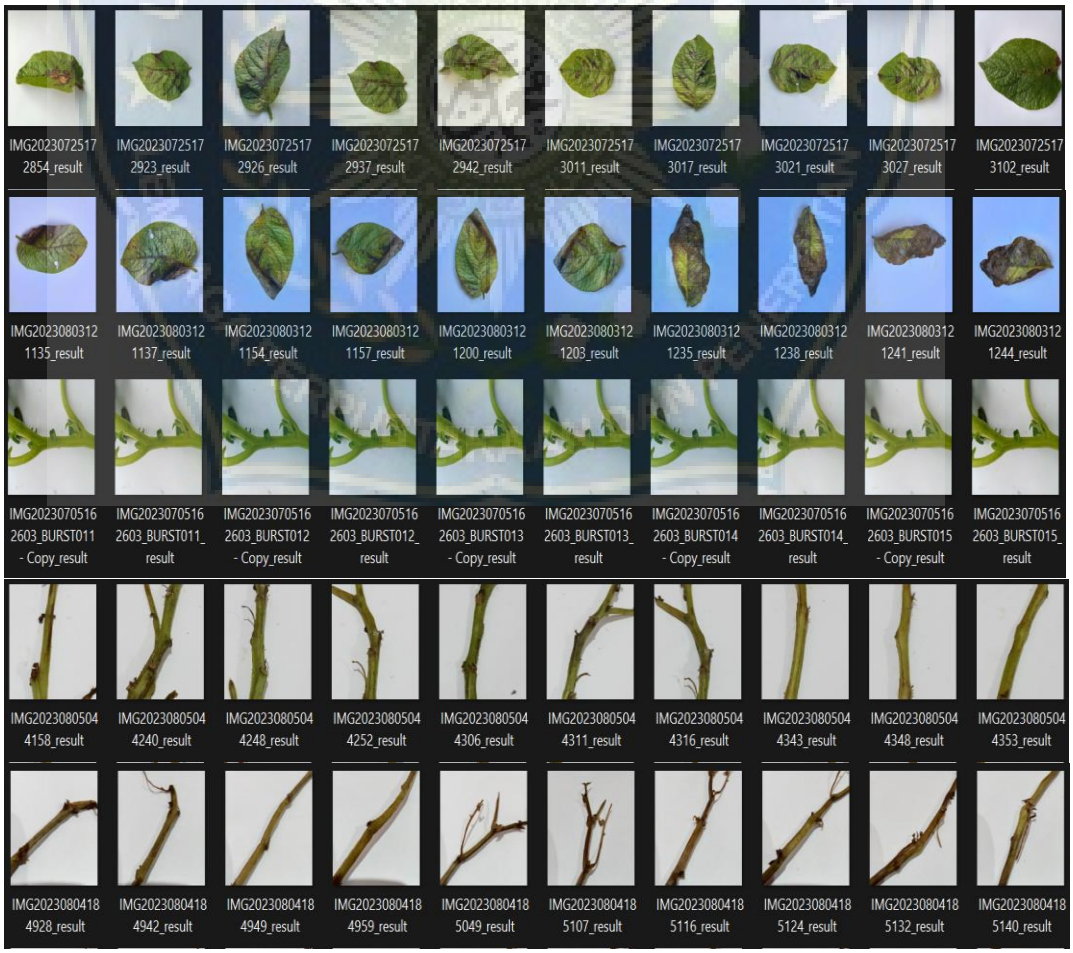
Lampiran 1. Nama penyakit tanaman kentang

No.	**Nama Penyakit Kentang**	**Gambar Penyakit**
1.	Daun sehat	
2.	*Early blight*	
3.	*Late blight*	
4.	Batang sehat	
5.	*Stem rust*	

6.	*Fusarium wilt*	
7.	Umbi sehat	
8.	*Common scab*	
9.	*Bacterial ring rot*	
10.	*Bacterial soft rot*	

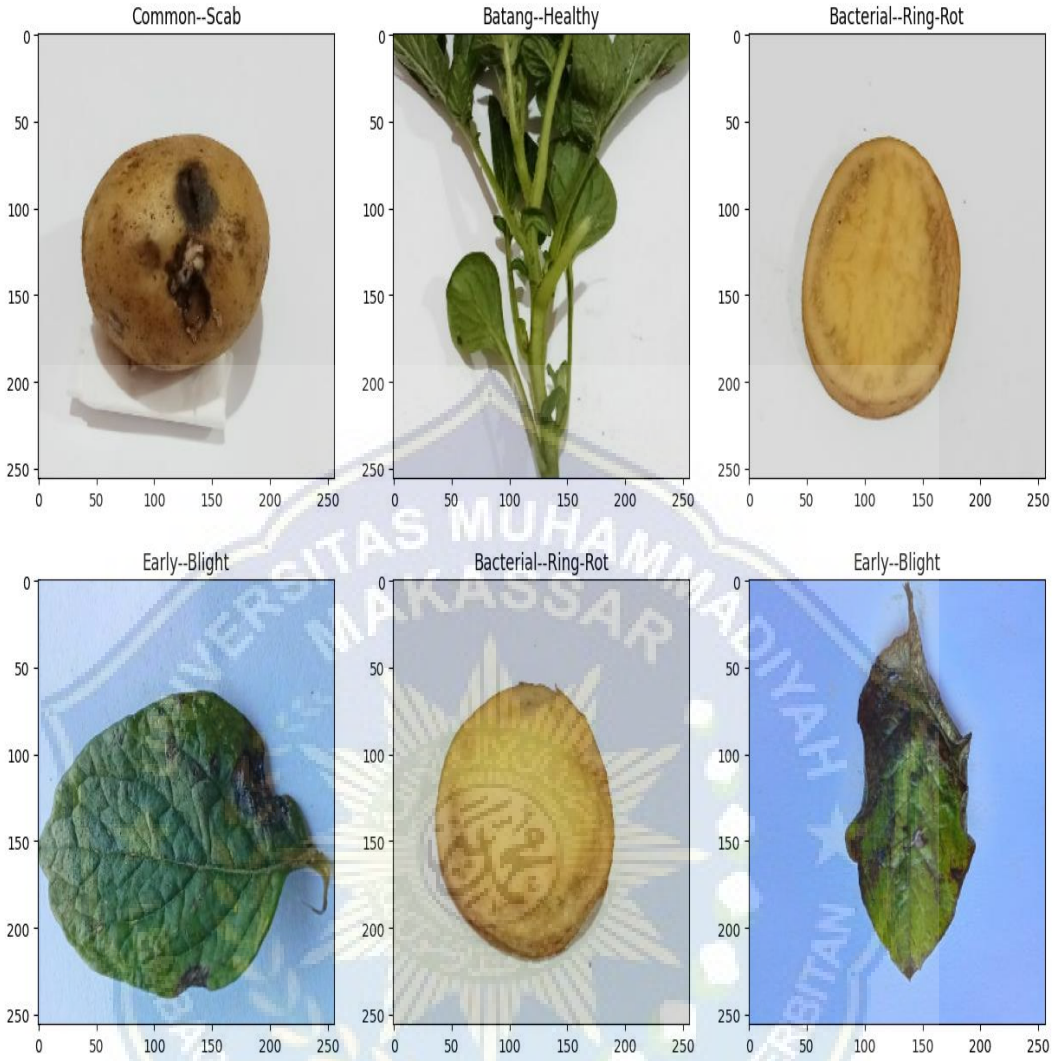
Lampiran 2. Jumlah dataset setiap penyakit

