KLASIFIKASI JENIS PENYAKIT DAUN KENTANG MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (CNN) MODEL ARSITEKTUR *GOOGLENET*

SKRIPSI

Diajukan untuk memenuhi salah satu syarat memperoleh gelar sarjana komputer

Oleh :
DAHLIATUL FITRIYAH NINGSIH
201769040016

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS YUDHARTA PASURUAN
2021

KLASIFIKASI JENIS PENYAKIT DAUN KENTANG MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (CNN) MODEL ARSITEKTUR *GOOGLENET*

SKRIPSI

Diajukan untuk memenuhi salah satu syarat memperoleh gelar sarjana komputer

Oleh :
DAHLIATUL FITRIYAH NINGSIH
201769040016

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS YUDHARTA PASURUAN
2021

PERNYATAAN PENULIS

JUDUL : KLASIFIKASI JENIS PENYAKIT DAUN KENTANG MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (CNN) MODEL ARSITEKTUR *GOOGLENET*

NAMA : DAHLIATUL FITRIYAH NINGSIH

NIM : 201769040016

"Saya menyatakan dan bertanggung jawab dengan sebenarnya bahwa Skripsi ini adalah hasil karya saya sendiri kecuali cuplikan dan ringkasan yang masing-masing telah saya jelaskan sumbernya. Jika pada waktu selanjutnya ada pihak lain yang mengklaim bahwa Skripsi ini sebagai karyanya, yang disertai bukti-bukti yang cukup, maka saya bersedia untuk dibatalkan gelar Sarjana Komputer saya beserta segala hak dan kewajiban yang melekat pada gelar tersebut".

Pasuruan, 4 Agustus 2021

Dahliatul Fitriyah Ningsih
Penulis

PERSETUJUAN SKRIPSI

JUDUL : KLASIFIKASI JENIS PENYAKIT DAUN KENTANG MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (CNN) MODEL ARSITEKTUR *GOOGLENET*

NAMA : DAHLIATUL FITRIYAH NINGSIH

NIM : 201769040016

Skripsi ini telah diperiksa dan disetujui

Pasuruan, 4 Agustus 2021

Kaprodi,

Mulmron Rosadi, M.Kom
NIP.Y 0690213121

Pembimbing,

M.Imron Rosadi, M.Kom
NIP.Y 0690213121

PENGESAHAN SKRIPSI

JUDUL : KLASIFIKASI JENIS PENYAKIT DAUN KENTANG MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (CNN) MODEL ARSITEKTUR *GOOGLENET*
NAMA : DAHLIATUL FITRIYAH NINGSIH
NIM : 201769040016

Skripsi ini telah diujikan dan dipertahankan di depan Dewan Penguji pada Sidang Skripsi tanggal 4 Agustus 2021. Menurut pandangan kami, Skripsi ini memadai dari segi kualitas untuk tujuan penganugerahan gelar Sarjana Komputer (S.Kom).

Pasuruan, 4 Agustus 2021

Pembimbing,

M. Imron Rosadi, M.Kom
NIP.Y 0690213121

Penguji Utama,

M. Faishol Amrulloh, M.Kom
NIP.Y 0691709007

Penguji Anggota,

Walidini Syaihul Huda, M.Kom
NIP.Y 0691709006

Kaprodi,

M. Imron Rosadi, M.Kom
NIP.Y 0690213121

Dekan Fakultas Teknik,

Mishadi Munir ,ST.,MT
NIP.Y 0690301015

HALAMAN PERSEMBAHAN

Skripsi ini saya persembahkan kepada :

- Kedua orang tua saya yang tiada lelah dalam memberikan do'a, motivasi serta kasih sayangnya yang ikhlas dan tiada batas. Semoga Allah memberikan kebahagiaan untuk orang tuaku di dunia dan akhirat. Aamiin.
- Bapak dan Ibu dosen serta seluruh karyawan Fakultas Teknik, jurusan Teknik Informatika Universitas Yudharta Pasuruan.
- Teman-teman seperjuanganku semua, Mahasiswa/i jurusan Teknik Informatika angkatan 2017 Universitas Yudharta Pasuruan.
- Semua pihak yang telah banyak membantu peneliti dalam proses penyelesaian skripsi ini.
- Seseorang yang terkasih dan tercinta. Terimakasih atas segala dukungan dan do'anya.

Peneliti mengucapkan terimakasih yang sedalam-dalamnya kepada semua pihak yang telah memberikan motivasi dan do'anya. Semoga Allah SWT membalas semua kebaikan-kebaikan itu. Aamiin.

MOTTO

"APA YANG MELEWATKANKU TIDAK AKAN PERNAH MENJADI TAKDIRKU, DAN APA YANG DITAKDIRKAN UNTUKKU TIDAK AKAN PERNAH MELEWATKANKU"

ABSTRACT

Potato is the largest vegetable commodity in Indonesia. Potato plants are not free from problems such as pests and diseases that can lead to crop failure. The process of identifying diseases that are carried out manually with the human sense of sight has drawbacks, namely subjective assessments that are influenced by lack of concentration and fatigue and need quite a lot of experience. Potato plant diseases are a late blight and there are other diseases in potato plants. which is often encountered in dry blight (early blight). Deep learning has good capabilities in Computer Vision, one of which is image classification or object classification in the image. Convolutional Neural Network is one of the Deep Learning algorithms designed to process data in two-dimensional form, for example, images or sounds which are the development of Multilayer Perceptron (MLP). The application of the Artificial Neural Network uses the Convolutional Neural Network method with GoogleNet architecture to identify potato leaf diseases. Tests were carried out with 2152 potato leaf image samples, 1700 images as training data ,and 452 images as test data. Tests show that this CNN method is expected to have an average percentage of accuracy above 70%.

Keywords : Potato leaf disease, Deep Learning, Convolutional Neural Network, GoogleNet

KLASIFIKASI JENIS PENYAKIT DAUN KENTANG MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (CNN) MODEL ARSITEKTUR *GOOGLENET*

Dahliatul Fitriyah Ningsih
Program Studi Teknik Informatika, Universitas Yudharta Pasuruan

ABSTRAK

Kentang merupakan komoditi sayuran terbesar di Indonesia. Tanaman kentang tidak luput dari masalah seperti adanya serangan hama dan penyakit yang bisa mengakibatkan kegagalan panen. Proses identifikasi penyakit yang dilakukan manual dengan indera penglihatan manusia memiliki kekurangan yaitu penilaian yang bersifat subyektif yang dipengaruhi oleh kurangnya konsentrasi dan rasa lelah serta perlu pengalaman yang cukup banyak.Penyakit tanaman kentang berupa penyakit busuk daun (*late blight*) dan terdapat penyakit lain pada tanaman kentang yang sering dijumpai adalah bercak kering (*early blight).Deep learning* memiliki kemampuan yang baik dalam *Computer Vision*, salah satunya yaitu *image classification* atau klasifikasi objek pada citra. *Convolutional Neural Network* merupakan salah satu algoritma *Deep Learning* dirancang untuk mengolah data dalam bentuk dua dimensi, misalnya gambar atau suara yang merupakan pengembangan dari *Multilayer Peceptron* (MLP). Penerapan Jaringan Syaraf Tiruan menggunakan metode *Convolutional Neural Network* dengan arsitektur *GoogleNet* untuk melakukan proses identifikasi penyakit daun kentang. Pengujian dilakukan dengan 2152 sampel citra daun kentang, 1700 citra sebagai data latih dan 452 citra sebagai data uji. Pengujian menunjukkan metode CNN inidiharapkan memiliki hasil persentase rata- rata *accuracy* diatas 70%.

Kata kunci : Penyakit daun kentang, *Deep Learning, Convolutional Neural Network, GoogleNet*

KATA PENGANTAR

Alhamdulillah puji syukur kepada Tuhan Yang Maha Esa karena atas berkat rahmat-Nya, penulis dapat menyelesaikan skripsi dengan baik dan benar. Sholawat serta salam tidak lupa dihaturkan kepada junjungan kita Nabi Muhammad SAW sebagai suri tauladan umat manusia.

Penyelesaia dan penulisan skripsi ini tidak terlepas dari bantuan dan dukungan dari berbagai pihak, baik secara langsung maupun tidak langsung. Oleh karena itu, penulis mengucapkan banyak terima kasih dan penghargaan kepada pihak-pihak yang telah membantu :

1. Kedua Orang Tua yang selalu mendoakan dan memberikan motivasi serta dukungan di setiap perjuangan penulis.
2. KH. Sholeh Bahrudin**,** selaku Pembina Yayasan Darut Taqwa yang selalu memberikan doa restunya.
3. Bapak Dr. H. Kholid Murtadlo, SE., ME., selaku Rektor Universitas Yudharta Pasuruan.
4. Bapak Misbach Munir, ST, MT selaku Dekan Universitas Yudharta Pasuruan.
5. Bapak Muhammad Imron Rosadi, S.Kom., M.Kom. selaku Ketua prodi Teknik Informatika
6. Bapak M. Imron Rosadi, S.Kom., M.Kom, selaku dosen pembimbing yang telah membimbing penulis dalam menyelesaikan proposal skripsi ini.
7. Serta sahabat- sahabat saya Miya, Anggy, Putra, Firdaus dan Rif'an yang senantiasa saling membantu dan saling memberi dorongan motivsi selama proses penulisan proposal skripsi ini.

Dengan segala kerendahan hati, penulis menyadari bahwa masih jauh dari kata sempurna, sehingga kritik dan juga saran sangat penulis harapkan

untuk mencapai hasil yang lebih baik. Harapan penulis semoga proposal skripsi ini bisa bermanfaat bagi penulis sendiri khususnya dan bagi pembaca pada umumnya. Amin.

Pasuruan, 4 Agustus 2021

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kentang merupakan komoditi sayuran terbesar di Indonesia yang dimana Indonesia termasuk negara agraris tentunya harus memiliki hasil panen yang baik dan besar terutama untuk tanaman kentang. Provinsi Jawa Timur, Jawa Tengah, Jawa Barat, Sumatera Utara dan Sulawesi Utara merupakan wilayah penghasil produksi kentang yang tebesar secara nasional pada tahun 2018 mencapai 1,28 juta ton, dikarenakan tanaman kentang merupakan komoditi yang sangat banyak dicari oleh konsumen. Petani sayuran merupakan ujung tombak dalam memberikan hasil yang baik bagi hasil pertanian sayuran. Apabila mendapatkan hasil panen yang baik maka merupakan suatu kebanggaan yang luar biasa, karena selain mendapatkan harga jual yang baik, tentunya juga bisa menyelamatkan negara dalam hal tidak mengimpor kentang dari luar Indonesia. Sehingga masyarakat Indonesia mengkonsumsi tanaman dari hasil negeri sendiri dan membeli dengan harga yang murah (Badan Pusat Statistik, 2018).

Mendapatkan hasil panen kentang yang baik merupakan hal paling utama dalam masa tanam, oleh sebab itu tanaman kentang tidak boleh teserang hama, agar mendapatkan hasil panen yang baik. Adapun penyakit utama yang menyerang tanaman kentang yaitu penyakit busuk daun atau biasa disebut (*Late Blight*) dan terdapat penyakit lain pada tanaman kentang yang sering dijumpai adalah bercak kering (*Early Blight*).

Proses identifikasi jenis penyakit pada daun kentang bisa dilakukan dengan panca indera penglihatan manusia dengan mengamati pola kerusakan pada daun kentang. Akan tetapi manusia memiliki kekurangan penilaian yang subjektif sehingga hasil identifikasi antar individu bisa berbeda karena kurangnya konsentrasi dan rasa lelah atau kecapekan dan juga memerlukan pengalaman yang cukup banyak. Karena komputer tidak dapat membedakan tekstur seperti halnya pengelihatan manusia, maka digunakan analisis tekstur untuk mengetahui pola suatu citra digital berdasarkan ciri yang diperoleh secara matematis. Kedepannya, pengolahan citra ini diharapkan akan menjadi salah satu pilihan dalam identifikasi penyakit daun entang.(Sari et al., 2020).

Beberapa peneliti yang sudah mengambil topik mengenai klasifikasi penyakit daun kentang dengan metode pengolahan citra yang ada. Salah satunya penelitian oleh (Rakhmawati et al., 2018) yang menggunakan metode segmentasi K-means sebagai Clustering, metode Gray Level Co-occurrence Matrix sebagai ekstraksi berdasarkan fitur tekstur dan fitur warna menggunakan Color Moment. Hasil dari kombinasi kedua fitur tersebut menghasilkan 7 fitur tekstur dan 6 fitur warna, kemudian digunakan sebagai input klasifikasi *Multi Support Vektor Machine* menggunakan *kernel Radial Basis Function*. Sehingga hasil dari penelitian ini mampu mendeteksi dan mengklasifikasi penyakit daun pada tanaman kentang dengan akurasi mencapai 80%.

Penelitian serupa dilakukan oleh (Huda & Nafi'yah, 2020) yang menggunakan fitur warna, serta fitur tekstur menggunakan metode Gray Level Co-Occurance Matrix (kontras, korelasi, energi, dan

homogeneity). Kemudian identifikasi penyakit menggunakan algoritma SVM dan KNN. Dari hasil uji coba klasifikasi atau identifikasi penyakit daun Kentang menggunakan algoritma SVM ini menunjukkan bahwa fitur warna paling tinggi akurasinya 88%, sedangkan fitur tekstur dan bentuk secara berturut-turut akurasinya 79,67%, dan 53%. Dan dari hasil menggunakan algoritma KNN menunjukkan K=5 dengan fitur warna nilai akurasinya paling tinggi 85%. Sehingga hasil keseluruhan, fitur warna paling baik dalam melakukan identifikasi penyakit daun Kentang.

Penelitian serupa dilakukan oleh (Lee, 2020) mengusulkan metode CNN (*Convolutional Neural Network*) yang cocok untuk deteksi penyakit kentang dengan model arsitektur VGG19 dengan dataset lebih dari 200 gambar. Dalam model yang diusulkan ini untuk mendeteksi status kesehatan daun kentang diwujudkan dengan struktur konvolusional jaringan saraf dalam penilaian otomatis penyakit tanaman. Akurasi penyakit penilaian juga dapat mencapai akurasi 99%.

Metode *Deep Learning* merupakan cabang ilmu dari *Machine Learning* berbasis jaringan saraf tiruan yang dapat mengajarkan komputer untuk melakukan suatu tindakan yang dianggap alami oleh manusia. Dalam metode *Deep Learning*, sebuah komputer dapat belajar mengklasifikasi secara langsung dari data gambar, teks atau suara.Lapisan dalam *Deep Learning* terdiri dari tiga bagian yaitu *input layer, hidden layer dan output layer*.(Kusumaningrum, 2018)

Terdapat salah satu metode Deep Learning yang memiliki hasil paling signifikan dalam pengenalan citra yaitu Convolutional Neural Network (CNN).

Karena CNN telah meniru sistem pengenalan citra pada visual cortex manusia sehingga mampu mengolah informasi citra. Akan tetapi CNN memiliki kelemahan seperti metode Deep Learning lainnya yaitu proses pelatihan model yang lama (Wijaya & Soelaiman, 2016). Pada penelitian ini akan memanfaatkan kelebihan dari metode CNN yaitu mampu mengklasifikasikan dan memiliki hasil paling signifikan dalam pengenalan objek yang diperuntukan untuk data gambar sebagai salah satu solusi dalam identifikasi jenis penyakit daun kentang sehingga diharapkan masyarakat Indonesia lebih paham tentang jenis penyakit pada daun kentang.Terdapat beberapa model arsitektur pada metode CNN yaitu AlexNet, VGG16, VGG19, ResNet50, *GoogLeNet*, Inception-ResNetV2, dan Squeezenet. Pada penelitian ini berfokus pada model arsitektur *Googlenet*. Dengan kontribusi dari penelitian ini saya membuat perbedaan dari penilitian sebelumnya yaitu pada penelitian sebelumnya mendeteksi penyakit daun kentang berdasarkan kelas penyakit dan sehat sedangkan pada penelitian ini mendeteksi penyakit daun kentang berdasarkan kelas penyakitnya *Early blight dan Late Blight*. Perbedaan juga terletak pada model arsitektur yang di gunakan yaitu pada penelitian sebelumnya menggunakan VGG16 sedangkan pada penelitian ini menggunakan *GoogleNet*.

