KLASIFIKASI PENYAKIT PADA DAUN TANAMAN PADI MENGGUNAKAN ARSITEKTUR *DENSENET-169*

SKRIPSI

OLEH:

MUHAMMAD YAZID ABUD ASSEWETH
188160054

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MEDAN AREA
MEDAN
2024

KLASIFIKASI PENYAKIT PADA DAUN TANAMAN PADI MENGGUNAKAN ARSITEKTUR *DENSENET-169*

SKRIPSI

Diajukan sebagai salah satu Syarat untuk Memperoleh
Gelar Sarjana di Fakultas Teknik
Universitas Medan Area

OLEH:

MUHAMMAD YAZID ABUD ASSEWETH
188160054

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MEDAN AREA
MEDAN
2024

LEMBAR PENGESAHAN

Judul Skripsi	: Klasifikasi Penyakit Pada Daun Tanaman Padi Menggunakan Arsitektur *DenseNet-169*
Nama	: Muhammad Yazid Abud Asseweth
NPM	: 188160054
Fakultas	: Teknik

Disetujui oleh
Komisi Pembimbing

Muhathir, S.T., M.Kom
Pembimbing

Diketahui Oleh:

Dr. Eng. Supriatno, S.T., M.T
Dekan Fakultas Teknik

Rizki Muliono, S.Kom., M.Kom
Ka. Prodi Teknik Informatika

Tanggal Lulus : 1 April 2024

HALAMAN PERNYATAAN

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

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

Medan, 1 April 2024
Penulis,

Muhammad Yazid Abud Asseweth
NPM 188160054

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

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

Nama	: Muhammad Yazid Abud Asseweth
NPM	: 188160054
Program Studi	: Teknik Informatika
Fakultas	: Teknik
Jenis karya	: Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Medan Area **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul :

Klasifikasi Penyakit Pada Daun Tanaman Padi Menggunakan Arsitektur DenseNet-169

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

Dibuat di : Medan
Pada tanggal : 1 April 2024
Yang menyatakan

(Muhammad Yazid Abud Asseweth)

ABSTRAK

Kestabilan produksi padi harus dijaga karena kebutuhan beras sebagai makanan pokok yang terus meningkat setiap tahunnya. Salah satu penyebab penurunan produksi padi adalah terserang penyakit. Sulitnya untuk mendeteksi penyakit pada tanaman padi disebabkan kurangnya pengetahuan dan tenaga profesional. Penelitian ini bertujuan untuk mendeteksi penyakit pada daun tanaman padi menggunakan Convolutional Neural Network (CNN) dengan arsitektur DenseNet-169. Dataset penyakit diperoleh dari situs Kaggle dengan jumlah data sebanyak 240 citra yang terdiri dari penyakit blight, blast, dan tungro. Proses augmentasi dilakukan untuk memperbanyak dataset. Dataset dibagi menjadi data latih dan data uji. Data latih kemudian dilakukan proses augmentasi untuk memperbanyak data. Dataset hasil augmentasi dibagi menjadi data latih dan data testing. Proses tuning hyperparameter menggunakan grid search dan random search dilakukan untuk mendapatkan nilai hyperparameter terbaik. Hasil akurasi terbaik diperoleh model DenseNet-169 dengan menggunakan nilai hyperparameter dari tuning menggunakan grid search. Model grid search memperoleh nilai accuracy sebesar 93%, recall 93%, precision 93%, dan f1-score 93%. Dengan demikian dapat disimpulkan bahwa arsitektur DenseNet-169 dengan tuning hyperparameter menggunakan grid search memiliki akurasi terbaik.