No.	Nama Penyakit	Jumlah Data
1.	Daun Sehat	500
2.	*Early Blight*	600
3.	*Late Blight*	600
4.	Batang Sehat	500
5.	*Stem Rust*	600
6.	*Fusarium wilt*	600
7.	Umbi Sehat	240
8.	*Common Scab*	500
9.	*Bacterial Ring Rot*	500
10.	*Bacterial Soft Rot*	500
Jumlah		**5.160**

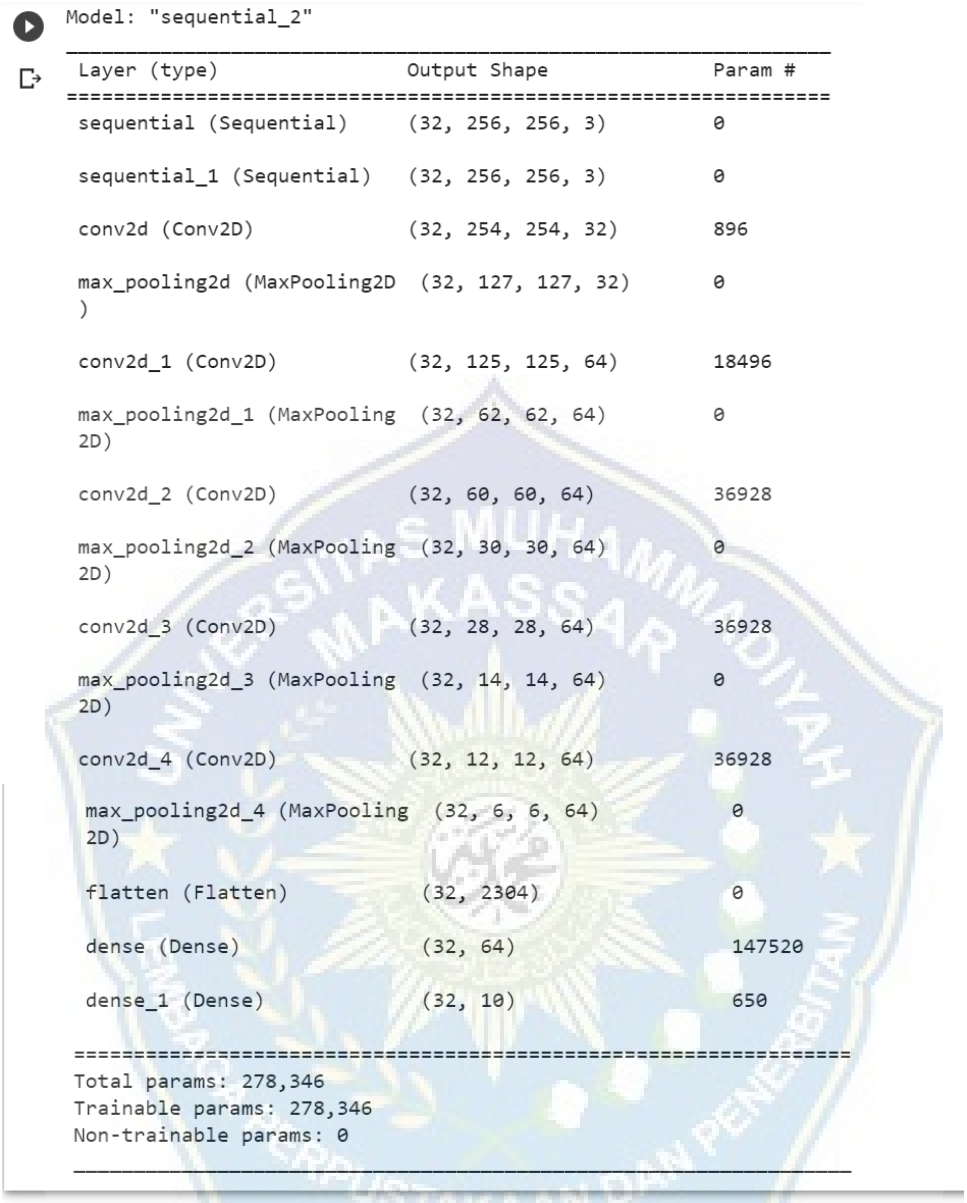


Lampiran 3. Hasil Pengumpulan Dataset

```
['Bacterial--Ring-Rot',
 'Bacterial--Soft--Rot',
 'Batang--Healthy',
 'Common--Scab',
 'Daun--Healthy',
 'Early--Blight',
 'Fusarium--wilt',
 'Late--Blight',
 'Stem--Rust',
 'Umbi--Healthy']
```