1.2 Rumusan Masalah

Adapun rumusan masalah dari penellitian ini adalah sebagai berikut:

1. Bagaimana penerapan model arsitektur *GoogleNet* pada *Convolutional Neural Network* (CNN) untuk mendeteksi penyakit daun kentang ?
2. Bagaimana tingkat akurasi penerapan model arsitektur *GoogleNet* pada *Convolutional Neural Network*(CNN) untuk mendeteksi penyakit daun kentang?

1.3 Tujuan Penelitian

Sesuai dengan perumusan masalah diatas maka penelitian ini mempunyai tujuan sebagai berikut:

1. Mampu mengimplementasi model *GoogleNet* pada *Convolutional Neural Network*(CNN)untuk mendeteksi penyakit daun kentang.
2. Mengetahui pengaruh model *GoogleNet* pada *Convolutional Neural Network*(CNN) untuk menentukan tingkat keakurasian dalam mendeteksi jenis penyakit daun kentang.

1.4 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Manfaat Teoristis

   Penelitian ini diharapkan menjadi referensi bagi penulis-penulis berikutnya dalam klasifikasi penyakit daun kentang menggunakan salah satu arsitektur dari algoritma *Convolutional Neural Network* (CNN) yaitu *GoogleNet*

2. Manfaat Praktis
    a. Bagi Penulis
       Menambah wawasan baru bagi penulis dalam menerapakan algoritma *GoogleNet* untuk klasifikasi penyakit daun kentang.
    b. Bagi Petani
       Membantu petani dalam mendiagnosis penyakit daun kentang dengan lebih mudah dan hasil akurasi yang maksimal.
    c. Bagi Masyarakat Umum
       Mengetahui penyakit yang dialami oleh tanaman kentang khususnya pada daun sehingga dapat melakukan langkah preventif terhadap penyebaran penyakit pada tanaman kentang yang masih sehat.

1.5 Batasan Masalah

Adapun batasan masalah pada penelitian ini dirinci sebagai berikut :

1. Jenis citra yang diidentifikasi adalah citra daun *Early Blight* dan *Late Blight*
2. Penelitian ini fokus dari klasifikasi penyakit dilihat dari gejala yang terdapat pada daun tanaman.
3. Jumlah datayang digunakan dalam testing 452 gambar dan training 1700 gambar.
4. Ukuran pixel yang digunakan dalam citra 100x100.

1.6 Sistematika Penulisan

Pada penelitian ini sistematika penulisan terbagi menjadi beberapa bab dengan pokok pembahasan sebagai berikut :

BAB I PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang permaslahan, rumusan masalah, tujuan penelitian, manfaat penelitian , batasan masalah, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini dijelaskam mengenai referensi jurnal penelitian terkait dan landasan teori.

BAB III METODE PENELITIAN

Bab ini menjelaskan mengenai kerangka pemikiran, metodologi penelitian, tahap pengolahan data serta diagram alir penelitian.

BAB IV HASIL dan PEMBAHASAN

Bab ini berisi uraian serta penjelasan, dan penjabaran setiap tahap yang dilakukan untuk menjawab rumusan masalah yang telah disebutkan.

BAB V PENUTUP

Bab ini berisi kesimpulan penelitian secara umum dan saran berdasarkan hasil penelitian yang telah dilakukan sebelumnya.

DAFTAR PUSTAKA

Pada daftar pustaka terdapat daftar referensi yang digunakan penulis sebagai acuan dalam melakukan penelitian seperti jurnal, artikel, buku , maupun skripsi.

HALAMAN INI SENGAJA DIKOSONGKAN

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

2.1.1 (Islam et al., 2017) Detection of Potato Diseases Using Image Segmentation and Multiclass Support Vector Machine.

Pada penelitian tersebut membahas tentang deteksi penyakit daun kentang dengan menggunakan metode segmentasi dengan menentukan ambang batas untuk saluran L *, a * dan b * yang akan menyegmentasikan daun dari latar belakang, metode GLCM untuk ektraksi tekstur dan klasifikasi menggunakan metode multiclass SVM dengan Kernel 'linier'. Hasil dari metode tersebut dengan database 300 gambar daun kentang terdiri darigambar 100 daun sehat dan 200 daun sakit, mendapatkan akurasi sebesar 95%.

2.1.2 (Ilahiyah & Nilogiri, 2018) Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan *Convolutional Neural Network*

Pada penelitian tersebut membahas tentang identifikasi jenis tumbuhan menggunakan metode Convolutional Neural Network (CNN) berdasarkan arsitektur model AlexNet. Dengan jumlah dataset 2000 citra. Masing-masing folder berjumlah 1800 citra pada citra latih, dan 200 citra pada citra uji, dengan hasil tiga kali percobaan pada setiap folder dan dihitung akurasinya. Dengan hasil percobaan pertama mendapatkan nilai akurasi sebesar 90 % dengan prediksi benar sebanyak

36 citra dari 40 citra uji. Hasil percobaan kedua hanya didapati dua prediksi salah dari 40 citra uji dengan hasil nilai akurasi 95%. Dan pada percobaan ketiga menghasilkan nilai akurasi sebesar 87%. Sehingga dari ketiga percobaan tersebut, didapatkan nilai akurasi sistem sebesar 90,8%.

2.1.3 (Rakhmawati et al., 2018)Klasifikasi Penyakit Daun Kentang Berdasarkan Fitur Tekstur dan Fitur Warna Menggunakan Support Vector Machine.

Pada penelitian tersebut membahas tentang klasifikasi penyakit daun kentang dengan menggunakan metode segmentasi *K-Means Clustering*, metode ekstraksi *Grey Level Co-occurrence Matrix*, dan fitur warna *Color Moment*, kemudian metode klasifikasi dengan *Multi Support Vektor Machine kernel Radial Basis Function*. Dengan Database image berjumlah masing–masing 100 citra early, 100 citra blight, dan 100 citra non-disease/normal. Dengan hasil 7 fitur tekstur dan 6 fitur warna dari kombinasi kedua fitur tersebut yang kemudian digunakan sebagai input klasifikasi Multi *Support Vektor Machine kernel Radial Basis Function*. Pada penelitian ini mampu mendeteksi dan mengklasifikasi penyakit daun pada tanaman kentang dengan hasil akurasi mencapai 80%.

2.1.4 (Kusumaningrum, 2018)Implementasi *Convolutional Neural Network* (CNN) Untuk Klasifikasi Jamur Konsumsi di Indonesia Menggunakan Keras.

Pada penelitian tersebut membahas tentang implementasi metode *Convolutional*

Neural Network (CNN) untuk klasifikasi jamur konsumsi di Indonesia dilakukan menggunakan *package Keras pada Rstudio versi 1.1.383* dengan jumlah data 100 gambar untuk masing-masing jenis jamur. Hasil menunjukkan tingkat akurasi yang diperoleh dari model *Convolutional Neural Network (CNN)* yaitu sebesar 100% pada proses training dan 81,667% pada proses testing.

2.1.5 (Arifin et al., 2019)Klasifikasi Penyakit Pada Orchidaceae Menggunakan Pengolahan Citra Dengan Metode *Convolutional Neural Network* (CNN).

Pada penelitian tersebut membahas tentang klasifikasi penyakit pada dun anggrek dengan menggunakan metode *Convolutional Neural Network* (CNN) berdasarkan arsitektur LeNet5. Hasil pengujian sistem dengan sampel tersebut tidak termasuk dataset dengan pengambilan 10 gambar pada setiap kategori, telah didapatkan total 40 gambar memiliki tingkat keberhasilan akurasi sebesar 85%, berdasarkan hasil akurasi tersebut dapat dikatakan bahwa sistem berjalan sesuai dengan fungsinya dengan cukup baik.

2.1.6 (Tiwari, 2020) Potato Leaf Diseases Detection Using Deep Learning.

Pada penelitian tersebut membahas tentang deteksi penyakit daun kentang menggunakan metode *Convolutional Neural Network* (CNN) berdasarkan beberapa arsitektur sperti VGG16, dan VGG19. Dalam model yang diusulkan ini 2.152 gambar daun kentang yang terdiri dari 1000 gambar penyakit awal, 1000 gambar penyakit busuk

daun, dan 152 gambar kentang sehat. Hasil menunjukkan bahwa pada model VGG19 memberikan hasil yang optimal dengan akurasi klasifikasi 97,7%.

2.1.7 (Ansari, 2020) Performance Comparison Of Machine Learning Classifier For The Detection Of Potato Leaf Diseases.

Pada penelitian tersebut membahas tentang deteksi penyakit daun kentang menggunakan metode klasifikasi Naive Bayes (NB), Support Vector Machine (SVM), dan *Convolutional Neural Network* (CNN). Dengan dataset 552 gambar dan tiga label kelas. Detail kelas set data yang digunakanadalah Kentang busuk daun, Kentang dan Kentang busuk daunsehat adalah 200, 200 dan 152. Hasil menunjukkan bahwa pengklasifikasi CNN mencapai akurasi tertinggi. pengklasifikasi ini telah dievaluasi melalui berbagai kinerja mengukur pada set data yang tersedia untuk umum yang menunjukkan CNN tersebut memperoleh akurasi tertinggi sekitar 91%.

2.1.8 (Lee, 2020) Health Detection for Potato Leaf with Convolutional Neural Network

Pada penelitian tersebut membahas tentang deteksi penyakit daun kentang menggunakan metode *Convolutional Neural Network* (CNN) dengan model VGG16. Dataset 2000 gambar dan hasil deteksi klasifikasi mencapai akurasi sebesar 99%.

2.1.9 (Huda & Nafi'yah, 2020) Identifikasi Penyakit Daun Kentang Berdasarkan Fitur Warna , Tekstur , dan Bentuk dengan SVM dan KNN.

Pada penelitian tersebut membahas tentang identifikasi penyakit daun kentang dengan menggunakan metode ekstraksi GLCM, fitur warna dengan nilai rata-rata RGB, standar deviasi RGB, skewness RGB, dan entropy RGB. Fitur tekstur, yaitu: nilai ratarata Grayscale, dan klasifikasi mengunakan metode SVM dan KNN. Dataset 1200 citra daun Kentang, dengan klasifikasi 3 jenis penyakit penyakit awal, penyakit busuk daun, dan sehat. Hasil klasifikasi atau identifikasi penyakit daun Kentang dengan algoritma SVM menunjukkan bahwa fitur warna paling tinggi akurasinya 88%, sedangkan fitur tekstur dan bentuk akurasinya 79,67%, dan 53%. Dan hasil dengan menggunakan algoritma KNN menunjukkan K=5 dengan fitur warna nilai akurasinya paling tinggi 85%. Dari hasil tersebut secara keseluruhan dapat dikatakan bahwa fitur warna paling baik dalam melakukan identifikasi penyakit daun Kentang.

2.1.10 (Nisa et al., 2020) Penerapan Metode *Convolutional Neural Network* untuk Klasifikasi Penyakit Daun Apel pada Imbalanced Data

Penelitian tersebut membahas tentang klasifikasi penyakit daun apel menggunakan metode *Convolutional Neural Network* (CNN) dengan model arsitektur InceptionV3. Dengan dataset Plant Pathology 2020 sebanyak 1.821 data citra dengan 4 kelas. Data citra akan dibagi menjadi 3 set data (latih, validasi, dan uji). Hasil Akurasi yang sesuai dari kelas apel healthy, multiple disease, rust, dan scab

masing-masing 90,6%, 62,3%, 94,3%, 92%. Ditemukan bahwa nilai pada kelas kedua tidak cukup bagus, dilihat dari perbedaan nilai yang cukup signifikan daripada ketiga nilai lainnya.

Tabel 2.1 Penelitian Terkait

Penulis	Judul	Metode	Dataset	Hasil
Monzurul Islam, Khan Wahid	Detection of Potato Diseases Using Image Segmentation and Multiclass Support Vector Machine	*multiclass support vector machine (SVM), GLCM*	gambar 100 daun sehat dan 200 daun sakit. set pelatihan berisi 180 gambar, dan set pengujian berisi 120 gambar.	Penyakit yang paling signifikan pada kentang, penyakit busuk daun dan penyakit awal hawar, diidentifikasi dengan sedikit usaha komputasi dengan hasil akurasi 89%
Ilahiyah & Nilogiri	Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network	*Convolutional Neural Network (CNN) dengan arsitektur AlexNet.*	2000 citra. Citra latih pada masing-masing folder berjumlah 1800 citra, sedangkan citra uji berjumlah	Percobaan pertama mendapatkan prediksi benar sebanyak 36 citra dari 40 citra uji nilai akurasi sebesar 90%., percobaan kedua 40 citra uji menghasilkan

			200 citra. Setiap folder dilakukan tiga kali percobaan dan dihitung akurasinya.	nilai akurasi 95 %. Percobaan ketiga menghasilkan nilai akurasi sebesar 87%. Dari ketiga percobaan tersebut, didapatkan nilai akurasi sistem sebesar 90,8 %.
Puji Utami Rakhmawati, Yuliana Melita Pranoto, Endang Setyati	Klasifikasi Penyakit Daun Kentang Berdasarkan Fitur Tekstur dan Fitur Warna Menggunakan Support Vector Machine.	*K-Means Clustering, Grey Level Co-occurrence Matrix, Color Moment, Multi Support Vektor Machine kernel Radial Basis Function*	*Database image* ini berjumlah masing–masing 100 citra *early*, 100 citra *blight*, dan 100 citra *non-disease*/normal	Kombinasi kedua fitur tersebut menghasilkan 7 fitur tekstur dan 6 fitur warna. klasifikasi Multi Support Vektor Machine kernel Radial Basis Function dengan akurasi mencapai 80%.
Kusumaningrum	Implementasi *Convolutional Neural Network* (CNN) Untuk	*Implementasi metode Convolutional*	100 gambar untuk masing-masing	Tingkat akurasi yang diperoleh dari model *Convolutional*

	Klasifikasi Jamur Konsumsi di Indonesia Menggunakan Keras	*Neural Network (CNN)*	jenis jamur	*Neural Network* (CNN) yaitu sebesar 100% pada proses training dan 81,667% pada proses testing.
Arifin	Klasifikasi Penyakit Pada Orchidaceae Menggunakan Pengolahan Citra Dengan Metode *Convolutional Neural Network* (CNN)	*Convolutional Neural Network (Cnn) berdasarkan arsitektur LeNet*		Hasil dari pengujian pengambilan 10 gambar setiap kategori, didapatkan total 40 gambar memiliki tingkat keberhasil sebesar 85%,
Tiwari	Potato Leaf Diseases Detection Using Deep Learning.	*Convolutional Neural Network* (CNN) Model VGG16,dan VGG19	1000 gambar penyakit busuk daun, dan 152 gambar kentang sehat.	Hasil menunjukkan bahwa pada model VGG19 memberikan hasil yang optimal dengan akurasi klasifikasi 97,7%
Jamshoro	Performance Comparison Of Machine	*Naive Bayes (NB),*	552 gambar dan	Pengklasifikasi ini telah dievaluasi

	Learning Classifier For The Detection Of Potato Leaf Diseases.	*Support Vector Machine (SVM), dan Jaringan Neural Konvol1` usional (CNN)*	tiga label kelas. Det ail kelas set data yang digunakan adalah Kentang busuk daun, Kentang dan Kentang busuk daun sehat adalah 200, 200 dan 152	melalui berbagai kinerja, mengukur pada set data yang tersedia untuk umum yang menunjukkan CNN tersebut memperoleh akurasi tertinggi sekitar 91%.
Trong-Yen Lee, Jui-Yuan Yu, Yu-Chun Chang	Health Detection for Potato Leaf with *Convolutional Neural Network*	CNN (*Convolut ional Neural Network*) Model warna RGB	database gambar lebih dari 2.000	Metode yang diusulkan untuk mendeteksi status kesehatan daun kentang diwujudkan dengan struktur konvolusional jaringan saraf dalam penilaian otomatis penyakit

				tanaman. Akurasi penyakit penilaian juga dapat mencapai akurasi 99%.
M. Rofiq Khoirul Huda, Nur Nafi'iyah	Identifikasi Penyakit Daun Kentang Berdasarkan Fitur Warna, Tekstur, dan Bentuk dengan SVM dan KNN	*GLCM, SVM, KNN* fitur warna: nilai rata-rata RGB, standar deviasi RGB, skewness RGB, dan entropy RGB. Fitur tekstur, yaitu: nilai ratarata grayscale.	1200 citra daun Kentang, dengan klasifikasi 3 jenis penyakit: penyakit awal, penyakit busuk daun, dan sehat	Hasil klasifikasi dengan algoritma SVM menunjukkan bahwa fitur warna paling tinggi akurasinya 88%, sedangkan fitur tekstur dan bentuk akurasinya 79,67%, dan 53%. Dan hasil dengan menggunakan algoritma KNN menunjukkan K=5 dengan fitur warna nilai akurasinya paling tinggi 85%.
Penerapa	*Convolutional*	*dataset*	dataset	Akurasi yang

n Metode *Convolutional Neural Network* untuk Klasifikasi Penyakit Daun Apel pada *Imbalanced Data*	*Neural Network (*CNN) dengan arsitektur *Alexnet*	*Plant Pathology* 2020 - FGV C7 sebanyak 1.821 data citra dengan 4 kelas. Data dibagi menjadi 3 set data (latih, validasi, dan uji)	Plant Pathology 2020 sebanyak 1.821 data citra dengan 4 kelas.	sesuai dari kelas apel healthy, multiple disease, rust, dan scab masing-masing 90,6%, 62,3%, 94,3%, 92%. Hal ini menunjukkan bahwa penanganan model pada data yang tidak seimbang belum cukup bagus pada salah satu kelas.