Kata Kunci: **DenseNet169, Penyakit Daun Padi, *Convolutional Neural Network*, Klasifikasi, *Tuning Hyperparameter.***

ABSTRACT

The stability of rice production must be maintained because the need for rice as a staple food continues to increase every year. One of the causes of decreased rice production is disease. The difficulty of detecting diseases in rice plants is due to a lack of knowledge and professional staff. This research aims to detect diseases in rice plant leaves using Convolutional Neural Network (CNN) with the DenseNet-169 architecture. The disease dataset was obtained from the Kaggle site with a total of 240 images consisting of blight, blast and tungro diseases. The augmentation process is carried out to increase the dataset. The dataset is divided into training data and test data. The training data is then subjected to an augmentation process to increase the data. The augmented dataset is divided into training data and testing data. The hyperparameter tuning process using grid search and random search is carried out to get the best hyperparameter values. The best accuracy results were obtained by the DenseNet-169 model using hyperparameter values from tuning using grid search. The grid search model obtained an accuracy value of 93%, recall 93%, precision 93%, and f1-score 93%. Thus it can be concluded that the DenseNet-169 architecture with hyperparameter tuning using grid search has the best accuracy.

Keywords: DenseNet169, Rice Leaf Disease, Convolutional Neural Network, Classification, Hyperparameter Tuning.

RIWAYAT HIDUP

Penulis merupakan putra atau anak ke-3 (tiga) dari ayah H. Abud Ali Asseweth dan ibu Hj. Desi Ani Fitri Nasution yang dilahirkan di Medan, Sumatera Utara pada tanggal 26 Juli 2000.

Pada tahun 2015 Penulis terdaftar sebagai siswa SMK Negeri 1 Percut Sei Tuan Jurusan Teknik Komputer dan Jaringan, selama masa persekolahan, Penulis melaksanakan Praktek Kerja Lapangan (PKL) yang bertempat di PT. Telekomunikasi Indonesia, Tbk dan PT. Media Antar Nusa. Pada tahun 2018, Penulis lulus dari SMK Negeri 1 Percut Sei Tuan, lalu kemudian terdaftar sebagai mahasiswa Fakultas Teknik Prodi Teknik Informatika Universitas Medan Area.

Selamat mengikuti perkuliahan, Penulis mengikuti Studi Independen Bersertifikat di Dicoding (PT. Presentologics) dengan paket pelatihan Pengembang Aplikasi Android dan Multi-Platform selama 3 bulan.

KATA PENGANTAR

Dengan mengucap puja dan puji syukur kehadirat Allah SWT yang telah menganugerahkan segala rahmat, nikmat dan karunianya sehingga penulis dapat menyelesaikan penyusunan skripsi ini, adapun judul dari penelitian ini yaitu **"Klasifikasi Penyakit Pada Daun Tanaman Padi Menggunakan Arsitektur DenseNet-169"**. Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Program Strata-1 pada Fakultas Teknik Program Studi Teknik Informatika di Universitas Medan Area.

Dalam proses menyelesaikan skripsi ini, penulis menyadari bahwa tulisan ini masih jauh dari kata sempurna dan juga terdapat banyak kekurangan baik dari segi bahasa, isi, dan tulisan. Penulis juga mengharapkan kritik dan saran yang sifatnya membangun dari para pembaca. Kemudian penulis ingin mengucapkan terima kasih kepada:

1. Bapak Prof. Dr. Dadan Ramdan, M.Eng., M.Sc., selaku Rektor Universitas Medan Area.
2. Bapak Dr. Eng. Supriatno, S.T., M.T., selaku Dekan Fakultas Teknik Universitas Medan Area.
3. Bapak Rizki Muliono, S.Kom., M.Kom., selaku Kepala Program Studi Teknik Informatika dan Ketua Panitia Sidang yang telah membimbing, mengarahkan, dan memberikan motivasi kepada penulis dalam proses menyelesaikan skripsi ini.
4. Bapak Muhathir, S.T., M.Kom., selaku Dosen Pembimbing penulis yang telah meluangkan waktu untuk membimbing penulis dan memberikan ilmu yang bermanfaat sehingga skripsi ini bisa diselesaikan.