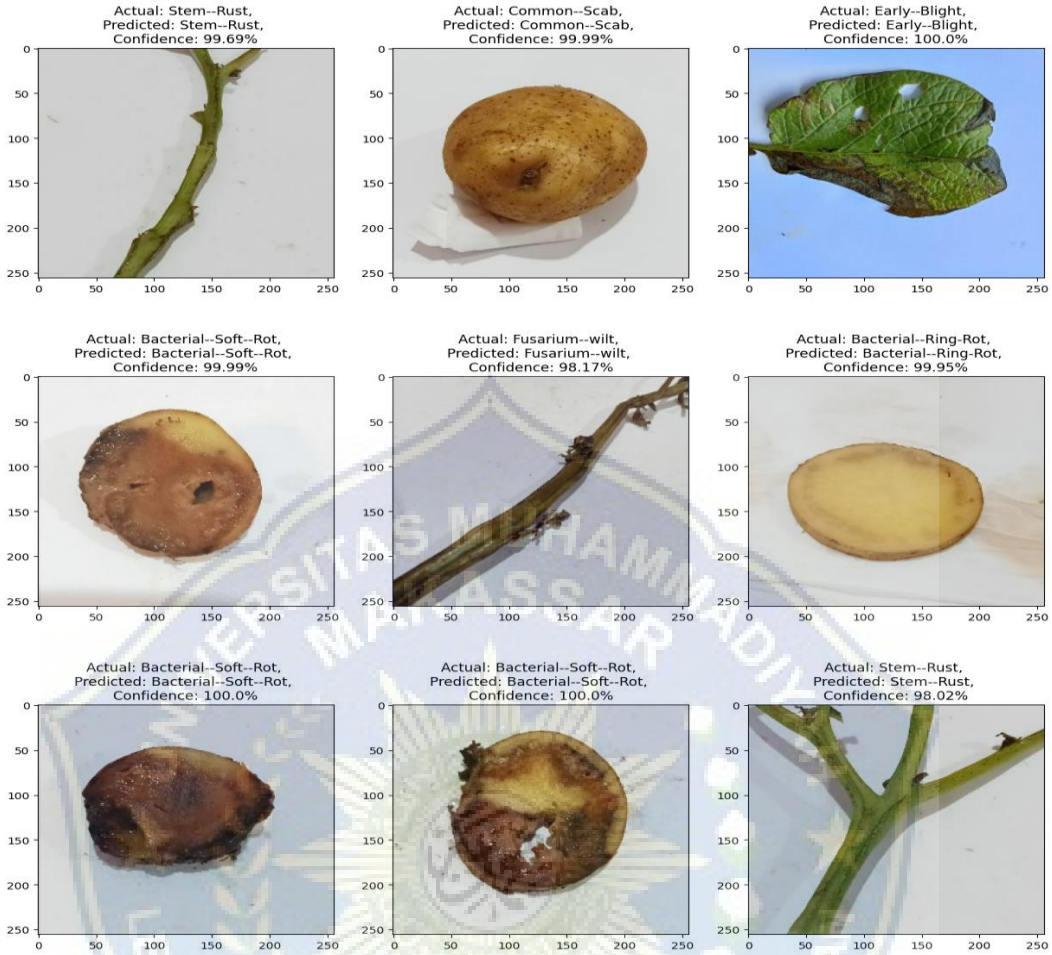


Lampiran 4. Hasil tampilan visual dari *batch* gambar.



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (32, 256, 256, 3)         0

 sequential_1 (Sequential)   (32, 256, 256, 3)         0

 conv2d (Conv2D)             (32, 254, 254, 32)        896

 max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0
 )

 conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

 max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0
 2D)

 conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928

 max_pooling2d_2 (MaxPooling  (32, 30, 30, 64)         0
 2D)

 conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928

 max_pooling2d_3 (MaxPooling  (32, 14, 14, 64)         0
 2D)

 conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928

 max_pooling2d_4 (MaxPooling  (32, 6, 6, 64)           0
 2D)

 flatten (Flatten)           (32, 2304)                0

 dense (Dense)               (32, 64)                  147520

 dense_1 (Dense)             (32, 10)                  650

=================================================================
Total params: 278,346
Trainable params: 278,346
Non-trainable params: 0
_________________________________________________________________
```

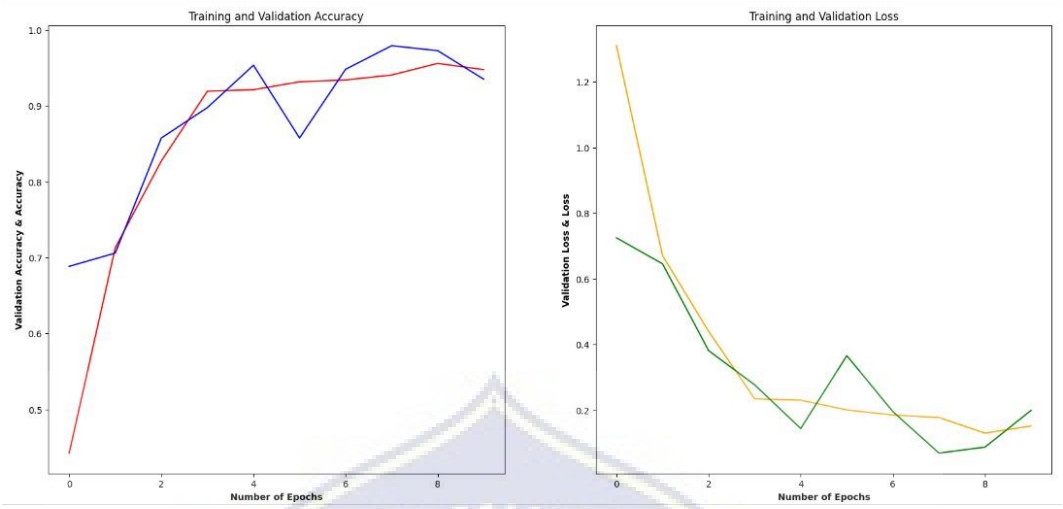
Lampiran 5. Hasil ringkasan arsitektur model CNN



Lampiran 6. Hasil prediksi dan *confidence* kelas