2.2 Landasan Teori

2.2.1 Kentang

Kentang atau dalam nama latin Solanum tuberosum L merupakan tanaman dari suku Solanaceae yang termasuk umbi batang dan dapat dimakan yang disebut dengan "kentang" juga. Tanaman ini disebut dengan herba (tanaman pendek tidak berkayu) semusim dan menyukai iklim yang sejuk dan cocok ditanam pada dataran tinggi. Beberapa penyakit tanaman kentang yang dapat dipengaruhi oleh cuaca, pengolahan tanah, pengobatan tanaman, dan perawatan tanaman hingga pasca panen. Dengan cara pengolahan tanah yang baik, pemilihan bibit yang unggul, pengobatan tanaman sesuai dengan takaran atau ukuran dan langkah yang tepat, serta perawatan tanaman kentang sampai pasca panen merupakan usaha terbaik untuk menanggulangi penyakit tanaman kentang.(Safitri, 2017)

2.2.2 Penyakit Daun Kentang *Early Blight*

Penyakit tersebut menyerang pada daun kentang dengan gejala berbercak kecil tersebar tidak teratur, warna coklat tua, meluas ke daun muda. Pada permukaan kulit umbi berbercak gelap tidak beraturan, kering, berkerut dan keras, adanya bercak abu-abu hingga coklat yang muncul dan perlahan-lahan tumbuh secara konsentris pada bagian yang bersih. Apabila penyakit ini berkembang, maka seluruh daun dapat mengalami klorosis dan gugur. Gejala tersebut disebabkan oleh *Alternaria Solani*, yaitu jamur yang melewati musim dingin pada puing-

puing tanman yang terinfeksi di tanah. Biasanya daun bagian bawah sering terinfeksi ketika kontak dengan tanah yang terkontaminasi. Jamur tersebut sering menyerang setelah periode curah hujan tinggi dan merusak di daerah tropis dan subtropis(Semangun, 1989). Contoh gambar *Early Blight* sebagai berikut :

Gambar 2.1 Penyakit daun kentang *Early Blight*
Sumber : www.kaggle.com

2.2.3 Penyakit Daun Kentang *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.Gejala penyakit ini dapat mudah menyebar pada batang, tangkai, umbi dan buah, dan serangan penyakit ini dapat berkembang dengan cepat pada saat musim hujan dengan kelembaban di sekitar kanopi. Penyakit busuk daun ini

termasuk penyakit yang paling merugikan pada tanaman kentang di seluruh dunia. Kehilangan hasil panen yang diakibatkan oleh penyakit ini dapat mencapai 60-80% bahkan dapat mengakibatkan kehilangan hasil sampai 100%(Semangun, 1989). Contoh gambar *Late Blight* sebagai berikut :

Gambar 2.2 Penyakit daun kentang *Late Blight*
Sumber : www.kaggle.com

2.2.4 Daun kentang *Healthy*

Daun kentang yang sehat tidak terinfeksi hama dan gejala penyakit akan menghasilkan panen baik, jika hasil panen umbi yang baik (sehat tidak terkena hama dan penyakit, permukaannya rata, kulit tidak lecet) mencapai 90-95%. Namun, kadang tidak mendukung hanya 70%, bahkan yang ditanam di dataran rendah dan ditanam saat curah hujan tinggi mengalami penyusutan hasil panen sampai 50%, karena kelembaban tanahnya tinggi (Tabloid Sinar Tani,

n.d.) Contoh gambar daun *Healthy* sebagai berikut :

Gambar 2.3 Daun kentangSehat
Sumber : www.kaggle.com

2.2.5 Citra Digital

Citra digital yaitu representasi visual dari suatu objek atau benda. Proses menangkap atau capturesuatu objek yang merepresentasikan citra digital disebut dengan akuisisi citra. Secara matematis citra didefinisikan sebagai fungsi dua dimensi *f(x, y)* dimana *x* dan *y* adalah koordinat spasial (bidang) dan amplitudo *f* pada titik *x, y* merupakan nilai intensitas atau nilai derajat keabuan dari suatu citra. Citra dikatakan sebuah citra digital apabila nilai *x, y* dan nilai intensitas *f* terbatas dalam besaran diskrit. Citra digital terdiri dari sejumlah elemen yang terbatas dan pada masing-masing titik *x, y* memiliki nilai tertentu. Istilah paling umum untuk menyatakan elemen-elemen dalam citra yaitu piksel Citra digital *M x*

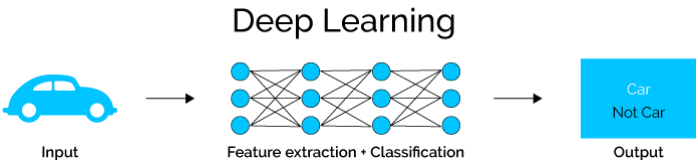
N piksel dapat direpresentasikan dengan matriks baris *M* dan kolom *N*. Matriks yang merepresentasikan citra digital ditunjukkan pada Persamaan berikut :

$$f(x,y) = \begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & f(2,3) & \cdots & f(2,N) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f(M,1) & f(M,2) & f(M,3) & \cdots & f(M,N) \end{bmatrix}$$

Citra menjadi data utama yang digunakan dalam implementasi pengelolahan citra digital. Pengelolahan citra digital merupakan teknologi yang digunakan untuk penggalian informasi dalam suatu citra dan pemprosesan citra melalui berbagai algoritma matematika. Pengelolahan citra digital telah banyak digunakan dalam berbagai bidang seperti teknik, ilmu komputer, biology science, medical science, dan lain-lain.

2.2.6 *Deep Learning*

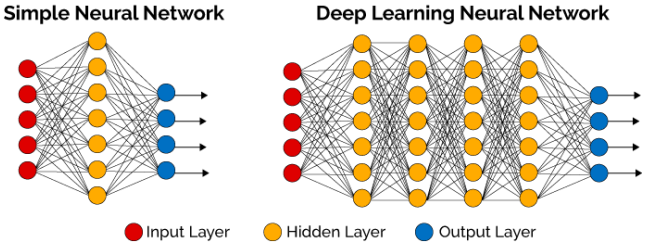
Deep learning yakni algoritma dalam ilmu machine learning yang berusaha belajar dalam berbagai level, yaitu sesuai dengan tingkat abstraksi yang berbeda. Biasanya pada *deep learning* menggunakan jaringan syaraf tiruan. Konsep dalam tingkat model statistik yang dipelajari dalam deep learning ini berbeda dimana konsep tingkat yang lebih tinggi ditentukan dari tingkat yang lebih rendah, kemudian konsep tingkat yang lebih rendah dapat membantu untuk mendeskripsikan banyak konsep tingkat yang lebih tinggi. *Deep learning* menggambarkan bagian dari *machine learning* yang menjadi acuan penelitian paling popular saat ini pada bidang pengolahan citra digital (Apakah Deep Learning_MMSI BINUS University, n.d.).



Gambar 2.4 Contoh struktur pemodelan jaringan pada *Deep Learning*
Sumber : (Dadang, 2018)

Jaringan Syaraf Tiruan memiliki arsitektur jaringan yang kurang kompleks dan membutuhkan lebih banyak informasi tentang

data input sehingga harus terlebih dahulu menentukan algortima mana yang akan digunakan seperti algoritma *Model Hebb, Perceptron, Adaline, Propagasi Maju, dll*. Sedangkan pada pembelajaran algortima Deep Learning tidak memerlukan informasi apapun terhadap data yang akan dipelajarinya, dan algoritma tersebut dapat secara mandiri melakuan tuning (penyetelan) dan pemilihan model yang paling optimal.

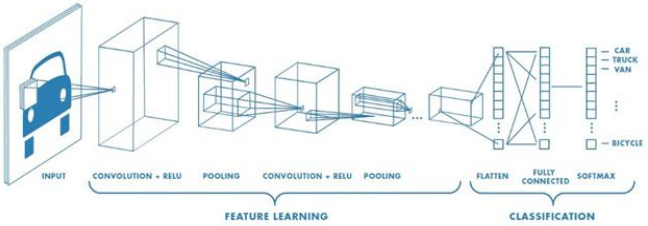


Gambar 2.5 Perbedaan antara lapisan layer pada Jaringan Saraf Tiruan dengan Jaringan *Deep Learning*
Sumber : (Dadang, 2018)

2.2.7 *Convolutional Neural Network* (CNN)

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. *Convolutional Neural Network* terdiri dari *input layer*, *output layer*, *dan beberapa hidden layer*. *Hidden layer* di dalam CNN pada umumnya terdiri dari yaitu

convolution layer, fungsi aktivasi, Rectified Linear Unit (ReLU), local response normalization layer, pooling layer, dropout layer dan fully connected layer. Layer tersebut digunakan untuk mempelajari fitur dari data citra dengan membaca dan mengolah nilai-nilai pikselnya. *Fully connected layers* merupakan layer terakhir yang memberi nilai probabilitas pada masing-masing kelas. (Haq et al., 2021)Ilustrasi layer pada algoritma CNN ditunjukkan pada gambar berikut :



Gambar 2.6 Layer pada algoritma CNN
Sumber : (Haq et al., 2021)

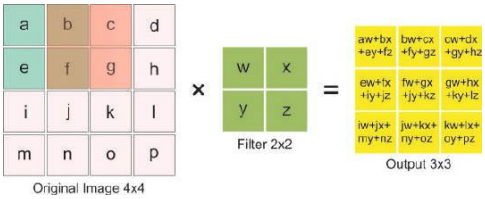
a. Feature Learning

Feature Learning mempunyai lapisan - lapisan yang berguna untuk mentranslasikan suatu input menjadi features berdasarkan ciri dari input tersebut yang berbentuk angka-angka dalam vektor. Ekstraksi fitur mempunyai lapisan yang terdiri dari *Convolutional Layer, Rectified Linear Unit (ReLU)* dan *Pooling Layer.*

1) Convolution layer

Convolution layer merupakan layer paling penting dari CNN. Konvolusi merupakan sebuah matriks yang diterapkan pada citra dengan operasi matematika. Convolution layer ini dapat diterapakan untuk beberapa operasi konvolusi pada data

citra secara berurutan guna mempelajari fitur tepi, warna, dan bentuk dari citra input. Operasi konvolusi diterapkan pada dua matriks, yaitu matriks yang berisi piksel citra input dan matriks kernel atau filter. Filter dalam *Convolutional Neural Network* merupakan matriks berisi nilai random antara -1 sampai 1 yang digunakan untuk mempelajari fitur-fitur dalam citra pada area-area kecil sesuai dengan ukuran filternya. Ukuran filter bergantung pada jenis-jenis arsitekturnya. Filter bergeser sebanyak stride pada nilai-nilai piksel citra input. Operasi konvolusi dengan stride 1 di ilustrasikan pada Gambar berikut :



Gambar 2.7 Operasi Konvolusi

Sumber : (Haq et al., 2021)

Pada *convolutional layer*, neuron tersusun menjadi feature maps. Setiap neuron pada *feature map* sebagai *receptive field*, pada *neuron convolution* layer sebelumnya melalui serangkaian bobot yang dilatih atau biasa disebut dengan filter bank, yang diilustrasikan pada persamaan (1):

$h(x,y) = f(x,y) * g(x,y)$.....(1)

$h(x,y)$ = hasil dari proses convolution

$f(x,y)$ = nilai matrix citra

$g(x, y)$ = kernel convolution

2) Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) merupakan fungsi aktivasi yang paling umum digunakan dalam algoritma CNN. ReLU merubah nilai input *neuron feature map* yang dihasilkan dari *convolution laye*r berada pada range 0 hingga infinity. Misalkan x merupakan nilai input neuron, operasi fungsi aktivasi ReLU didefinisikan pada persamaan berikut :

F(x) = max(0, x).....(2)

f(x) = nilai dari ReLu activation

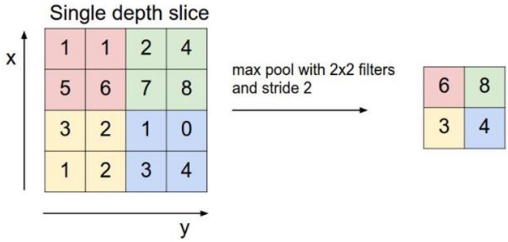
x = nilai matrix dari citra

Nilai 0 ditetapkan sebagai fungsi aktivasi sebagai pengganti nilai negatif pada *feature map* dan nilai input *neuron feature map* tetap jika bernilai lebih dari atau sama dengan 0.

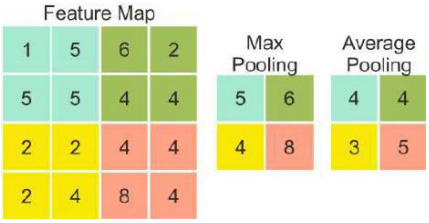
3) *Pooling Layer*

Pooling Layer yaitu lapisan dalam *Convolutional Neural Network* yang menggunakan fungsi dengan *Feature Map* sebagai masukan dan mengolahnya dengan berbagai macam operasi statistik berdasarkan nilai piksel terdekat. Lapisan Pooling tersebut yang dimasukkan diantara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN, Pooling layer berfungsi mempercepat proses perhitungan komputasi yaitu dengan cara mengurangi volume pada setiap tumpukan feature map tanpa menghilangkan

informasi-informasi penting. *Pooling layer* memiliki beberapa jenis seperti *max pooling layer* dan *average pooling layer*. Filter berukuran 2x2 merupakan bentuk lapisan Pooling pada umumnya yang kemudian diaplikasikan dengan langkah sebanyak 2 lalu beroperasi pada setiap irisan dari input. Maka bentuk seperti ini akan mengurangi *Feature Map* hingga 75% dari ukuran aslinya. Contoh operasi *Max Pooling* ditunjukkan dalam Gambar berikut :



Gambar 2.8 Operasi*Max Polling*
Sumber : (Ilahiyah & Nilogiri, 2018)



Gambar 2.9 Jenis-jenis *Pooling Layer*
Sumber : (Haq et al., 2021)

Ukuran dimensi *pooling layer* ditentukan sesuai stride dan ukuran dimensi kernel yang ditunjukkan pada persamaan :

$$O = \frac{i-k}{s} + 1 \dots\dots\dots\dots\dots\dots\dots (3)$$

Yang mana O merupakana ukuran dimensi pooling layer, i merupakan ukuran dimensi matriks dari input citra, k merupakan ukuran dimensi matriks kernel dan s merupakan jumlah stride.

b. Classification

Lapisan ini berguna untuk mengklasifikasikan tiap neuron yang telah diekstraksi fitur pada sebelumnya. Terdiri dari :

1) *Flatten*

Flatten ini guna untuk membentuk ulang fitur (reshape feature map) menjadi sebuah vector agar bisa digunakan sebagai input dari fully-connected layer.