5. Bapak Zulfikar Sembiring, S.Kom., M.Kom., selaku Dosen Teknik Informatika yang telah memberikan ilmu yang bermanfaat dan motivasi kepada penulis.
6. Ibu Susilawati, S.Kom., M.Kom., selaku Sekretaris Panitia Sidang Tugas Akhir penulis yang telah memberikan ilmu yang bermanfaat.
7. Bapak Andre Hasudungan Lubis, S.Ti., M.Sc., selaku Dosen Pembanding Tugas Akhir penulis yang telah memberikan ilmu yang bermanfaat dan motivasi kepada penulis.
8. Ibu Nurul Khairina, S.Kom., M.Kom. selaku Dosen Teknik Informatika yang telah memberikan ilmu yang bermanfaat dan motivasi kepada penulis.
9. Seluruh Dosen Teknik Informatika Universitas Medan Area yang selama ini telah membekali penulis dengan ilmu yang sangat bermanfaat.
10. Seluruh Pegawai Universitas Medan Area yang telah membantu dalam proses administrasi.
11. Teristimewa kepada kedua orang tua penulis, Alm. Bapak Abud Ali Asseweth dan Ibu Desi Ani Fitri Nasution yang dengan penuh kasih sayang telah mendidik penulis serta dengan doa restunya penulis dapat menyelesaikan pendidikan hingga perguruan tinggi.
12. Kedua Saudara Penulis, Kakak Balqis dan Abang Hamzah yang selalu mendukung dan mendoakan serta sangat banyak membantu penulis hingga berhasil menyelesaikan pendidikan saat ini.

13. Sahabat jauh penulis, Putri Aisyah Ayuningtika yang selalu mendukung dan mendoakan serta sangat banyak membantu penulis hingga berhasil menyelesaikan pendidikan saat ini.
14. Sahabat dekat penulis, Nur Ridha Tiara Syahputri Lubis yang selalu mendukung dan mendoakan serta sangat banyak membantu penulis hingga berhasil menyelesaikan pendidikan saat ini.
15. Sahabat dekat penulis, Kori Isabella Hutabarat yang selalu mendukung, mendoakan, dan mengingatkan untuk menyelesaikan pendidikan, serta sangat banyak membantu penulis hingga berhasil menyelesaikan pendidikan saat ini.
16. Teman-teman Teknik Informatika Reguler 2018, terima kasih atas persahabatan dan persaudaraannya selama ini. Semoga Allah mudahkan untuk menyelesaikan study S-1 ini.

Penulis menyadari dalam penyusunan skripsi ini, masih terdapat banyak kekurangan dan jauh dari kesempurnaan, oleh karena itu kritik dan saran yang membangun sangat diharapkan untuk kemajuan penelitian selanjutnya. Semoga skripsi ini dapat berguna dan bermanfaat bagi kita semua.

Medan, Februari 2024
Penulis,

Muhammad Yazid Abud Asseweth
NPM 188160054

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Padi (*Oryza Sativa*) adalah tanaman penghasil beras yang merupakan sumber karbohidrat utama dan pangan bagi sebagian besar masyarakat di Indonesia. Stabilitas produksi padi harus tetap dijaga karena permintaan beras terus meningkat setiap tahunnya (Saputra, dkk., 2020). Penurunan produksi padi dapat disebabkan banyak faktor, salah satunya adalah terserang penyakit. Tanaman padi yang terkena penyakit akan menyebabkan kurangnya kuantitas dan kualitas padi yang diproduksi sehingga kemungkinan gagal panen dapat terjadi (Julianto, Sunyoto, & Wibowo, 2022). Penanganan penyakit padi yang tidak sesuai dan kurang cepat dapat menyebabkan terjadinya gagal panen yang berimbas kepada produk padi yang menurun dan penghasilan petani juga berkurang (Anggiratih, dkk., 2021). Penyakit pada tanaman padi dapat diamati berdasarkan perubahan pada batang, akar, daun, dan bagian lainnya (Alwy, dkk., 2023). Daun padi digunakan sebagai tahap awal untuk mengidentifikasi penyakit dikarenakan daun padi yang lebih lebar dari bagian lainnya sehingga gejala penyakit seperti perubahan warna dan bentuk bercak dapat terlihat jelas (Agustiani, dkk., 2022)

Umumnya untuk mengetahui penyakit tanaman padi dapat dilakukan pengamatan secara visual terhadap gejala pada tanaman atau melakukan uji coba di laboratorium (Lu, Tan, & Jiang, 2021). Proses identifikasi penyakit di laboratorium dinilai kurang efektif karena harus menggunakan bahan kimia dan memerlukan waktu yang lama (Julianto, Sunyoto, & Wibowo, 2022). Sementara identifikasi dengan pengamatan secara visual dianggap sering terjadi kesalahan karena

dilakukan secara subjektif (F. Jiang, dkk., 2020). Kedua cara konvensional tersebut memerlukan seorang ahli untuk dapat mengidentifikasi penyakit pada tanaman padi. Karena kurangnya pengetahuan dan tenaga profesional, petani kesulitan dalam mengidentifikasi jenis penyakit yang menyerang tanaman padi (Khoiruddin, Junaidi, & Saputra, 2022).