```
Epoch 1/10
258/258 [==============================] - 1561s 20ms/step - loss: 1.3102 - accuracy: 0.4425 - val_loss: 0.7248 - val_accuracy: 0.6886
Epoch 2/10
258/258 [==============================] - 6s 23ms/step - loss: 0.6708 - accuracy: 0.7131 - val_loss: 0.6456 - val_accuracy: 0.7061
Epoch 3/10
258/258 [==============================] - 5s 18ms/step - loss: 0.4400 - accuracy: 0.8274 - val_loss: 0.3812 - val_accuracy: 0.8578
Epoch 4/10
258/258 [==============================] - 5s 18ms/step - loss: 0.2343 - accuracy: 0.9194 - val_loss: 0.2768 - val_accuracy: 0.8978
Epoch 5/10
258/258 [==============================] - 5s 20ms/step - loss: 0.2301 - accuracy: 0.9214 - val_loss: 0.1438 - val_accuracy: 0.9536
Epoch 6/10
258/258 [==============================] - 5s 18ms/step - loss: 0.2003 - accuracy: 0.9318 - val_loss: 0.3658 - val_accuracy: 0.8578
Epoch 7/10
258/258 [==============================] - 5s 18ms/step - loss: 0.1849 - accuracy: 0.9342 - val_loss: 0.1952 - val_accuracy: 0.9483
Epoch 8/10
258/258 [==============================] - 5s 20ms/step - loss: 0.1772 - accuracy: 0.9408 - val_loss: 0.0689 - val_accuracy: 0.9794
Epoch 9/10
258/258 [==============================] - 5s 18ms/step - loss: 0.1297 - accuracy: 0.9561 - val_loss: 0.0871 - val_accuracy: 0.9728
Epoch 10/10
258/258 [==============================] - 5s 18ms/step - loss: 0.1515 - accuracy: 0.9478 - val_loss: 0.1991 - val_accuracy: 0.9352
```

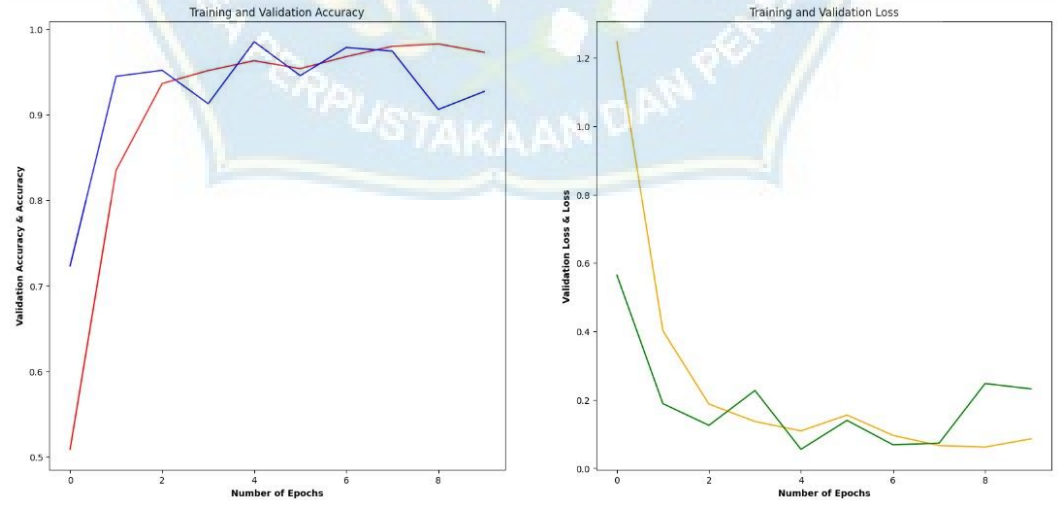
Lampiran 7. Hasil pengujian *batch size* 16 *split* data 80:10:10



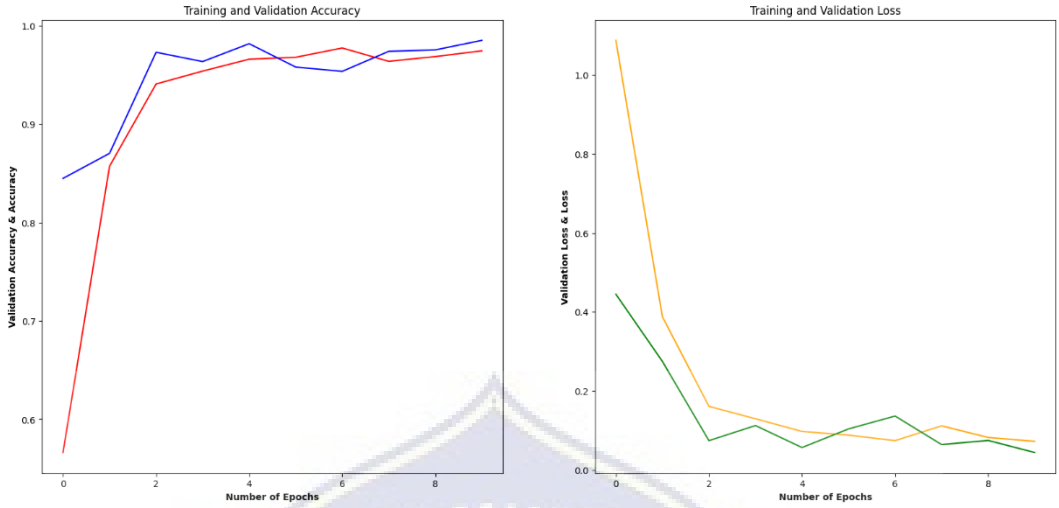
Lampiran 8. Grafik akurasi pengujian *batch size* 16 *split* data 80:10:10

```
Epoch 1/10
56/56 [==============================] - 934s 79ms/step - loss: 1.2473 - accuracy: 0.5087 - val_loss: 0.5648 - val_accuracy: 0.7233
Epoch 2/10
56/56 [==============================] - 3s 52ms/step - loss: 0.4018 - accuracy: 0.8351 - val_loss: 0.1887 - val_accuracy: 0.9449
Epoch 3/10
56/56 [==============================] - 3s 53ms/step - loss: 0.1874 - accuracy: 0.9365 - val_loss: 0.1254 - val_accuracy: 0.9520
Epoch 4/10
56/56 [==============================] - 3s 53ms/step - loss: 0.1367 - accuracy: 0.9517 - val_loss: 0.2272 - val_accuracy: 0.9129
Epoch 5/10
56/56 [==============================] - 4s 63ms/step - loss: 0.1092 - accuracy: 0.9632 - val_loss: 0.0552 - val_accuracy: 0.9854
Epoch 6/10
56/56 [==============================] - 3s 53ms/step - loss: 0.1554 - accuracy: 0.9539 - val_loss: 0.1398 - val_accuracy: 0.9458
Epoch 7/10
56/56 [==============================] - 3s 53ms/step - loss: 0.0960 - accuracy: 0.9680 - val_loss: 0.0684 - val_accuracy: 0.9787
Epoch 8/10
56/56 [==============================] - 3s 63ms/step - loss: 0.0662 - accuracy: 0.9801 - val_loss: 0.0732 - val_accuracy: 0.9744
Epoch 9/10
56/56 [==============================] - 3s 53ms/step - loss: 0.0619 - accuracy: 0.9829 - val_loss: 0.2476 - val_accuracy: 0.9062
Epoch 10/10
56/56 [==============================] - 3s 62ms/step - loss: 0.0859 - accuracy: 0.9730 - val_loss: 0.2319 - val_accuracy: 0.9272
```

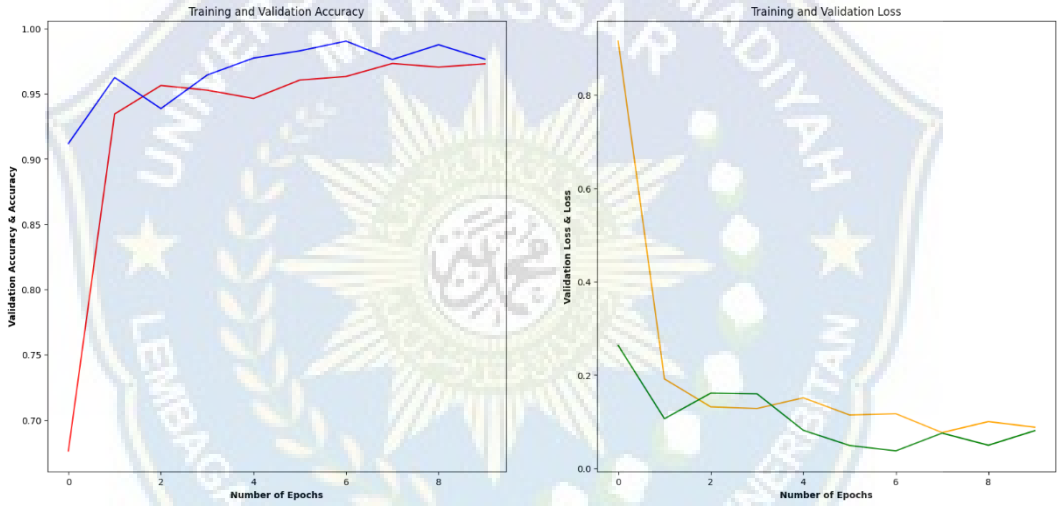
Lampiran 9. Hasil pengujian *batch size* 16 *split* data 70:15:15



Lampiran 10. Grafik akurasi pengujian *batch size* 16 *split* data 70:15:15



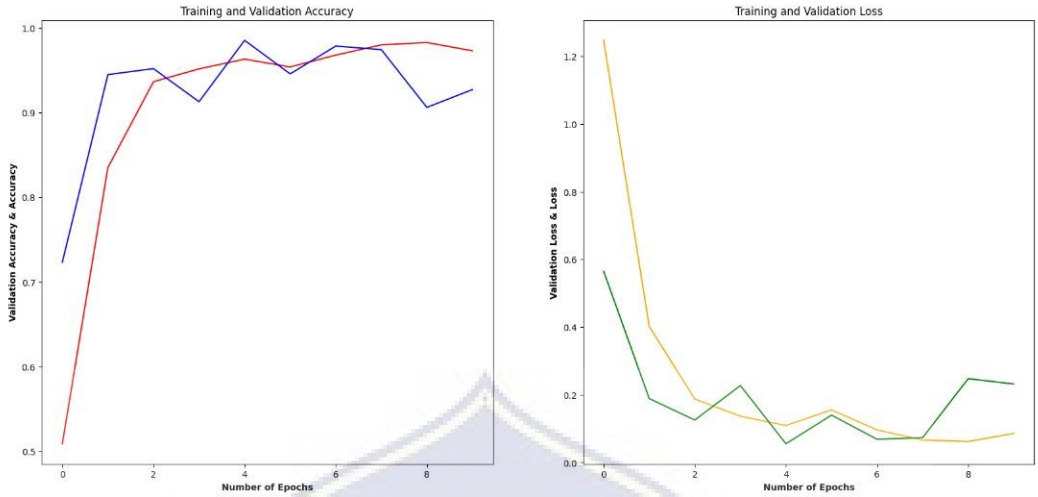
Grafik akurasi pengujian *batch size* 16 *split* data 70:15:15



Grafik akurasi pengujian *batch size* 16 *split* data 70:15:15