2) *Fully connected layer*

Fully connected layer melakukan klasifikasi pada output yang dihasilkan dari convolution layer dan pooling layer. Setiap jaringan pada fully connected layer saling terhubung dengan jaringan pada lapisan sebelumnya. Fully connected layer dianggap sebagai layer penyatuan akhir yang mengklasifikasi fitur kedalam kelas-kelas berdasarkan set data pelatihan. Penentuan kelas data citra dilihat dari kombinasi fitur yang paling kuat. Fully connected ayers didefinisikan pada Persamaan berikut :

$$y_j = b_j + \sum_i w_{ij} x_i$$

Dimana x_i merupakan input pada *fully connected layer* yang merupakan hasil dari

pembelajaran fitur, *w* merupakan bobot jaringan berukuran *i x j* dengan *i* menunjukkan jumlah fitur dan *j* menunjukkan jumlah target kelas, *b* merupakan bias, dan *y* merupakan output dari *fully connected layer*.

3) Aktifasi *Softmax*

Aktivasi *Softmax* atau disebut Softmax Classifier ini merupakan bentuk lain dari algoritma Logistic Regression yaitu mampu mengklasifikasi lebih dari dua kelas. Standar klasifikasi yang pada umumnya dilakukan oleh algoritma *Logistic Regression* adalah tugas untuk klasifikasi pada kelas biner. Fungsi aktivasi softmax merupakan fungsi aktivasi yang biasanya terletak pada output layer dengan rentang nilai 0 hingga 1. Bentuk persamaan yang muncul pada Softmax adalah sebagai berikut :

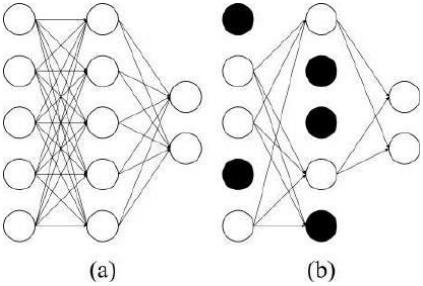
$$f_j \quad f_j(Z) = \frac{e^{zJ}}{\sum_k e^{zk}} \quad \text{.....(4)}$$

Dari rumus tersebut, notasi fj menyatakan hasil fungsi untuk setiap elemen ke-j pada vektor keluaran kelas. Pada Argumen z tersebut, hipotesis yang diberikan oleh model pelatihan agar dapat diklasifikasikan oleh fungsi Softmax. Aktivasi Softmax ini juga dapat memberikan hasil yang lebih intuitif dan memiliki interpretasi probabilistik yang lebih baik dibanding dengan algoritma klasifikasi yang lainnya. Softmax memungkinkan menghitung probabilitas

untuk semua label. Dari label yang ada maka akan diambil sebuah vektor nilai bernilai riil kemudian akan dirubah menjadi vektor dengan nilai antara nol dan satu yang apabila semua dijumlah akan bernilai satu.

2.2.8 *Dropout Layer*

Algoritma CNN memiliki *hidden layer* yang sangat kompleks dan sering kali menyebabkan *overfitting*. Salah satu teknik mengatasi *overfitting* yaitu dengan menggunakan dropout layer. *Dropout layer* merupakan teknik sederhana yang dapat mempengaruhi kinerja algoritma CNN dalam proses testing model. Sebagai contoh dropout layer ditunjukkan pada Gambar berikut :



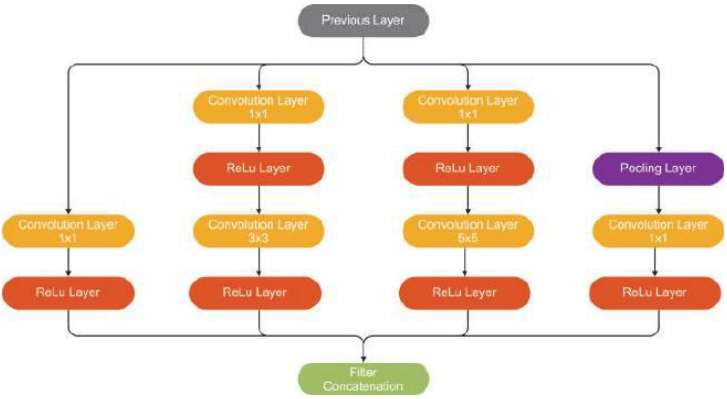
Gambar 2.10 (a) Contoh neural network tanpa dropout dan (b) Contoh neural network menggunakan dropout

Sumber : (Haq et al., 2021)

Dropout layer bekerja menghilangkan beberapa neuron secara acak dengan probabilitas sebesar p pada setiap iterasi proses training model dan p dapat ditentukan dengan uji coba. Setelah diterapkan dropout layer, jaringan akan lebih tipis dari sebelumnya yang mengurangi overfitting pada model dan dapat menpercepat proses training.

2.2.9 *GoogleNet*

GoogleNet merupakan arsitektur yang diperkenalkan oleh Google pada tahun 2014 dan menempati peringkat pertama dalam kompetisi *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) 2014 sebagai arsitektur dengan kinerja terbaik. Nilai error yang didapatkan *GoogleNet* dalam klasifikasi data citra mencapai 6.7%. Model pembelajaran arsitektur *GoogleNet* dapat mencapai akurasi tinggi dengan memperdalam layer untuk meningkatkan kinerja jaringan saraf. *GoogleNet* memiliki 144 layer dengan ukuran citra pada layer input 224x 224x3. Keunggulan *GoogleNet* terletak pada *inception modules* yang tidak dimiliki CNN pada umumnya. Inception modules terdiri dari beberapa convolution kecil yang bertujuan untuk mereduksi jumlah parameter tanpa mengurangi kinerja jaringan dengan susunan seperti pada Gambar berikut :



Gambar 2.11*Inception modules*
Sumber :(Haq et al., 2021)

2.2.10 *Confusion Matrix*

Confusion matrix adalah alat evaluasi visual yang digunakan dalam sistem klasifikasi. *Confusion matrix* ini berguna untuk mengukur seberapa baik model klasifikasi yang telah dibuat. *Confusion matrix* berukuran *n n*, dimana *n* merupakan jumlah kelas yang berbeda Confusion matrix menentukan akurasi yang didapat dari nilai beberapa parameter, seperti True Positif (TP), False Positif (FP), True Negatif (TN), dan False Negatif (FN). (Foeady et al., 2019). Tabel confusion matrix ditunjukkan pada Tabel 2.2 berikut :

Tabel 2.2*Confusion Matrix*

Kelas Sebenarnya	Kelas Hasil Klasifikasi	
	Predicted	Predicted
Actual	True Positive (TP)	True Negative (TN)
Actual	False Positive (FP)	False Negative (FN)

1. TP adalah *True Positive*, yaitu jumlah data positif yang terklasifikasi dengan benar oleh sistem.
2. TN adalah *True Negative*, yaitu jumlah data negatif yang terklasifikasi dengan benar oleh sistem.
3. FN adalah *False Negative*, yaitu jumlah data negatif namun terklasifikasi salah oleh sistem.
4. FP adalah *False Positive*, yaitu jumlah data positif namun terklasifikasi salah oleh sistem.

Berdasarkan nilai-nilai True Positif (TP), False Positif (FP), True Negatif (TN), dan False Negatif (FN), evaluasi klasifikasi dianalisis dari beberapa indikator yaitu diantaranya indikator akurasi, spesifisitas, dan sensitifitas. Akurasi merupakan rasio antara jumlah terprediksi benar dari semua data. Spesifisitas merupakan nilai yang menunjukkan banyak data bernilai negatif yang mampu terklasifikasi dengan benar masuk ke dalam kelas negatif. Sensitivitas merupakan nilai yang menunjukkan banyak data bernilai positif yang mampu terklasifikasi dengan benar masuk ke dalam kelas positif. Indikator tersebut dihitung dengan Persamaan berikut :

$$Akurasi = \left(\frac{TP+TN}{TP+TN+FP+FN}\right) \times 100\% ...(5)$$

$$Spesifisitas = \left(\frac{TP}{TP+FP}\right) \times 100\% .....(6)$$

$$Sensitifitas = \left(\frac{TP}{TP+FN}\right) \times 100\% .....(7)$$

2.2.11 *Python*

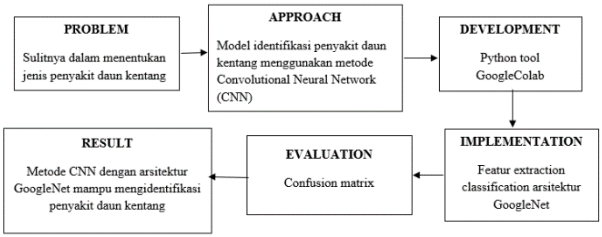
Python adalah bahasa pemrograman interpretatif multiguna yang tidak seperti bahasa lain yaitu susah untuk dibaca dan dipahami, bahasa python lebih menekankan pada keterbacaan kode agar lebih memudahkan untuk memahami sintaks. Maka dari itu bahasa *Python* sangat mudah dipelajari dan dipahami baik bagi pemula maupun bagi yang sudah menguasai bahasa pemrograman lain. Pemrograman *Python* mempunyai kode yang simpel dan mudah diimplementasikan, sehingga seorang programmer dapat lebih fokus mengutamakan pengembangan aplikasi yang telah dibuat, tanpa sibuk mencari syntax error. Python dapat diaplikasikan dalam berbagai bentuk dari web, desktop, hingga analisis data. Google saat ini telah memberikan fasilitas bahasa pemrograman Python melalui Google Colaboratory atau Google Interactive Notebook. Google Colab menyediakan tiga jenis prosesor yang handal dalam keperluan mesin pembelajaran (machine learning). Google Colab ini dapat dijadikan referensi bagi para programmer untuk meningkatkan skill dan juga belajar *python* (Enterprise, 2019).

HALAMAN INI SENGAJA DIKOSONGKAN

BAB III

METODE PENELITIAN

3.1 Kerangka Pemikiran



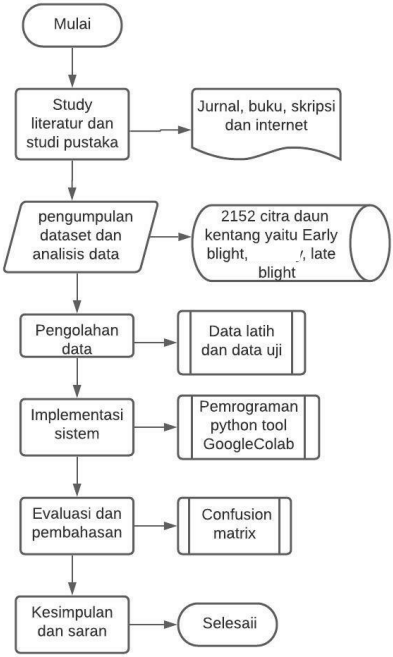
Gambar 3.1 Kerangka Pemikiran

Gambar diatas merupakan kerangka berpikir dari konsep klasifikasi penyakit daun kentang. Penelitian ini diharapkan bermanfaat dan dapat memudahkan dalam mengetahui jenis penyakit daun kentang.

3.2 Metodologi Penelitian

Penelitian deteksi penyakit daun kentang ini menggunakan *Convolutional Neural Network* (CNN) dengan jenis arsitektur *GoogleNet* berdasarkan klasifikasi data citra. Penelitian ini termasuk jenis penelitian kuantitatif karena didasarkan pada data yang digunakan, yaitu dengan menggunakan data numerik yang berisi nilai-nilai piksel dari data citra. Hasil dari penelitian ini adalah sistem klasifikasi citra penyakit daun kentang ini diharapkan dapat membantu para petani dalam mengidentifikasi jenis penyakit dau kentang secara cepat dan tepat.

3.3 Alur Penelitian



Gambar 3.2 Alur Penelitian

Adapun penjelasan dalam gambar alur peneletian di atas adalah sebagai berikut:

1. Studi literatur

   Studi literatur merupakan tahap paling awal yang bertujuan untuk mempersiapkan semua bahan yang dibutuhkan yaitu seperti pengumpulan data atau sumber yang terkait, seperti permasalahan, dasar teori, metode penelitian,kebutuhan software, dan juga penelitian sejenis untuk keperluan pada penelitian tersebut.

2. Pengumpulan dataset

Di dalam penelitian proses pengumpulan data, harus mnggunakan metode yang sesuai dengan sifat individualitas penelitian yang dilakukan (Bachri, 2010). Berdasarkan sumber yang ada data dibagi menjadi dua yakni data primer dan skunder. Pada Penelitian ini akan menggunakan data skunder. Berikut penjelasan dari data primer dan skunder:

a. Data Primer

Data primer merupakan metode pengambilan data secara langsung dari sumber data, seperti angket, observasi, dan wawancara yang dilakukan peneliti secara langsung. Pada metode pengumpulan data ini dibutuhkan biaya dan waktu yang banyak dari data skunder.

b. Data Skunder

Data skunder merupakan metode pengambilan data yang dilakukan oleh penelitian lain. Meskipun data tersebut sudah pernah diteliti oleh pihak lain namun mempunyai tujuan berbeda, seperti pada kantor pemerintah, perpustakaan, toko buku, dan internet Data skunder ini didapatkan relatif lebih cepat dan biaya rendah.

Berikut beberapa alasan menggunakan data skunder:

1) Lebih praktis dikarenakan data sudah tersedia tidak membutuhkan waktu yang banyak dan biaya yang mahal untuk mendapatkan data yang dibutuhkan.

2) Karena data yang dibutuhkan lebih valid
3) Kemungkinan data yang didapatkan lebih banyak dibandingkan data primer

3. Pengolahan data

Pada tahap ini dilakukan memisahkan data latih dan data uji untuk melakukan proses penelitian sehingga mecapai tujuan penelitian dengan metode penelitian yang telah di tentukan. Metode yang diusulkan dalam metode ini yakni metode *Deep Learning Convolutional Neural Network* (CNN) dengan menggunakan arsitektur *Googlenet*.

4. Implementasi sistem

Implementasi merupakan pelaksanaan atau penerapan, jadi yang dilaksanakan dan diterapkan yaitu pokok-pokok yang telah dirancang atau didesain, kemudian dijalankan sepenuhnya. Pada tahapan dilakukannya implemenatasi sistem yakni dengan peng-codean menggunakan Bahasa pemrograman *Phyton.*

5. Evaluasi dan pengujian

Pengujian ini dilakukan untuk menguji tingkat akurasi hasil dari metode yang diusulkan. Adapun pengujian dalam penelitian ini menggunakan *confusion matrix.*

3.4 Tahap Pengumpulan Data

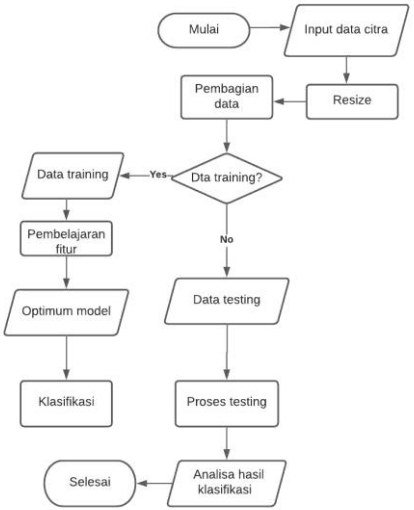
Pada penelitian ini menggunakan data citra penyakit daun kentang sebagai data input. Data tersebut diperoleh secara online dari *Kaggle Datasets* yang terbagi menjadi tiga kelas yaitu *Early Blight* (bercak kering), *Late Blight* (busuk daun) dan *Healthy* (daun sehat)dengan jumlah 2152 data, kemudian terbagi pada proses training dengan jumlah 1700 data dan proses testing 452. Selanjutnya diklasifikasi untuk mendeteksi adanya penyakit daun kentang secara otomatis dengan cepat dan tepat. Sampel data citra *Early Blight* dan *Late Blight* ditunjukkan pada tabel berikut :

Tabel 3.1 Dataset Penyakit Daun Kentang

No	Nama Penyakit Daun	Gambar Penyakit daun
1.	*Early Blight*	
2.	*Late Blight*	

3.5 Tahap Pengolahan Data

Pada tahap ini merupakan proses yang dilakukan untuk deteksi penyakit daun kentang menggunakan CNN *GoogleNet*berdasarkan klasifikasi data citra dijelaskan sebagai berikut:



Gambar 3.3 SistemPenelitian

1. Input data citra penyakit daun kentang diperoleh dari*Kaggle*. Data berisi data citra *Early Blight, Late Blight,* dan *Healthy.*
2. *Pre-processing* merupakan langkah awal pengelolahan data yang dilakukan dengan tujuan mendapatkan data yang optimal. Dengan melakukan resize citra daun pada data citra dari ukuran citra 256 x 256 menjadi citra dengan berukuran 100 x 100.Data yang telah melalui tahapan *pre-processing* tersebut yang nantinya akan digunakan sebagai data input pada proses klasifikasi.Langkah-langkahpre-processing data citra

dilakukan sesuai dengan diagram alir pada gambar berikut :

Gambar 3.4 Diagram alir *Pre-Processing*

3. Pembelajaran fitur dan klasifikasi CNN *GoogleNet*

Proses pembelajaran fitur dan klasifikasi dilakukan dalam satu arsitektur.Arsitektur yang digunakan pada penelitian ini adalah CNN *GoogleNet*,dimana dalam arsitektur *GoogleNet* terdiri dari beberapa layer. Layer-layertersebut antara lain *convolution layer, ReLU layer, pooling layer, dan fully connected layer.* Fitur-fitur data citra dipelajari dengan *convolution layer, ReLU layer, dan pooling layer.* Hasil dari proses pembelajaran fitur data citra diklasifikasi pada fully connected layer dengan penentuan kelas berdasarkan probabilitas kelas tertinggi pada *softmax layer*. Proses pembelajaran sistem klasifikasi menghasilkan model optimum yang selanjutnya diuji

pada proses testing. Pada penelitian ini dilakukan uji coba pembagian data, inisialisasi probabilitas *dropout layer*, dan *batchsize* untuk mendapatkan nilai optimal. Uji coba dilakukan dengan menggunakan data training 80% dan data testing 20%. Percobaan inisialisasi probabilitas *dropout* yaitu 05, 0.6, dan 0.7 dengan nilai *batchsize* 16, 32, 64, dan 128. Model optimum yang dihasilkan di evaluasi menggunakan indikator akurasi.