Penelitian tentang klasifikasi penyakit pada daun tanaman padi telah banyak dilakukan sebelumnya seperti penelitian yang dilakukan oleh (Purnamawati, dkk., 2020) tentang Deteksi Penyakit Daun pada Tanaman Padi Menggunakan Algoritma *Decision Tree, Random Forest, Naïve Bayes*, SVM dan KNN mendapatkan hasil dari setiap perbandingan algoritma terbagi kedalam 3 macam model yaitu model *overfit* yaitu algoritma *Random Forest, Decision Tree* dan *Naïve Bayes,* model *underfit* yaitu (SVM) dan *Good Models* yaitu KNN, metode yang menghasilkan akurasi terbaik adalah KNN. Penelitian yang dilakukan oleh (Shinta, dkk. , 2023) tentang Klasifikasi Penyakit Daun Tanaman Padi Menggunakan CNN dengan Arsitektur VGG-19 menyampaikan bahwa akurasi pengujian model yang menggunakan augmentasi data mendapatkan akurasi yang lebih tinggi dibandingkan tanpa augmentasi data. Akurasi dengan augmentasi data sebesar 94,31%, sedangkan akurasi tertinggi tanpa menggunakan augmentasi data sebesar 93,18%. Penelitian yang dilakukan oleh (Sitompul, Okprana, & Prasetio, 2022) tentang Identifikasi Penyakit Tanaman Padi Melalui Citra Daun Menggunakan *DenseNet201* berhasil memperoleh hasil yang baik dengan nilai akurasi yang didapat pada data training sebesar 92.59% dan 82.99% pada data testing, pada penelitian ini dilakukan identifikasi terhadap 4 jenis citra daun tanaman padi.

Seiring dengan pesatnya perkembangan teknologi khususnya pada bidang kecerdasan buatan memiliki manfaat yang besar. *Deep learning* yang merupakan bagian dari kecerdasan buatan dapat diterapkan pada berbagai jenis pekerjaan seperti pengenalan objek, diagnosis suatu penyakit serta memprediksi peluang dan kejadian (Paraijun, Aziza, & Kuswardani, 2022). Algoritma *deep learning* yang sering digunakan adalah *Convolutional Neural Network* (CNN) (Julianto, Sunyoto, & Wibowo, 2022). CNN sering digunakan dalam pemrosesan citra dikarenakan tingginya akurasi yang dihasilkan dalam pengenalan objek dan citra (Yuliany, Aradea, & Rachman, 2022). Dalam penerapannya, algoritma *Convolutional Neural Network* (CNN) memiliki beragam arsitektur terkenal seperti *VGG-16*, *VGG-19*, *Inception* V1, *Inception* V2, *Inception* V3, *Inception* V4, *ResNet-50*, *ResNet-101*, *AlexNet*, *MobileNet*, *DenseNet-121*, *DenseNet-169*, *DenseNet-201* dan berbagai macam lainnya. Berdasarkan laporan penelitian terdahulu tentang klasifikasi penyakit pada daun tanaman padi belum ditemukan penyelesaian masalah tersebut menggunakan arsitektur *DenseNet-169*.