```
Epoch 1/10
56/56 [==============================] - 934s 79ms/step - loss: 1.2473 - accuracy: 0.5087 - val_loss: 0.5648 - val_accuracy: 0.7233
Epoch 2/10
56/56 [==============================] - 3s 52ms/step - loss: 0.4018 - accuracy: 0.8351 - val_loss: 0.1887 - val_accuracy: 0.9449
Epoch 3/10
56/56 [==============================] - 3s 53ms/step - loss: 0.1874 - accuracy: 0.9365 - val_loss: 0.1254 - val_accuracy: 0.9520
Epoch 4/10
56/56 [==============================] - 3s 53ms/step - loss: 0.1367 - accuracy: 0.9517 - val_loss: 0.2272 - val_accuracy: 0.9129
Epoch 5/10
56/56 [==============================] - 4s 63ms/step - loss: 0.1092 - accuracy: 0.9632 - val_loss: 0.0552 - val_accuracy: 0.9854
Epoch 6/10
56/56 [==============================] - 3s 53ms/step - loss: 0.1554 - accuracy: 0.9539 - val_loss: 0.1398 - val_accuracy: 0.9458
Epoch 7/10
56/56 [==============================] - 3s 53ms/step - loss: 0.0960 - accuracy: 0.9680 - val_loss: 0.0684 - val_accuracy: 0.9787
Epoch 8/10
56/56 [==============================] - 3s 63ms/step - loss: 0.0662 - accuracy: 0.9801 - val_loss: 0.0732 - val_accuracy: 0.9744
Epoch 9/10
56/56 [==============================] - 3s 53ms/step - loss: 0.0619 - accuracy: 0.9829 - val_loss: 0.2476 - val_accuracy: 0.9062
Epoch 10/10
56/56 [==============================] - 3s 62ms/step - loss: 0.0859 - accuracy: 0.9730 - val_loss: 0.2319 - val_accuracy: 0.9272
```

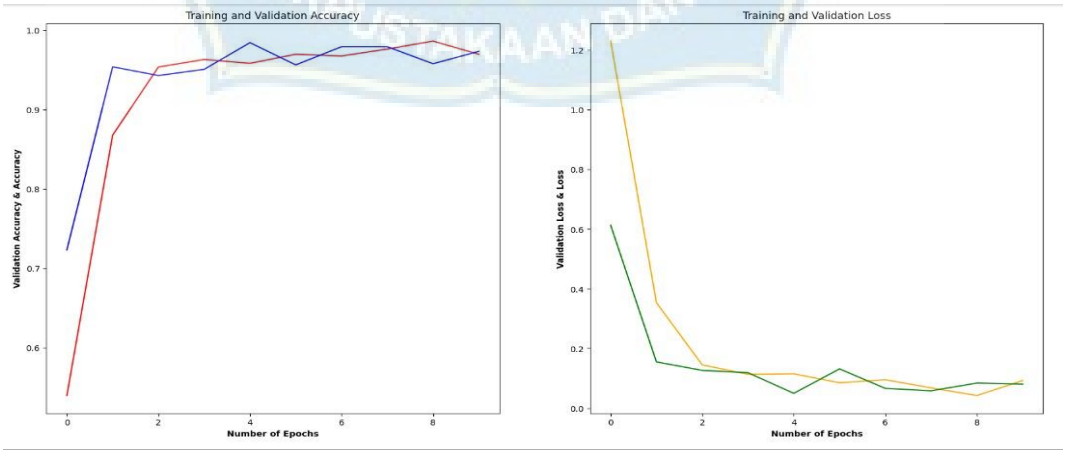
Hasil pengujian *batch size* 64 *split* data 70:15:15



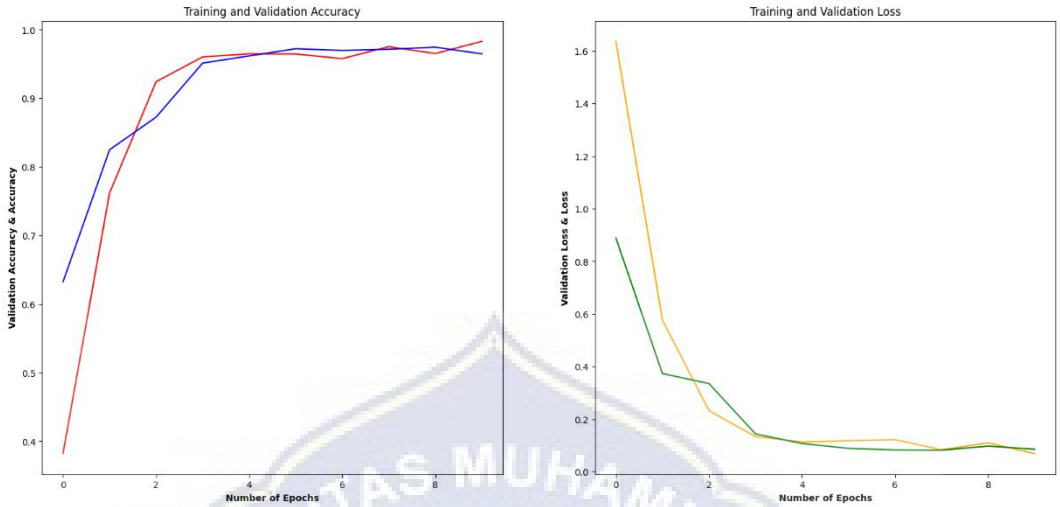
Grafik akurasi pengujian *batch size* 64 *split* data 70:15:15