4. Analisa Hasil Klasifikasi

   Analisa hasil klasifikasi ini dengan cara evaluasi menggunakan Confusion Matrix dengan menilai tingkat akurasi metode *Convolutional Neural Network*. Confusion Matrix tersebut merupakan tabel yang berisikan hasil perhitungan klasifikasi secara keseluruan evaluasi data diukur dengan akurasi. .Dari hasil tersebut nantinya dapat dilihat model jaringan yang paling optimal untuk melakukan klasifikasi penyakit daun kentang . Selain itu, dari analisa hasil klasifikasi dapat dilihat apakah model jaringan yang dibentuk dapat diterapkan dalam deteksi penyakit daun kentang.

3.6 Kebutuhan Alat dan Bahan

a. Kebutuhan Perangkat Keras
   1. Laptop Asus dengan spesifikasi Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz
   2. RAM 4,00GB
   3. System Type 64-Bit

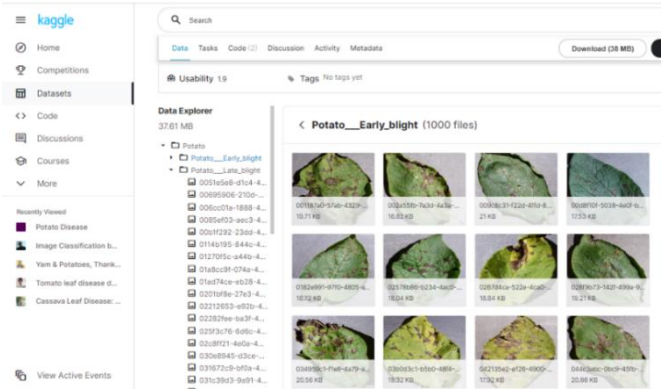
b. Kebutuhan Perangkat Lunak
   1. Microsoft Windows 10 sebagai notebook.
   2. Microsoft Word 2016 sebagai media penulisan.
   3. Google Colaboratory Python sebagai pengolahan data dan melihat hasil akurasi.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Deskripsi Data

Sebagaimana yang telah dipaparkan pada bab 3, deteksi penyakit daun kentang secara automatis berdasarkan klasifikasi data citra dilakukan denganmengimplementasikan arsitektur *GoogleNet* pada algoritma CNN. Data yangdigunakan pada penelitian ini merupakan data citra daun kentang yang diperoleh dari website Kaggle Data citra penyakit daun kentang sebanyak 2.152. Pada penelitian ini menggunakan 2 kelas yaitu kelas *Early Blight* dan *Late Blight*. Hasil download data ditunjukkan pada Gambar 4.1.



Gambar 4.1 Hasil download data *Kaggle*

4.2 Memuat Dataset

Pada tahap ini dilakukan download atau memanggil dataset yang sudah di upload pada Google Drive**.** Kemudian melakukan split dataset awal menjadi data train dan test.

```
[5]
    from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
[ ]  # split dataset awal menjadi data train dan test
     X_train, X_test, y_train, y_test = train_test_split(
         X, y, test_size=0.20, random_state=100)
```

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

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

```
[ ]  print('train size', len(df_tr))
     print('test size', len(df_te))

     train size 1721
     test size 431
```

Gambar 4.2 Memuat dataset

4.3 Model *Convolutional Neural Network* (CNN)

Pada tahap ini dilakukan pembuatan model CNN pada sistem yang akan dibuat. Sebelum model dibuat, hal utama yang dilakukan yaitu mengimport library, kemudian input layer untuk membuat model CNN.

```
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
# Helper libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import glob
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
```

Gambar 4.3 Source code import *library*

```
#Build The CNN
model = Sequential() #Create the architecture
model.add(Conv2D(64, (5, 5), activation='relu',
input_shape=(128,128,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Gambar 4.4 source code model CNN

Pada gambar tersebut dijabarkan source code yang akan digunakan untuk membuat model CNN dengan keterangan sebagai berikut:

a. Menambahkan sequential untuk model *neural network* berupa *sequential network* yang merupakan dasar dari inisialisasi neural network.
b. Menambahkan *Convolution2D layer*, *layer convolusi2d* digunakan pada data berupa gambar, berfungsi untuk mengenali gambar berdasarkan piksel-piksel yang terdapat pada gambar, dengan menggunakan fungsi activation relu. Parameter filter diisi sebanyak 64, berarti jumlah filter yang dikeluarkan dalam proses konvolusi sebanyak 64 dengan ukuran *(kernel_size=(5, 5*)). *Input_shape (128, 128, 3*) dengan arti semua ukuran gambar yang masuk ke dalam input layer berukuran 128x128 pixels dan berukuran 3 lapisan warna dalam bentuk matriks. *ReLu (Rectifier Linear Unit*) activation yang digunakan ini adalah fungsi yang umum digunakan untuk CNN.
c. Menambahkan *MaxPooling2D layer* dengan ukuran 2x2 dengan cara kerja mengambil nilai maximum dan hasil analisisnya akan lebih cepat karena ukuran *feature maps* yang kecil dari data yang diolah.untuk mengurangi resolusi gambar untuk mempertahankan informasi pada gambar tersebut.
d. Pada baris 6,7,8 menambahkan proses hasil feature maps setelah *maxpooling* pada baris selanjutnya akan menghasilkan hasil yang lebih detail pada *feature maps* yang baru. Kemudian dilakukan *maxpooling* guna memperkecil ukuran. Sehingga feature yang ditangkan benar-benar berisi dan compact. Proses tersebut digunakan untuk meningkatkan performa dari model CNN yang telah dibuat, dengan

menambahkan layer proses *convolution2d dan maxpooling2d.*

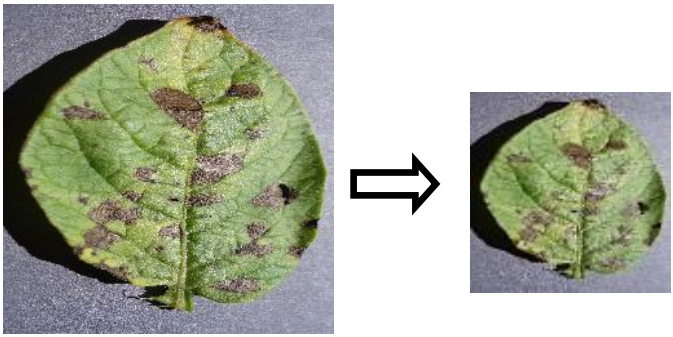
e. Tahap selanjutnya memberi *flatten*. Proses tersebut membuat semua matriks menjadi berukuran *single vector* yang akan menjadi inputan bagi *neural network*.
f. Pada baris 11 selanjutnya memberi dense atau menambah *hidden layer* tertulis sebesar 128 yang berarti terdapat 128 *neuron* yang ada pada layer tersebut. Kemudian parameter selanjutnya adalah *activation='relu'.*
g. Selanjutnya baris 12 merupakan pendefinisian *output layer*. Jenis datanya berupa kategori (*glioma, meningioma, pituitary*) maka terdapat 3 neuron sesuai dengan banyaknya macam label pada dataset. Terdapat dua jenis untuk prediksi label dari *neural network*, *sigmoid dan softmax*, untuk data kategori dalam model ini yang digunakan adalah *softmax*.

4.4 *Pre – Processing*

Sebelum data yang didapatkan bisa diolah, maka dilakukan tahapan preprocessing agar dihasilkan data berupa gambar seperti yang diharapkan. Kemudian sebelum gambar dapat dimasukkan ke dalam struktur yang diusulkan langkah selanjutnya yang dilakukan adalah dengan mengubah piksel data citra penyakit daun kentang yang sudah sama berukuran 256 × 256 di resize menjadi ukuran 100 × 100 dengan tujuan mempercepat komputasi.Hasil perubahan data citra awal hingga data citra baru yang telah di resize ditunjukkan pada Gambar 4.5.

```
[ ]  # re-size all the images to this
     IMAGE_SIZE = [100, 100]

     valid_path = '/content/drive/MyDrive/dataset/test'
     valid_path = '/content/drive/MyDrive/dataset/train'
```

Gambar 4.5 Source code *PreProcessing*



Gambar 4.6 Hasil *PreProcessing*

Setelah pre-processing selesai dilakukan, langkah selanjutnya yaitu pembagian data untuk proses training dan proses testing. Pada penelitian ini data dibagi menjadi 80% data training dan 20% data testing. Data training digunakan untuk melatih model *GoogleNet* untuk mendapatkan model klasifikasi terbaik, data data testing digunakan sebagai data pengujian yang menerapkan model hasil proses training sebelumnya.

4.5 Pelatihan Model *GoogleNet*

Sama seperti model machine learning, tahapan yang dilakukan pada model deep learning setelah melakukan pre-processing yaitu proses pembelajaran fitur untuk mengenali ciri-ciri pada data citra dan juga klasifikasi. Perbedaan proses pembelajaran fitur dan proses klasifikasi antara machine learning dan deep learning terdapat pada model machine learning proses pembelajaran fitur dan klasifikasi dilakukan secara terpisah, sedangkan pada model deep learning proses ekstraksi fitur dan klasifikasi dilakukan pada satu proses yang sama. Proses pembelajaran fitur umumnya diterapkan pada lapisan konvolusi yang terdapat pada *GoogleNet*. Proses model dengan import library ditunjukan oada Gambar 4.7.

```
# import the libraries as shown below
import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

Gambar 4.7 Source code library *GoogleNet*

Pada code tersebut menggunakan *import InceptionV3* karena *InceptionV3* termasuk modul untuk *GoogleNet* yang merupakan jaringan saraf *convolutional* untuk analisis gambar dan deteksi objek. Proses download data dan struktur yang dihasilkan ditunjukan pada gambar 4.8.

```
.]
  # Here we will be using imagenet weights
  inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
# don't train existing weights
for layer in inception.layers:
    layer.trainable = False
```

Gambar 4.8 Source code download data struktur *GoogleNet*

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 100, 100, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 49, 49, 32)   864         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 49, 49, 32)   96          conv2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 49, 49, 32)   0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 47, 47, 32)   9216        activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 47, 47, 32)   96          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 47, 47, 32)   0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 47, 47, 64)   18432       activation_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 47, 47, 64)   192         conv2d_2[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 47, 47, 64)   0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 23, 23, 64)   0           activation_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 23, 23, 80)   5120        max_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 23, 23, 80)   240         conv2d_3[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 23, 23, 80)   0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 21, 21, 192)  138240      activation_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 21, 21, 192)  576         conv2d_4[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)       (None, 21, 21, 192)  0           batch_normalization_4[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 10, 10, 192)  0           activation_4[0][0]
__________________________________________________________________________________________________
```

conv2d_22 (Conv2D)	(None, 10, 10, 64)	18432	mixed1[0][0]
batch_normalization_22 (BatchNo	(None, 10, 10, 64)	192	conv2d_22[0][0]
activation_22 (Activation)	(None, 10, 10, 64)	0	batch_normalization_22[0][0]
conv2d_20 (Conv2D)	(None, 10, 10, 48)	13824	mixed1[0][0]
conv2d_23 (Conv2D)	(None, 10, 10, 96)	55296	activation_22[0][0]
batch_normalization_20 (BatchNo	(None, 10, 10, 48)	144	conv2d_20[0][0]
batch_normalization_23 (BatchNo	(None, 10, 10, 96)	288	conv2d_23[0][0]
activation_20 (Activation)	(None, 10, 10, 48)	0	batch_normalization_20[0][0]
activation_23 (Activation)	(None, 10, 10, 96)	0	batch_normalization_23[0][0]
average_pooling2d_2 (AveragePoo	(None, 10, 10, 288)	0	mixed1[0][0]
conv2d_19 (Conv2D)	(None, 10, 10, 64)	18432	mixed1[0][0]
conv2d_21 (Conv2D)	(None, 10, 10, 64)	76800	activation_20[0][0]
conv2d_24 (Conv2D)	(None, 10, 10, 96)	82944	activation_23[0][0]
conv2d_25 (Conv2D)	(None, 10, 10, 64)	18432	average_pooling2d_2[0][0]
batch_normalization_19 (BatchNo	(None, 10, 10, 64)	192	conv2d_19[0][0]
batch_normalization_21 (BatchNo	(None, 10, 10, 64)	192	conv2d_21[0][0]
batch_normalization_24 (BatchNo	(None, 10, 10, 96)	288	conv2d_24[0][0]
batch_normalization_25 (BatchNo	(None, 10, 10, 64)	192	conv2d_25[0][0]
activation_19 (Activation)	(None, 10, 10, 64)	0	batch_normalization_19[0][0]
activation_21 (Activation)	(None, 10, 10, 64)	0	batch_normalization_21[0][0]

conv2d_93 (Conv2D)	(None, 1, 1, 192)	393216	average_pooling2d_8[0][0]
batch_normalization_85 (BatchNo	(None, 1, 1, 320)	960	conv2d_85[0][0]
activation_87 (Activation)	(None, 1, 1, 384)	0	batch_normalization_87[0][0]
activation_88 (Activation)	(None, 1, 1, 384)	0	batch_normalization_88[0][0]
activation_91 (Activation)	(None, 1, 1, 384)	0	batch_normalization_91[0][0]
activation_92 (Activation)	(None, 1, 1, 384)	0	batch_normalization_92[0][0]
batch_normalization_93 (BatchNo	(None, 1, 1, 192)	576	conv2d_93[0][0]
activation_85 (Activation)	(None, 1, 1, 320)	0	batch_normalization_85[0][0]
mixed9_1 (Concatenate)	(None, 1, 1, 768)	0	activation_87[0][0] activation_88[0][0]
concatenate_1 (Concatenate)	(None, 1, 1, 768)	0	activation_91[0][0] activation_92[0][0]
activation_93 (Activation)	(None, 1, 1, 192)	0	batch_normalization_93[0][0]
mixed10 (Concatenate)	(None, 1, 1, 2048)	0	activation_85[0][0] mixed9_1[0][0] concatenate_1[0][0] activation_93[0][0]
flatten (Flatten)	(None, 2048)	0	mixed10[0][0]
dense (Dense)	(None, 3)	6147	flatten[0][0]

Total params: 21,808,931
Trainable params: 6,147
Non-trainable params: 21,802,784

Gambar 4.9 Hasil struktur pada layer *GoogleNet*

a. Memuat Dataset dan Flow data

Pada tahap ini akan dilakukan pemuatan dataset dalam sistem yang akan dibangun menggunakan Python GoogleColab. Dan mendefinisikan darimana sumber datanya berasal (dictionary).

```
[35] # Make sure you provide the same target size as initialied for the image size
     training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/dataset/train',
                                                      target_size = (100, 100),
                                                      batch_size = 16,
                                                      class_mode = 'categorical')

     Found 1700 images belonging to 2 classes.

[36] test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/dataset/test',
                                                 target_size = (100, 100),
                                                 batch_size = 16,
                                                 class_mode = 'categorical')

     Found 452 images belonging to 2 classes.

[15]   # useful for getting number of output classes
     folders = glob('/content/drive/MyDrive/dataset/train/*')

[16] folders

     ['/content/drive/MyDrive/dataset/train/early_blight',
      '/content/drive/MyDrive/dataset/train/late_blight']

[17] folders = glob('/content/drive/MyDrive/dataset/test/*')

[18] folders

     ['/content/drive/MyDrive/dataset/test/early_blight',
      '/content/drive/MyDrive/dataset/test/late_blight']
```

Gambar 4.10 Source Code Untuk Memuat Dataset Dalam Sistem

Pada gambar diatas merupakan hasil source code untuk memuat data gambar yang sebelumnya sudah di proses pada ImageDataGenerator dan menampilkan total gambar sesuai

masing" kelasnya, dengan mengatur ukuran citra dan ukuran *Batch Size*nya.

b. Data Training

Tahap ini dilakukan pelatihan model pada dataset yang telah ada. Fungsi dari dilakukannya data training yaitu agar membuat prediksi atau menjalankan fungsi dari sebuah model yang akan dibangun. Pada dasarnya algoritma pada mesin perlu mencari korelasi atau belajar pola dari data yang telah diberikan yaitu data training.

```
# fit the model
# Run the cell. It will take some time to execute
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=10,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

Gambar 4.11 Source code *Training* data

Proses training dijalankan pada model.fit.generator, pada model.fit.generator terdapat step per epoch yaitubatch size untuk tiap tiap epoch, dan epoch terdiri dari 10 iterasi, kegunaan validation_data yaitu untuk memvalidasi model berdasarkan testing set, dan dievaluasi pada setiap epoch berakhir. Kemudian pada validation steps merupakan banyaknya steps / langkah untuk menyelesaikan 1 epoch, disini 1 steps merupakan 1 batch_size / 1 batch_size adalah 16 citra.

c. Pelatihan dan Pengujian Model

Pada model ini untuk menghitung kesalahan model selama proses optimasi pada loss function yang digunakan adalah loss function categorical cross-entropy. Model ini dilatih selama 10 epoch. Selanjutnya adalah melakukan proses pelatihan menggunakan data training dan diuji menggunakan data test yang telah dipersiapkan sebelumnya setelah model sudah dibuat.

Proses klasifikasi citra penyakit daun kentang menggunakan algoritma CNN *GoogleNet* dipengaruhi oleh beberapa hal antara lain pembagian data training dan testing, dropout, dan batchsize. Pada penelitian ini uji cobayang dilakukan yaiu pada pembagian data, dropout, dan batchsize untuk mendapatkan model klasifikasi yang terbaik. Pembagian data yang di uji coba yaitu data training dan testing sebanyank 80% dan 20%. Dimana akan diatur dropout layer yang dilakukan pada uji coba nilai probabilitas dengan inisialisasi probabilitas sama dengan 0.5, 0.6, dan 0.7. Dan pada Batchsize yang diujikan bernilai 16, 32, dan 64. Dan epoch 10. Model terbaik dipilih berdasarkan evaluasi sistem menggunakan confusion matrix dengan nilai akurasi. Pembagian kelas dapat dilihat pada tabel 4.1.

Tabel 4.1 Pembagian data kelas.

Nama Kelas	Data Train 80%	Data Test 20%
Early Blight	848	228
Late Blight	852	224
Jumlah	1700	452

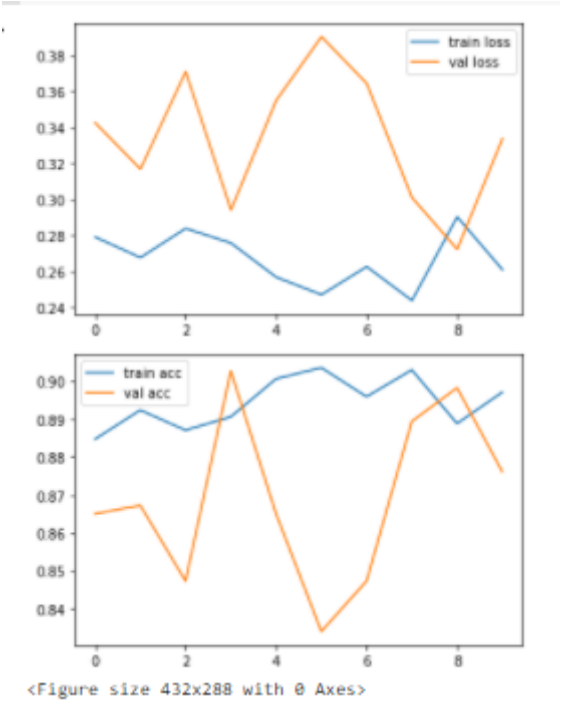
d. Uji coba pembagian data 80% dan 20%

Uji coba dengan pembagian data 80% dan 20% ditunjukkan pada Tabel 4.2.

Tabel 4.2 Hasil evaluasi percobaan perbandingan.

Epoch	Dropout	Batch Size	Akurasi	Sensitifitas	Spesifisitas
10	0.5	16	88.71%	94.54%	93.44%
		32	87.38%	99.45%	93.14%
		64	86.72%	92.89%	92.89%
		128	88.71%	94.31%	93.64%
	0.6	16	85.61%	91.48%	90.63%
		32	89.38%	94.78%	93.45%
		64	87.16%	93.36%	92.92%
		128	86.28%	92.41%	92.85%
	0.7	16	87.16%	90.99%	92.92%
		32	88.71%	94.47%	93.24%
		64	88.71%	94.31%	93.64%
		128	89.15%	95.26%	93.27%

Berdasarkan Tabel 4.2 dapat diketahui akurasi terbaik dari pembagian data training 80% yaitu 89.38% dengan inisialisasi nilai probabilitas pada dropout layer sama dengan 0.6 dan nilai batchsize sama dengan 32. Setiap percobaan batchsize dan dropout menunjukkan rata-rata nilai akurasi tidak stabil. Akan tetapi pada salah satu dropout terdapat kenaikan akurasi dan komputasi yang cepat. Hasil grafik pada proses evaluasi ditunjukan pada gambar 4.12



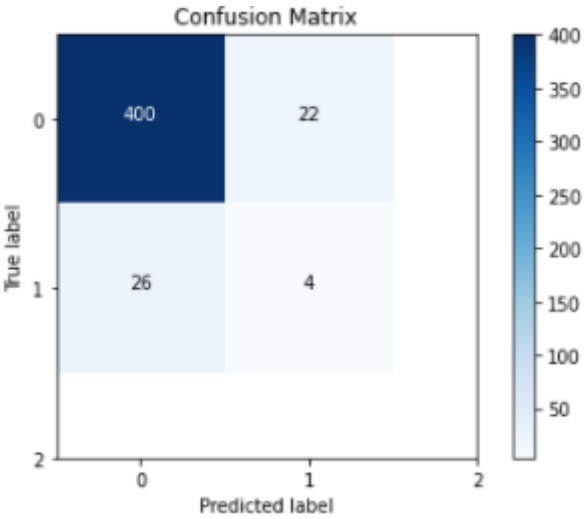
Gambar 4.12 Grafik hasil evaluasi

Pada grafik diatas,bahwa akurasi pada data training adalah 0.89 dan nilai *lossnya* adalah 0.26, Kemudian untuk data *validasinya* adalah 0.88 dan nilai *lossnya* adalah 0.34. 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. Pergerakan akurasi yang dihasilkan pada data test maupun train hampir stabil hal

ini kemungkinan disebabkan oleh jumlah data yang sama pada tiap kelompok.

e. Evaluasi confusion matrix

Dari proses training tersebut menghasilkan confusion matrix pada Gambar 4.13.



Gambar 4.13 Confusion matrix hasil evaluasi

Tabel 4.3 Hasil evaluasi Confusion Matrix

Prediksi	Aktual	
	Early Blight	*Late Blight*
Early Blight	400	22
Late Blight	26	4

Dari tabel confusion matrix pada dapat dilakukan perhitungan akurasi. Berikut contoh perhitungan untuk mendapatkan nilai akurasi untuk hasil evaluasi klasifikasi:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$
$$= \frac{400 + 4}{400 + 22 + 26 + 4} = \frac{404}{452}$$
$$= 0.8938 * 100\% = 89.38\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{400}{400 + 22} = \frac{400}{422}$$
$$= 0.93.45 * 100\% = 93.45\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{400}{400 + 2} = \frac{400}{402}$$
$$= 0.94.78 * 100\% = 94.78\%$$

Perhitungan tersebut dapat dijelaskan sebagai berikut :

1. True Positif (TP) merupakan jumlah data citra *early blight* yang terklasifikasi benar sebagai penyakit oleh sistem klasifikasi.
2. False Positif (FP) merupakan jumlah data citra *late blight* yang terklasifikasi salah sebagai *early blight* oleh sistem klasifikasi.
3. False Negative (FN) merupakan jumlah data citra *early blight* yang terklasifikasi salah sebagai data *late blight* oleh sistem klasifikasi.
4. True Negatif (TN) merupakan jumlah data citra *early blight* yang terklasifikasi benar sebagai *early blight* oleh sistem klasifikasi.

HALAMAN INI SENGAJA DIKOSONGKAN

BAB V

PENUTUP

5.1 Kesimpulan

Dari penelitian mengenai klasifikasi penyakit daun kentang menggunakan *convolutional neural network* (CNN) menggunakan *GoogleNet* dapat disimpulkan bahwa:

1. Deteksi penyakit daun kentang secara automatis menggunakan metode CNN *GoogleNet* bedasarkan klasifikasi data citra dilakukan dengan beberapa tahap. Tahap pertama yaitu dengan melakukan pre-processing data yaitu dengan resize. Hasil yang didapat dari penerapan resize untuk mempercepat komputasi model. Algoritma CNN memiliki beberapa lapisan yang digunakan untuk pembelajaran fitur dan klasifikasi sesuai dengan arsitektur *GoogleNet*. Proses ini dipengaruhi oleh persentase pembagian data training, batchsize dan dropout.
2. Dari beberapa uji coba yang dilakukan berdasarkan ukuran batch size, jumlah epoch, dan dropout didapatkan tingkat akurasi tertinggi sebesar 89,38%. Hasil paling optimal tersebut diperoleh dari arsitektur *GoogleNet* dengan ukuran batch size sebesar 32, epoch berjumlah 10, dan nilai probabilitas dropout sebesar 0.6 dan dengan pembagian data training dan testing sebesar 80% dan 20%.

3. 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.

5.2 Saran

Pada penelitian ini terdapat beberapa kekurangan yang bisa diperbaiki. Hal-hal yang dapat dilakukan untuk penelitian mendatang yaitu dengan menambah lebih banyak data dari sumber lainnya sehingga pola penyakit daun kentang yang dipelajari oleh sistem semakin beragam. Selain itu variansi data citra diharapkan juga lebih beragam dengan menerapkan beberapa metode augmentasi data karena hal tersebut mungkin mampu menambah tingkat akurasi dalam memprediksi gambar.

DAFTAR PUSTAKA

Ansari, M. A. (2020). *PERFORMANCE COMPARISON OF MACHINE LEARNING CLASSIFIERS FOR THE DETECTION OF POTATO LEAF DISEASES*. *2020*(December), 17–19.

Apakah Deep Learning_MMSI BINUS University. (n.d.).

Arifin, M. I., Pembimbing, D., Rachman, F., & Adhitya, R. Y. (2019). *KLASIFIKASI PENYAKIT PADA ORCHIDACEAE MENGGUNAKAN PENGOLAHAN CITRA DENGAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN)*.

Bachri, B. S. (2010). Meyakinkan Validitas Data Melalui Triangulasi Pada Penelitian Kualitatif. *Teknologi Pendidikan*, *10*, 46–62.

Badan Pusat Statistik. (2018). *Badan Pusat Statistik* (pp. 335–358). https://doi.org/10.1055/s-2008-1040325

Dadang, W. (2018). Memahami Kecerdasan Buatan berupa Deep Learning dan Machine Learning. In *Warstek.Com* (pp. 1–10). https://warstek.com/2018/02/06/deepmachinelearning/

Enterprise, J. (2019). *Python untuk Programmer Pemula - Jubilee Enterprise - Google Buku*.

Haq, D. Z., Matematika, P. S., Sains, F., Teknologi, D. A. N., Islam, U., & Sunan, N. (2021). *Angkatan*.

Huda, M. R. K., & Nafi'yah, N. (2020). *Identifikasi Penyakit Daun Kentang Berdasarkan Fitur Warna , Tekstur , dan Bentuk dengan SVM dan KNN Identification of Potato Leaf*

Disease Based on Color , Texture , and Shape Features with. 100–106.

Ilahiyah, S., & Nilogiri, A. (2018). *Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network*. 49–56.

Islam, M., Dinh, A., & Wahid, K. (2017). *Detection of Potato Diseases Using Image Segmentation and Multiclass Support Vector Machine*. 8–11.

Kusumaningrum, T. F. (2018). *IMPLEMENTASI CONVOLUTION NEURAL NETWORK (CNN) UNTUK KLASIFIKASI JAMUR KONSUMSI DI INDONESIA MENGGUNAKAN KERAS*.

Lee, T. (2020). *Health Detection for Potato Leaf with Convolutional Neural Network*. 289–293.

Nisa, C., Puspaningrum, E. Y., & Maulana, H. (2020). *Penerapan Metode Convolutional Neural Network untuk Klasifikasi Penyakit Daun Apel pada Imbalanced Data*. *1*, 169–175.

Rakhmawati, P. U., Pranoto, Y. M., & Setyati, E. (2018). *KLASIFIKASI PENYAKIT DAUN KENTANG BERDASARKAN FITUR TEKSTUR DAN FITUR WARNA MENGGUNAKAN SUPPORT VECTOR MACHINE*. 1–8.

Safitri, V. I. (2017). *DETEKSI PENYAKIT TANAMAN KENTANG MENGGUNAKAN METODE CERTAINTY FACTOR*. *1*(1), 798–805.

Sari, D. F., Swanjaya, D., Informatika, T., Teknik, F., Nusantara, U., & Kediri, P. (2020). *Implementasi Convolutional Neural Network Untuk Identifikasi Penyakit Daun Gambas*. 137–142.

Semangun, H. (1989). *PENYAKIT_PENYAKIT_TANAMAN_HORTIKULTURA_D.pdf*.

Tabloid Sinar Tani. (n.d.). *Menjaga Hasil Panen Kentang Tetap Berkualitas Baik*.

Tiwari, D. (2020). *Potato Leaf Diseases Detection Using Deep Learning*. *Iciccs*, 461–466.

Wijaya, A. Y., & Soelaiman, R. (2016). *Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) pada Caltech 101*. *5*(1).