```
Epoch 1/10
65/65 [==============================] - 417s 58ms/step - loss: 1.2303 - accuracy: 0.5392 - val_loss: 0.6131 - val_accuracy: 0.7230
Epoch 2/10
65/65 [==============================] - 3s 51ms/step - loss: 0.3534 - accuracy: 0.8677 - val_loss: 0.1548 - val_accuracy: 0.9539
Epoch 3/10
65/65 [==============================] - 3s 52ms/step - loss: 0.1449 - accuracy: 0.9536 - val_loss: 0.1265 - val_accuracy: 0.9429
Epoch 4/10
65/65 [==============================] - 4s 59ms/step - loss: 0.1132 - accuracy: 0.9631 - val_loss: 0.1186 - val_accuracy: 0.9507
Epoch 5/10
65/65 [==============================] - 3s 51ms/step - loss: 0.1149 - accuracy: 0.9583 - val_loss: 0.0495 - val_accuracy: 0.9844
Epoch 6/10
65/65 [==============================] - 3s 51ms/step - loss: 0.0850 - accuracy: 0.9697 - val_loss: 0.1314 - val_accuracy: 0.9563
Epoch 7/10
65/65 [==============================] - 4s 60ms/step - loss: 0.0951 - accuracy: 0.9675 - val_loss: 0.0661 - val_accuracy: 0.9793
Epoch 8/10
65/65 [==============================] - 3s 51ms/step - loss: 0.0678 - accuracy: 0.9763 - val_loss: 0.0580 - val_accuracy: 0.9793
Epoch 9/10
65/65 [==============================] - 3s 50ms/step - loss: 0.0421 - accuracy: 0.9863 - val_loss: 0.0839 - val_accuracy: 0.9578
Epoch 10/10
65/65 [==============================] - 3s 50ms/step - loss: 0.0923 - accuracy: 0.9697 - val_loss: 0.0804 - val_accuracy: 0.9734
```

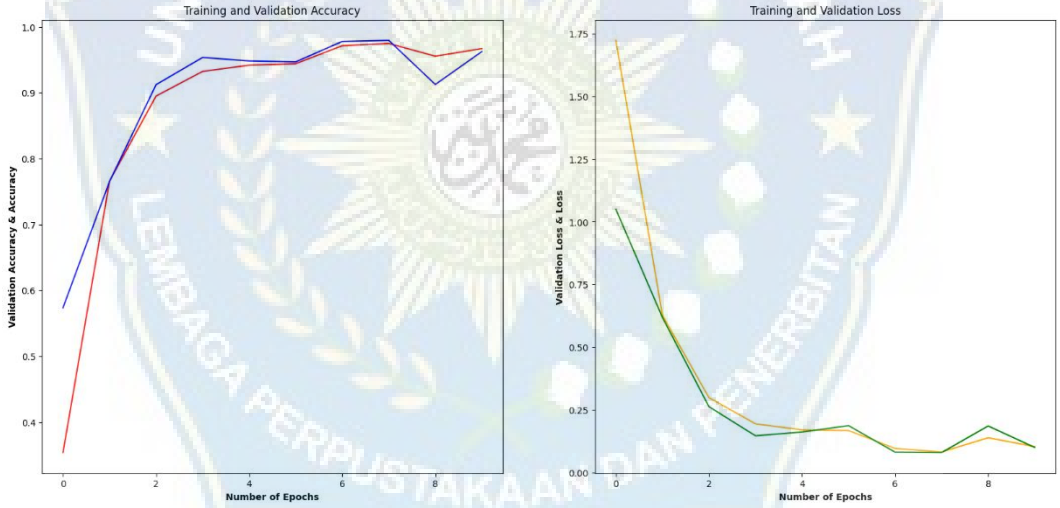
Hasil pengujian *batch size* 64 *split* data 80:10:10



Grafik akurasi pengujian *batch size* 64 *split* data 80:10:10



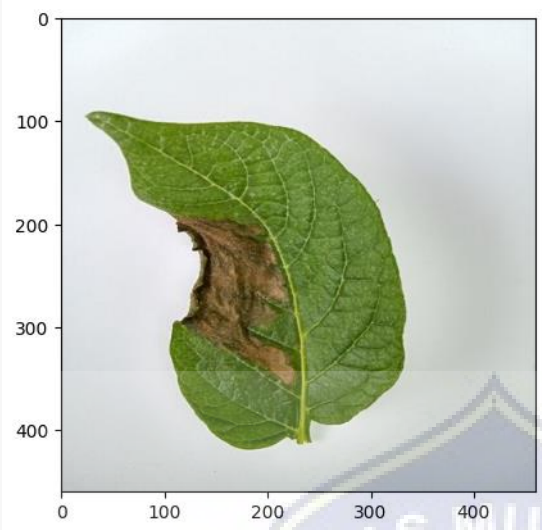
Grafik akurasi pengujian *batch size* 128 *split* data 80:10:10



Grafik akurasi pengujian *batch size* 128 *split* data 70:15:15

Percobaan Gambar 1

```
import cv2
from tensorflow.keras.preprocessing.image import img_to_array
gambar1 = cv2.imread('/content/drive/MyDrive/Potato--
Diseases/Late--Blight/IMG20230803115618_result.jpg')
image_rgb = cv2.cvtColor(gambar1, cv2.COLOR_BGR2RGB)
plt.imshow(image_rgb)

<matplotlib.image.AxesImage at 0x7aa0e2cae320>
```

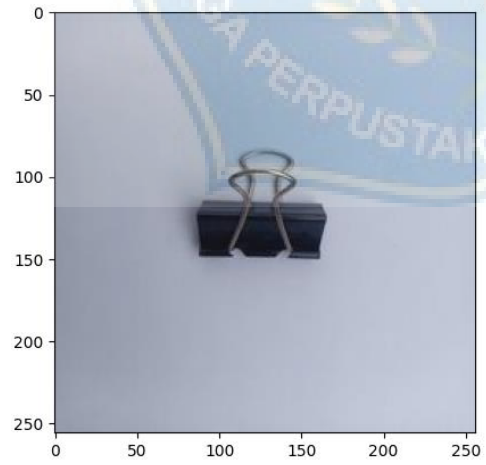



```
Hasil :
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
Predicted class: Late_Blight
```

Percobaan Gambar 2

```
gambar5 = cv2.imread('/content/drive/MyDrive/221.jpg')
image_rgb = cv2.cvtColor(gambar1, cv2.COLOR_BGR2RGB)
plt.imshow(image_rgb)
```

```
<matplotlib.image.AxesImage at 0x7a9f9df90d30>
```




```
Hasil :
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 20ms/step
Predicted class: Bukan--Penyakit--Kentang
```

Lampiran 11. *Source code*

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

import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
import cv2
from sklearn.metrics import confusion_matrix
import seaborn as sns

IMAGE_SIZE = 150
BATCH_SIZE = 32
CHANNELS = 3

dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/Potato--Diseases",
    shuffle = True,
    image_size = (IMAGE_SIZE, IMAGE_SIZE),
    batch_size = BATCH_SIZE
)

classes = dataset.class_names
classes

for image_batch, label_batch in dataset.take(1):
  print(image_batch.shape)

plt.figure(figsize=(10, 10))

for image_batch, label_batch in dataset.take(1):
  for i in range(6):
    ax = plt.subplot(2, 3, i + 1)
    plt.imshow(image_batch[i].numpy().astype("uint8"))
    plt.title(classes[label_batch[i]])

def get_dataset_partition_tf(dataset, train_split = 0.8,
validation_split = 0.1, test_split = 0.1, shuffle = True,
shuffle_size = 10000):
  dataset_size = len(dataset)

  if shuffle:
```

```
    dataset = dataset.shuffle(shuffle_size, seed = 18)

  train_size = int(train_split * dataset_size)
  validation_size = int(validation_split * dataset_size)

  train_dataset = dataset.take(train_size)
  validation_dataset =
dataset.skip(train_size).take(validation_size)
  test_dataset =
dataset.skip(train_size).skip(validation_size)

  return train_dataset, validation_dataset, test_dataset

train_dataset, validation_dataset, test_dataset =
get_dataset_partition_tf(dataset)

len(train_dataset)

len(validation_dataset)

len(test_dataset)

train_dataset =
train_dataset.cache().shuffle(1000).prefetch(buffer_size =
tf.data.AUTOTUNE)
validation_dataset =
train_dataset.cache().shuffle(1000).prefetch(buffer_size =
tf.data.AUTOTUNE)
test_dataset =
train_dataset.cache().shuffle(1000).prefetch(buffer_size =
tf.data.AUTOTUNE)

resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE,
IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1./255)
])

data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_a
nd_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2)
])

input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
```

```
n_classes = 10

model = models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32, (3, 3), activation = 'relu', input_shape
= input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3, 3), activation =
'relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3, 3), activation =
'relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation = 'relu'),
    layers.Dense(n_classes, activation = 'softmax')
])
model.build(input_shape = input_shape)

model.summary()

model.compile(
    optimizer = 'adam',
    loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits =
False),
    metrics = ['accuracy']
)

history = model.fit(
    train_dataset,
    epochs = 10,
    batch_size = BATCH_SIZE,
    verbose = 1,
    validation_data = validation_dataset,
)

scores = model.evaluate(test_dataset)

accuracy = history.history['accuracy']
validation_accuracy = history.history['val_accuracy']

loss = history.history['loss']
validation_loss = history.history['val_loss']
```

```
plt.figure(figsize = (20, 9), linewidth = 1.5)
plt.subplot(1, 2, 1)
plt.plot(range(10), accuracy, label = 'Training Accuracy', c =
'red')
plt.plot(range(10), validation_accuracy, label = 'Validation
Accuracy', c = 'blue')
plt.title('Training and Validation Accuracy')
plt.xlabel('Number of Epochs', fontweight='bold')
plt.ylabel('Validation Accuracy & Accuracy',
fontweight='bold')

plt.subplot(1, 2, 2)
plt.plot(range(10), loss, label = 'Training Loss', c =
'orange')
plt.plot(range(10), validation_loss, label = 'Validation
Loss', c = 'green')
plt.title('Training and Validation Loss')
plt.xlabel('Number of Epochs', fontweight='bold')
plt.ylabel('Validation Loss & Loss', fontweight='bold')
plt.show()

def predict(model,img):
    image = tf.keras.utils.img_to_array(img)
    image = tf.expand_dims(image,0)

    pred = model.predict(image)
    pred_class = classes[np.argmax(pred[0])]
    confidence = round(100 * (np.max(pred[0])),2)
    return pred_class,confidence


import numpy as np
y_test = []
y_pred = []

for image_batch,label_batch in test_dataset.take(1):
    y_test = label_batch[:]
    pred = model.predict(image_batch)
    y_pred = [np.argmax(i) for i in pred]

cm = confusion_matrix(y_test,y_pred)
cm

sns.heatmap(cm,annot=True)
```

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
print(f'confusion_matrix:\n {confusion_matrix(y_test,
y_pred)}')
print (classification_report(y_test, y_pred, zero_division=0))


import numpy as np

for images_batch, labels_batch in test_dataset.take(1):
  first_image = images_batch[0].numpy().astype('uint8')
  first_label = labels_batch[0]

  plt.imshow(first_image)
  print("actual label: ", classes [first_label])

  batch_prediction = model.predict(images_batch)
  print("predicted label: ", classes
[np.argmax(batch_prediction[0])])
def predict(model, img):
  img_array =
tf.keras.preprocessing.image.img_to_array(images[i].numpy())
  img_array = tf.expand_dims(img_array, 0)  #creating a batch

  predictions = model.predict(img_array)

  predicted_class = classes[np.argmax(predictions[0])]
  confidence = round(100 * (np.max(predictions[0])), 2)
  return predicted_class, confidence

plt.figure(figsize = (10, 15))
for images, labels in test_dataset.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))

    predicted_class, confidence = predict(model,
images[i].numpy())
    actual_class = classes[labels[i]]