Lampiran 1
Lembar Bimbingan

LEMBAR BIMBINGAN SKRIPSI

Nama	:	DAHLIATUL FITRIYAH NINGSIH
NIM	:	201769040016
Jurusan	:	Teknik Informatika
Konsentrasi	:	Jaringan
Judul	:	KLASIFIKASI JENIS PENYAKIT DAUN KENTANG MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (CNN) MODEL ARSITEKTUR *GOOGLENET*

No	Hari	Tgl	BAB	Materi Bimbingan	TTD Pembimbing
1.	Rabu	17 Feb 2021	-	Pemaparan usulan topik	
2.	Rabu	24 Feb 2021	-	Pemantapan topik & Judul	
3.	Rabu	03 Mar 2021	I	Revisi BAB I	
4.	Rabu	10 Mar	I & II	Revisi BAB I &	

No	Hari	Tgl	BAB	Materi Bimbingan	TTD Pembimbing
		2021		Revisi BAB II	
5.	Kamis	25 Mar 2021	III	Check plagiasi & Revisi BAB III	
6.	Rabu	31 Maret 2021	III	Revisi BAB III	
7.	Rabu	07 April 2021	III	Revisi format penulisan	
8.	Kamis	15 April 2021	I-III	Revisi BAB I, II & III	
9.	Kamis	20 Mei 2021	-	Pembahasan hasil seminar proposal	
10.	Kamis	27 Mei 2021	-	Pembahasan hasil seminar proposal	
11.	Rabu	02 Juni	I, II, III	Revisi keseluruhan	

	Hari	Tgl	BAB	Materi Bimbingan	TTD Pembimbing
		2021			
	Kamis	10 Juni 2021	III	Revisi penulisan& pembangunan projek	
13.	Selasa	15 Juni 2021	I dan IV	Membahas BAB IV dan proses pembuatan project	
14.	Rabu	23 Juni 2021	IV	Progres pembuatan project	
15.	Rabu	30 Juni 2021	IV	Revisi project	
16.	Sabtu	03 Juli 2021	IV	Revisi project	

No	Hari	Tgl	BAB	Materi Bimbingan	TTD Pembimbing
17.	Kamis	08 Juli 2021	IV	Pengujian	
18.	Kamis	15 Juli 2021	IV	Pengujian	
19.	Kamis	22 Juli 2021	IV	Pengujian & BAB IV	
20.	Rabu	28 Juli 2021	V	Kesimpulan & Saran	

Pasuruan 4 Agustus 2021
Pembimbing

M. Imron Rosadi, M.Kom
NIP.Y 0690213121

Lampiran 2

Coding aplikasi

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.5
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
```

```
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
# Helper libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import glob
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
```

```
#Build The CNN
model = Sequential() #Create the architecture
model.add(Conv2D(64, (5, 5), activation='relu',
input_shape=(128,128,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

```
# import the libraries as shown below
import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
#from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
#import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import pandas as pd
from tensorflow.keras.preprocessing.image import img_to_array, load_img
import random

import tensorflow as tf
import tensorflow.keras.layers as Layers
import tensorflow.keras.activations as Activations
import tensorflow.keras.models as Models
import tensorflow.keras.optimizers as Optimizers
import tensorflow.keras.metrics as Metrics
import tensorflow.keras.utils as Utils
import pandas as pd
import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

```
[ ] # re-size all the images to this
    IMAGE_SIZE = [100, 100]

    valid_path = '/content/drive/MyDrive/dataset/test'
    valid_path = '/content/drive/MyDrive/dataset/train'
```

```
[ ]
    # Here we will be using imagenet weights
    inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
[ ] class InceptionV3():
        @staticmethod
        def build(numChannels, imgRows, imgCols, numClasses,  pooling= "max", activation= "relu"):
            # initialize the model
            model = Sequential()
            inputShape = (imgRows, imgCols, numChannels)

            # add first set of layers: Conv -> Activation -> Pool
            model.add(Conv2D(filters= 6, kernel_size= 5, input_shape= inputShape))
            model.add(Activation(activation))

            if pooling == "max":
                model.add(MaxPooling2D(pool_size= (3, 3), strides= (2, 2)))
            else:
                model.add(AveragePooling2D(pool_size= (7, 7), strides= (2, 2)))

            # add second set of layers: Conv -> Activation -> Pool
            model.add(Conv2D(filters= 16, kernel_size= 5,))
            model.add(Activation(activation))

            if pooling == "avg":
                model.add(AveragePooling2D(pool_size=(7, 7), strides=(2, 2)))
            else:
                model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

            # Flatten -> FC 120 -> Dropout -> Activation
            model.add(Flatten())
            model.add(Dense(64))
            model.add(Dropout(0.5))
            model.add(Activation(activation))

            # FC 84 -> Dropout -> Activation
            model.add(Dense(32))
            model.add(Dropout(0.5))
            model.add(Activation(activation))

            # FC 4-> Softmax
            model.add(Dense(numClasses))
            model.add(Activation("softmax"))

            return model
```

```
[ ] # don't train existing weights
    for layer in inception.layers:
        layer.trainable = False
```

```
[ ]   # useful for getting number of output classes
    folders = glob('/content/drive/MyDrive/dataset/train/*')
```

```
[ ] folders
```

```
['/content/drive/MyDrive/dataset/train/late_blight',
 '/content/drive/MyDrive/dataset/train/early_blight']
```

```
[ ] folders = glob('/content/drive/MyDrive/dataset/test/*')
```

```
[ ] folders
```

```
['/content/drive/MyDrive/dataset/test/late_blight',
 '/content/drive/MyDrive/dataset/test/early_blight']
```

```
[ ] # our layers - you can add more if you want
    x = Flatten()(inception.output)
```

```
[ ] prediction = Dense(len(folders), activation='softmax')(x)
```

```
[ ] test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/dataset/test',
                                                target_size = (100, 100),
                                                batch_size = 16,
                                                class_mode = 'categorical')
```

```
Found 452 images belonging to 2 classes.
```

```
[ ] test_set.labels
```

```
training_set.labels
```

```
[ ] # fit the model
    # Run the cell. It will take some time to execute
    r = model.fit_generator(
      training_set,
      validation_data=test_set,
      epochs=20,
      steps_per_epoch=len(training_set),
      validation_steps=len(test_set)
    )
```

```
[ ] import matplotlib.pyplot as plt
```

```
# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

```
# view the structure of the model
model.summary()
```

```
[ ] # tell the model what cost and optimization method to use
    model.compile(
      loss='categorical_crossentropy',
      optimizer='adam',
      metrics=['accuracy']
    )
```

```
model.compile(optimizer="sgd",
              loss="categorical_crossentropy",
              metrics=[keras.metrics.Precision(), keras.metrics.Recall(), keras.metrics.SpecificityAtSensitivity(0.5),
                       keras.metrics.SensitivityAtSpecificity(0.5), 'accuracy'])
```

```
[ ] # Use the Image Data Generator to import the images from the dataset
    from tensorflow.keras.preprocessing.image import ImageDataGenerator

    train_datagen = ImageDataGenerator(rescale = 1./255,
                                       shear_range = 0.2,
                                       zoom_range = 0.2,
                                       horizontal_flip = True)

    test_datagen = ImageDataGenerator(rescale = 1./255,
                                      shear_range = 0.2,
                                      zoom_range = 0.2,
                                      horizontal_flip = True)
```

```
[ ] # Make sure you provide the same target size as initialied for the image size
    training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/dataset/train',
                                                     target_size = (100, 100),
                                                     batch_size = 16,
                                                     class_mode = 'categorical')
```

```
Found 1700 images belonging to 2 classes.
```

```
from keras import metrics

model.compile(loss='mean_squared_error', optimizer='sgd',
              metrics=[metrics.mae,
                       metrics.categorical_accuracy])
```

```
from sklearn import metrics
```

```
from sklearn.metrics import confusion_matrix
conf = metrics.confusion_matrix(test_set.labels, y_pred)
conf
```

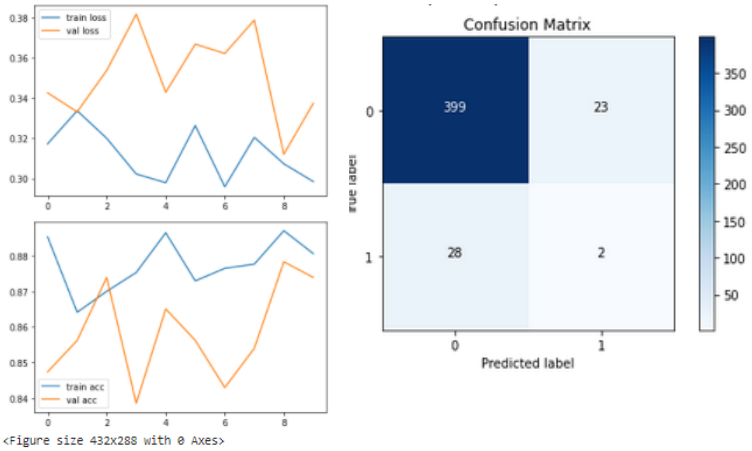
```
model.compile(optimizer="sgd",
              loss="categorical_crossentropy",
              metrics=[keras.metrics.Precision(name='precision'),
                       keras.metrics.Recall(name='recall'),
                       keras.metrics.SpecificityAtSensitivity(0.5, name='specificity_at_sensitivity'),
                       keras.metrics.SensitivityAtSpecificity(0.5, name='sensitivity_at_specificity'),
                       'accuracy'])
```

```
import itertools
classes = [0, 1, 2]
# plot confusion matrix
plt.imshow(conf, interpolation='nearest', cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)

fmt = 'd'
thresh = conf.max() / 2.
for i, j in itertools.product(range(conf.shape[0]), range(conf.shape[1])):
    plt.text(j, i, format(conf[i, j], fmt),
             horizontalalignment="center",
             color="black" if conf[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

Lampiran 3

Hasil perbandingan

a. Epoch 10, Dropout 0.5, Batch Size 16



<Figure size 432x288 with 0 Axes>

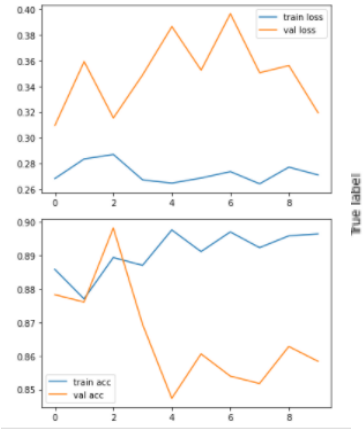
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{399 + 2}{399 + 23 + 28 + 2}$$

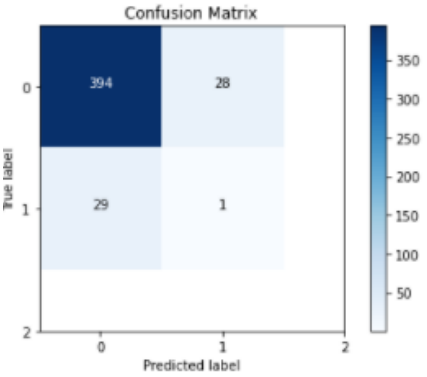
$$= \frac{401}{452} = 0.8871 * 100\% = 88.71\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{399}{399 + 23} = \frac{399}{422}$$
$$= 0.9454 * 100\% = 94.54\%$$

$$Spesifisitas = \frac{TP}{TP + \square N} = \frac{399}{399 + 28} = \frac{399}{427}$$
$$= 0.9344 * 100\% = 93.44\%$$

b. Epoch 10, Dropout 0.5, Batch Size 32



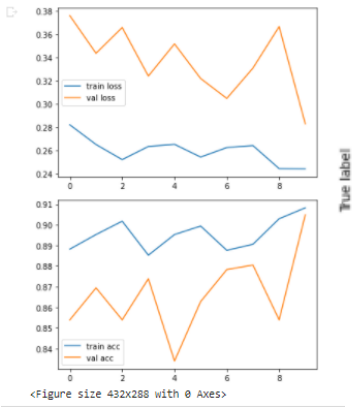


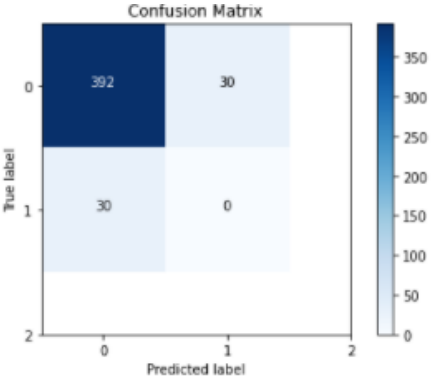
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{392 + 0}{394 + 28 + 29 + 1}$$
$$= \frac{395}{452} = 0.8738 * 100\% = 87.38\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{394}{394 + 28} = \frac{394}{422}$$
$$= 0.9945 * 100\% = 99.45\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{394}{394 + 29} = \frac{394}{427}$$
$$= 0.9314 * 100\% = 93.14\%$$

c. Epoch 10, Dropout 0.5, Batch Size 64



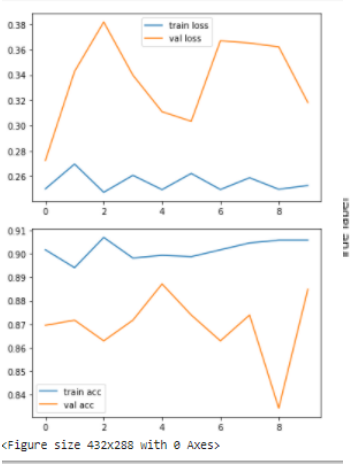


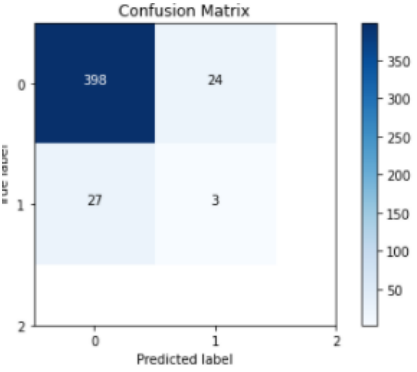
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{392 + 0}{392 + 30 + 30 + 0}$$
$$= \frac{392}{452} = 0.8672 * 100\% = 86.72\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{392}{392 + 30} = \frac{392}{422}$$
$$= 0.9289 * 100\% = 92.89\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{392}{392 + 30} = \frac{392}{422}$$
$$= 0.9289 * 100\% = 92.89\%$$

d. Epoch 10, Dropout 0.5, Batch Size 128



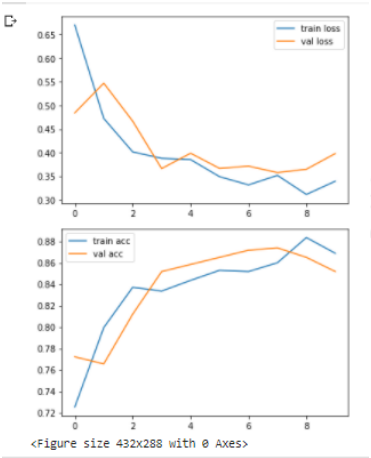


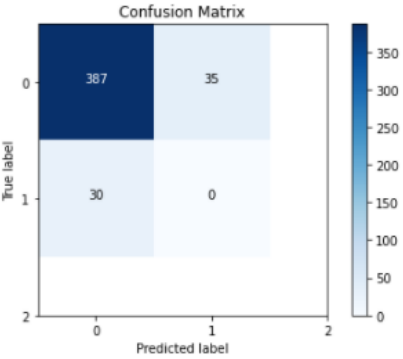
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{398 + 3}{398 + 24 + 27 + 3}$$
$$= \frac{401}{452} = 0.8871 * 100\% = 88.71\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{398}{398 + 24} = \frac{398}{422}$$
$$= 0.9431 * 100\% = 94.31\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{398}{398 + 27} = \frac{398}{425}$$
$$= 0.9364 * 100\% = 93.64\%$$

e. Epoch 10, Dropout 0.6, Batch Size 16



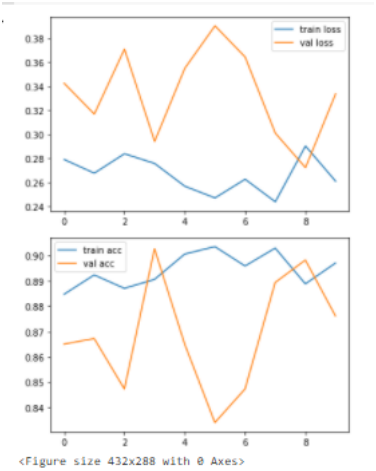


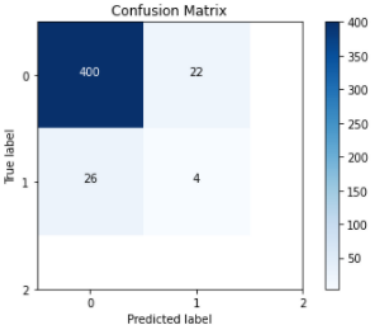
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{387 + 0}{387 + 35 + 30 + 0}$$
$$= \frac{387}{452} = 0.8561 * 100\% = 85.61\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{387}{387 + 35} = \frac{387}{422}$$
$$= 0.9148 * 100\% = 91.48\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{387}{387 + 30} = \frac{387}{427}$$
$$= 0.9063 * 100\% = 90.63\%$$

f. Epoch 10, Dropout 0.6, Batch Size 32



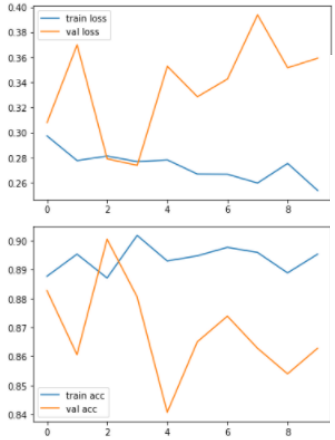


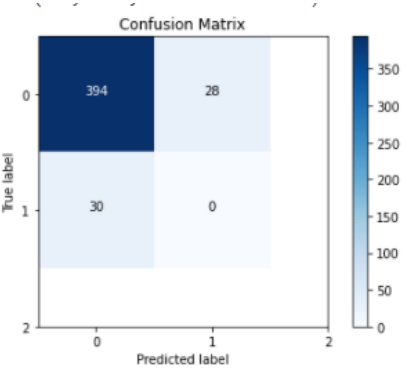
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{400 + 4}{400 + 22 + 26 + 4}$$
$$= \frac{404}{452} = 0.8938 * 100\% = 89.38\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{400}{400 + 22} = \frac{400}{422}$$
$$= 0.93.45 * 100\% = 93.45\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{400}{400 + 2} = \frac{400}{402}$$
$$= 0.94.78 * 100\% = 94.78\%$$

g. Epoch 10, Dropout 0.6, Batch Size 64





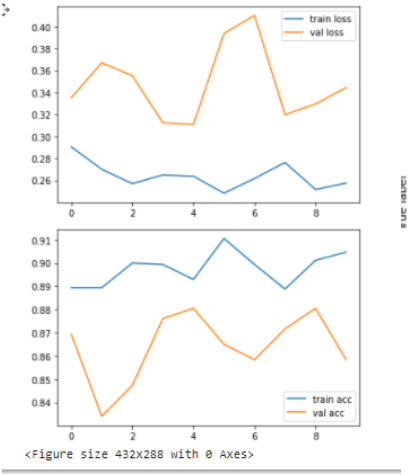
<Figure size 432x288 with 0 Axes>

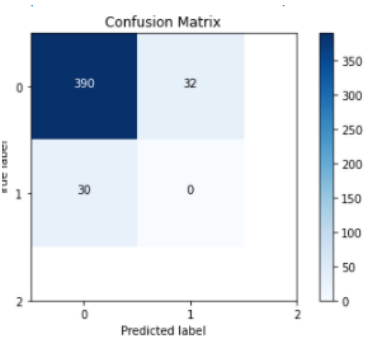
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{394 + 0}{394 + 28 + 30 + 0}$$
$$= \frac{394}{452} = 0.87.16 * 100\% = 87.16\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{394}{394 + 28} = \frac{394}{422}$$
$$= 0.93.36 * 100\% = 93.36\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{394}{394 + 30} = \frac{394}{424}$$
$$= 0.9292 * 100\% = 92.92\%$$

h. Epoch 10, Dropout 0.6, Batch Size 128



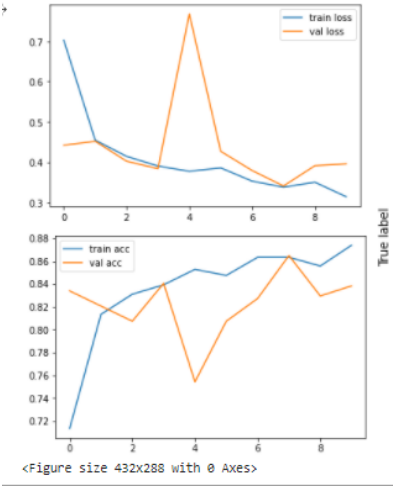


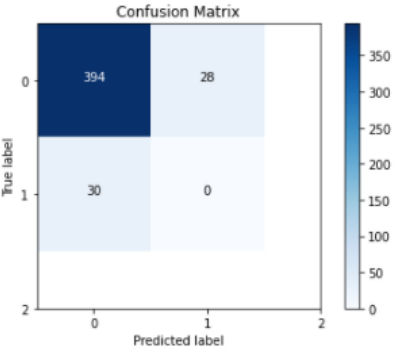
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{390 + 0}{390 + 32 + 30 + 0}$$
$$= \frac{390}{452} = 0.8628 * 100\% = 86.28\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{390}{390 + 32} = \frac{390}{422}$$
$$= 0.9241 * 100\% = 92.41\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{390}{390 + 30} = \frac{390}{420}$$
$$= 0.9285 * 100\% = 92.85\%$$

i. Epoch 10, Dropout 0.7, Batch Size 16



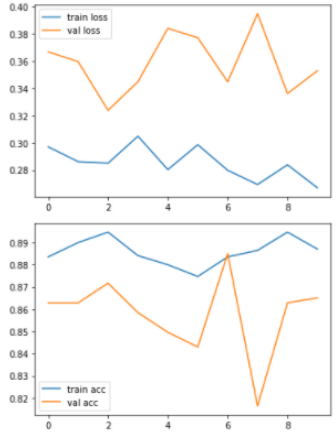


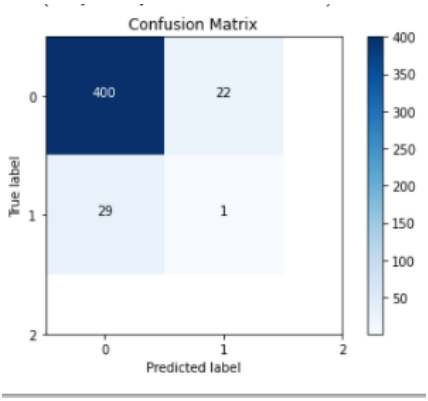
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{394 + 0}{394 + 28 + 30 + 0}$$
$$= \frac{394}{452} = 0.8716 * 100\% = 87.16\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{394}{394 + 28} = \frac{394}{422}$$
$$= 0.9099 * 100\% = 90.99\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{394}{394 + 30} = \frac{394}{424}$$
$$= 0.9292 * 100\% = 92.92\%$$

j. Epoch 10, Dropout 0.7, Batch Size 32



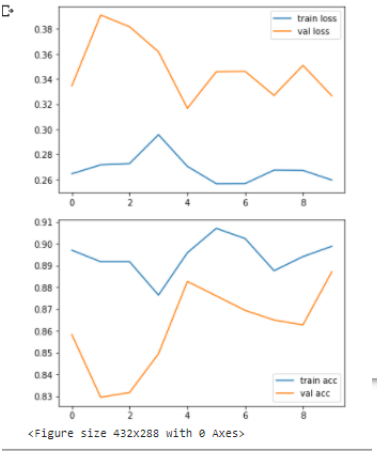


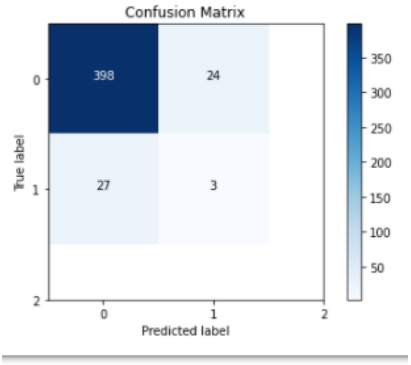
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{400 + 1}{400 + 22 + 29 + 1}$$
$$= \frac{401}{452} = 0.8871 * 100\% = 88.71\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{400}{400 + 22} = \frac{400}{422}$$
$$= 0.93.45 * 100\% = 93.45\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{400}{400 + 29} = \frac{400}{429}$$
$$= 0.9324 * 100\% = 93.24\%$$

k. Epoch 10, Dropout 0.7, Batch Size 64



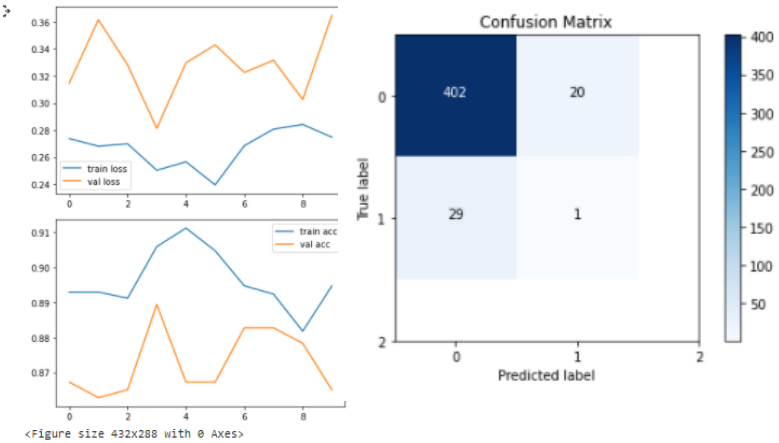


$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{398 + 3}{398 + 24 + 27 + 3}$$
$$= \frac{401}{452} = 0.8871 * 100\% = 88.71\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{398}{398 + 24} = \frac{398}{422}$$
$$= 0.9431 * 100\% = 94.31\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{398}{398 + 27} = \frac{398}{435}$$
$$= 0.9364 * 100\% = 93.64\%$$

I. Epoch 10, Dropout 0.7, Batch Size 128



<Figure size 432x288 with 0 Axes>

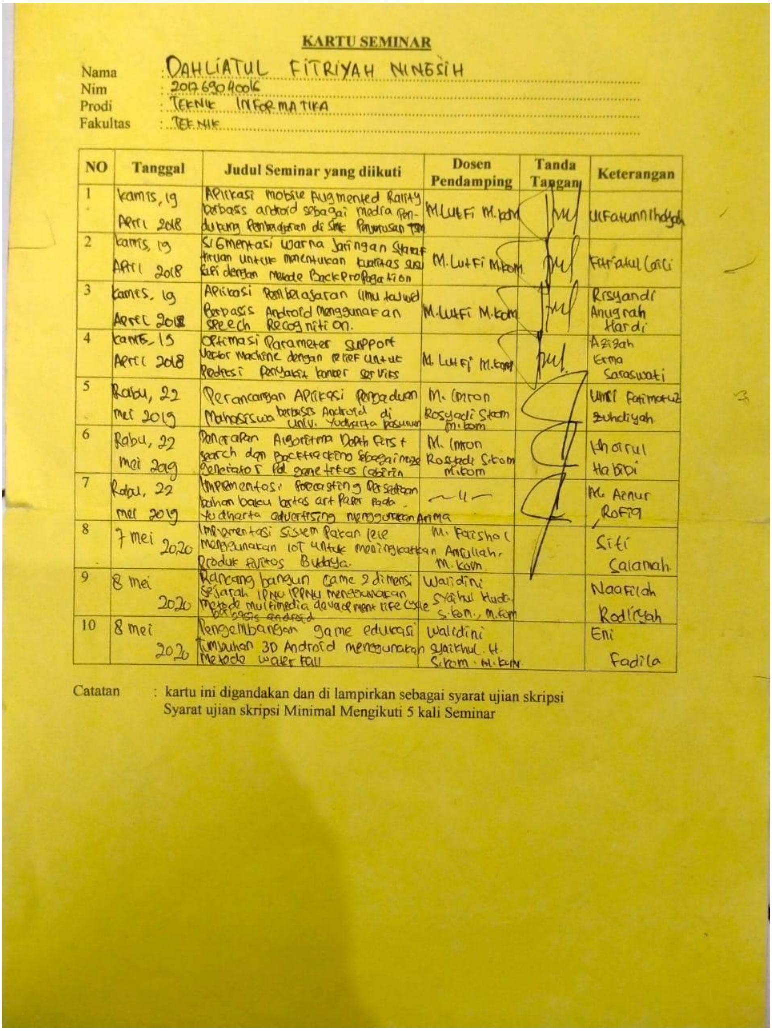
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{402 + 1}{402 + 20 + 29 + 1}$$
$$= \frac{403}{452} = 0.8915 * 100\% = 89.15\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{402}{402 + 20} = \frac{402}{422}$$
$$= 0.9526 * 100\% = 95.26\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{402}{402 + 29} = \frac{402}{431}$$
$$= 0.9327 * 100\% = 93.27\%$$

Lampiran 4

Bukti kartu seminar



KARTU SEMINAR

Nama : DAHLIATUL FITRIYAH NINGSIH
Nim : 2017690400 16
Prodi : TEKNIK INFORMATIKA
Fakultas : TEKNIK

NO	Tanggal	Judul Seminar yang diikuti	Dosen Pendamping	Tanda Tangan	Keterangan
1	kamis, 19 April 2018	Aplikasi mobile Augmented Reality berbasis android sebagai media pendukung Pembelajaran di SMK Penyusunan TGM	M.Lutfi M.kom	[signature]	Ulfatunnihdiyah
2	kamis, 19 April 2018	Sigmentasi warna Jaringan Syaraf tiruan untuk menentukan kualitas susu sapi dengan metode BackPropagation	M.Lutfi M.kom	[signature]	Fitriatul Laili
3	kamis, 19 April 2018	Aplikasi Pembelajaran ilmu tajwid Berbasis Android menggunakan speech Recognition.	M.Lutfi M.kom	[signature]	Risyandi Anugrah Hardi
4	kamis, 19 April 2018	Optimasi Parameter support vector machine dengan relief untuk Prediksi Penyakit kanker serviks	M. Lutfi M.kom	[signature]	Azizah Erma Saraswati
5	Rabu, 22 mei 2019	Perancangan Aplikasi Pengaduan Mahasiswa berbasis Android di Univ. Yudharta pasuruan	M. Imron Rosyadi S.kom M.kom	[signature]	Umi Fatimatuz Zuhdiyah
6	Rabu, 22 mei 2019	Penerapan Algoritma Depth First search dan Backtracking sebagai maze generator pd game tetris labirin	M. Imron Rosyadi S.kom M.kom	[signature]	Khoirul Habibi
7	Rabu, 22 mei 2019	Implementasi forecasting persediaan bahan baku kertas art paper pada Yudharta advertising menggunakan Arima	—"—	[signature]	M. Aenur Rofiq
8	7 mei 2020	Implementasi Sistem Pakan lele menggunakan IoT untuk meningkatkan Produk Fitos Budaya.	M. Faishol Amrullah, M.kom	[signature]	Siti Salamah.
9	8 mei 2020	Rancang bangun Game 2 dimensi Sejarah IPNU IPPNU menggunakan Metode multimedia development life cycle berbasis android	Waliidini Syaihul Huda S.kom., M.kom		Naafilah Rodliyah
10	8 mei 2020	Pengembangan game edukasi Tumbuhan 3D Android menggunakan Metode water fall	Walidini Syaikhul H. S.kom. M.kom		Eni Fadila

Catatan : kartu ini digandakan dan di lampirkan sebagai syarat ujian skripsi
Syarat ujian skripsi Minimal Mengikuti 5 kali Seminar

Lampiran 5
Lembar Bebas Plagiasi

UNIVERSITAS YUDHARTA
PASURUAN
FAKULTAS TEKNIK
Kantor Pusat :
Jl. Yudharta No. 07 (Pesantren Ngalah) Sengonagung Purwosari Pasuruan Telp./ Fax. 0343-611186
e-mail: fakultasteknik@yudharta.ac.id

SURAT KETERANGAN BEBAS PLAGIASI
Nomor : 0338/S9/FT.UYP/II/08/2021

Yang bertanda tangan dibawah ini:
Nama : Misbach Munir, ST., MT
NIP.Y : 0690201015
Jabatan : Dekan Fakultas Teknik

Dengan ini menerangkan bahwa skripsi atas nama mahasiswa :
Nama : Dahliatul Fitriyah Ningsih
NIM : 201769040016
Prodi : Teknik Informatika
Judul Skripsi : Lasifikasi Jenis Penyakit Daun Kentang Menggunakan Convolutional Neural Network (CNN) Model Arsitektur Googlenet
Hasil Plagiasi : 10%

Demikian surat keterangan ini kami buat untuk digunakan sebagaimana mestinya.

Pasuruan, 28 Agustus 2021
Dekan Fakultas Teknik

Misbach Munir, ST., MT.
NIP.Y. 0690201015

Lampiran 6
Curriculum Vitae

CURRICULUM VITAE

PROFIL

Nama	: Dahliatul Fitriyah Ningsih
TTL	: Pasuruan, 18 Januari 1999
Status	: Belum kawin
Jenis Kelamin	: Perempuan
Alamat	: Jln. Pegadaian Sumbertejo-Tejowangi, Purwosari
No. Hp	: 085855971147

RIWAYAT PENDIDIKAN

2005 – 2011	: MI Miftahul Huda tejowangi
2011 – 2014	: MTs Miftahul Huda Tejowangi
2014 – 2017	: SMAN 1 Purwosari

PENGALAMAN ORGANISASI

2013- 2014	: Pramuka
2017 – 2020	: Himpunan Mahasiswa Informatika
2018 – 2020	: Pimpinan Komisariat Perguruan Tinggi Ipnu-Ippnu

KEMAMPUAN

Mampu mengoperasikan Microsoft Office Word

Mampu mengoperasikan Microsoft Excel

Video Editing