    plt.title(f"Actual: {actual_class}, \nPredicted:
{predicted_class}, \nConfidence: {confidence}%")
```

Lampiran 12. Surat Keterangan Bebas Plagiasi

MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
UPT PERPUSTAKAAN DAN PENERBITAN
Alamat kantor: Jl Sultan Alauddin NO 259 Makassar 90221 Tlp (0411) 866972,881593, Fax.(0411) 865588

بسم الله الرحمن الرحيم

SURAT KETERANGAN BEBAS PLAGIAT

UPT Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar,
Menerangkan bahwa mahasiswa yang tersebut namanya di bawah ini:

Nama : Resky Utami
Nim : 105841107319
Program Studi : Informatika

Dengan nilai:

No	Bab	Nilai	Ambang Batas
1	Bab 1	10 %	10 %
2	Bab 2	24 %	25 %
3	Bab 3	9 %	10 %
4	Bab 4	5 %	10 %
5	Bab 5	2 %	5 %

Dinyatakan telah lulus cek plagiat yang diadakan oleh UPT- Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar Menggunakan Aplikasi Turnitin.

Demikian surat keterangan ini diberikan kepada yang bersangkutan untuk dipergunakan seperlunya.

Makassar, 28 Agustus 2023
Mengetahui

Kepala UPT- Perpustakaan dan Penerbitan,

[illegible] M I P
NBM 964 [illegible]

Jl. Sultan Alauddin no 259 makassar 90222
Telepon (0411)866972,881 593,fax (0411)865 588
Website: www.library.unismuh.ac.id
E-mail : perpustakaan@unismuh.ac.id

Lampiran 13. Hasil Scan Plagiasi Per Bab

BAB I Resky Utami - 105841107319

ORIGINALITY REPORT

10%	10%	3%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	benydollo.blogspot.com Internet Source	2%
2	lib.unnes.ac.id Internet Source	2%
3	library.universitaspertamina.ac.id Internet Source	2%
4	vdocuments.site Internet Source	2%
5	repository.umsu.ac.id Internet Source	2%

Exclude quotes On

Exclude bibliography On

Exclude matches < 2%

BAB II Resky Utami 105841107319

by Tahap Tutup

Submission date: 26-Aug-2023 12:59PM (UTC+0700)
Submission ID: 2151548538
File name: TUTUP_BAB_II.docx (1.22M)
Word count: 3339
Character count: 21355

BAB II Resky Utami 105841107319

ORIGINALITY REPORT

24%	24%	8%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

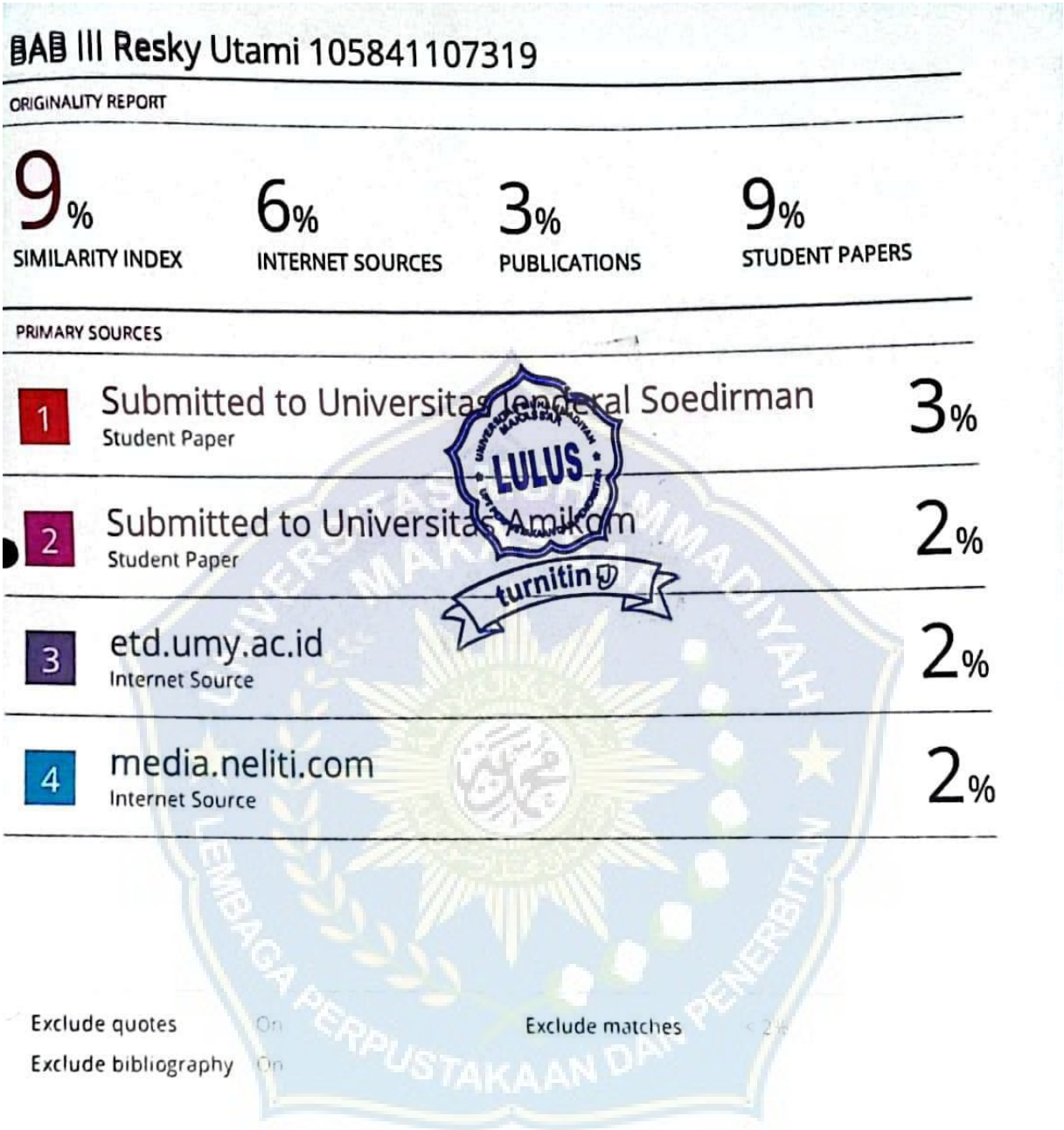
PRIMARY SOURCES

1	semnas.biologi.fmipa.unp.ac.id Internet Source	5%
2	repository.yudharta.ac.id Internet Source	5%
3	journal.uim.ac.id Internet Source	3%
4	pdfs.semanticscholar.org Internet Source	3%
5	Submitted to Universitas Amikom Student Paper	2%
6	jtek.ft-uim.ac.id Internet Source	2%
7	ojs.trigunadharma.ac.id Internet Source	2%
8	dspace.uii.ac.id Internet Source	2%

BAB III Resky Utami 105841107319

by Tahap Tutup

Submission date: 26-Aug-2023 04:41PM (UTC+0700)
Submission ID: 2151592466
File name: TUTUP_BAB_III_1.docx (160.25K)
Word count: 1150
Character count: 7483



BAB III Resky Utami 105841107319

ORIGINALITY REPORT

9%	6%	3%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universitas Jenderal Soedirman Student Paper	3%
2	Submitted to Universitas Amikom Student Paper	2%
3	etd.umy.ac.id Internet Source	2%
4	media.neliti.com Internet Source	2%

LULUS

turnitin

Exclude quotes On

Exclude bibliography On

Exclude matches < 2%

BAB IV Resky Utami 105841107319

by Tahap Tutup

BAB IV Resky Utami 105841107319

ORIGINALITY REPORT

5%	5%	4%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	github.com Internet Source	5%

Exclude quotes	On	Exclude matches	< 2%
Exclude bibliography	On		

BAB V Resky Utami 105841107319

by Tahap Tutup

BAB V Resky Utami 105841107319

ORIGINALITY REPORT

2%	2%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.ummat.ac.id Internet Source	2%

Exclude quotes On

Exclude bibliography On

Exclude matches < 2%