Beberapa penelitian terdahulu yang menggunakan arsitektur *DenseNet-169* diantaranya adalah penelitian yang dilakukan oleh (Breve, 2022) yaitu *COVID-19 detection on Chest X-ray images: A comparison of CNN architectures and ensembles* memaparkan perbandingan kinerja antara arsitektur *Convolutional Neural Network* dalam mendeteksi penyakit *COVID-19* berdasarkan citra *X-ray,* sebanyakan 21 arsitektur digunakan sabagai perbandingan dalam penelitian ini dengan hasil yang menunjukkan bahwa arsitektur *DenseNet-169* mendapat hasil dengan akurasi tertinggi sebesar 98.15%. Adapun penelitian yang dilakukan oleh (Fahrezantara, Rizal, & Pratiwi, 2022) terkait Pemanfaatan *Convolutional Neural*

Network dalam Klasifikasi Penyakit Tanaman Singkong Menggunakan Arsitektur *Densenet* memaparkan akurasi yang diperoleh dari penggunaan *DenseNet-169* untuk mendeteksi penyakit pada tanaman singkong sebesar 98,73%. Penelitian yang dilakukan oleh (Putri, Sudrajad, & Nastiti, 2022) terkait *Classification of Face Mask Detection Using Transfer Learning Model DenseNet-169* memaparkan tentang bagaimana mengklasifikasikan gambar seseorang apakah menggunakan masker atau tidak, penelitian ini menggunakan arsitektur *DenseNet-169* dan berhasil mendapatkan akurasi sebesar 96%.

Berdasarkan hasil penelitian terdahulu arsitektur *DenseNet-169* menunjukkan kinerja yang baik dan berhasil melewati kinerja dari arsitektur lainnya dalam penyelesaian berbagai macam masalah terkait klasifikasi, oleh karena itu arsitektur *DenseNet-169* dipilih dalam penelitian ini untuk melakukan klasifikasi penyakit pada daun tanaman padi. Penelitian ini diharapkan mampu mengklasifikasikan penyakit pada daun tanaman padi menggunakan arsitektur *DenseNet-169* dengan kinerja yang baik.

1.2 Rumusan Masalah

Adapun rumusan masalah pada penelitian ini untuk menganalisa kinerja arsitektur *DenseNet-169* dalam mengklasifikasi penyakit daun tanaman padi.

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini untuk menunjukkan kinerja arsitektur *DenseNet-169* dalam mengklasifikasikan penyakit daun tanaman padi.

1.4 Manfaat Penelitian

Adapun manfaat dari penelitian ini untuk mengetahui bagaimana kinerja arsitektur *DenseNet-169* dalam mengklasifikasikan penyakit daun tanaman padi.

1.5 Batasan Masalah

Adapun batasan-batasan masalah dari penelitian ini diantaranya adalah:

1. Data citra daun tanaman padi yang digunakan pada penelitian ini berjenis data sekunder yang berasal dari situs repositori data gratis yaitu *Kaggle.*
2. Terdapat 3 jenis citra penyakit daun yang dipakai yaitu: *blight*, *blast*, dan *tungro.*
3. Jumlah data citra daun yaitu 240 gambar dengan pembagian citra jenis *blight* sebanyak 80 citra, jenis *blast* sebanyak 80 citra dan jenis *tungro* sebanyak 80 citra.
4. Arsitektur yang digunakan adalah *DenseNet-169*

BAB II
TINJAUAN TEORI

2.1. Tanaman Padi

Padi atau dengan nama latinnya *Oryza Sativa* adalah salah satu tanaman yang paling banyak ditanam di Indonesia. Padi juga menghasilkan beras yang merupakan salah satu bahan pangan pokok bagi mayoritas masyarakat di Indonesia (Julianto, Sunyoto, & Wibowo, 2022). Kebutuhan akan tanaman padi menjadikan sektor pertanian sebagai aspek yang sangat penting terkait ketahanan pangan. Kebutuhan akan beras yang didapatkan dari padi meningkat seiring dengan bertambahnya jumlah penduduk Indonesia setiap tahunnya (Saputra, dkk., 2020).

2.2. Penyakit Daun Tanaman Padi

Setiap bagian pada tanaman padi memiliki peranan penting, salah satunya adalah daun, daun berfungsi untuk menyerap energi matahari yang digunakan tanaman sebagai bahan untuk fotosintesis. Proses fotosintesis akan terganggu apabila daun pada tanaman dihinggapi oleh sebuah penyakit. Tanaman padi yang telah terkena penyakit akan menyebabkan penurunan kualitas dan kuantitas dari padi yang dihasilkan. Berkurangnya kualitas dan kuantitas dari tanaman padi yang dihasilkan akan mengakibatkan terjadinya gagal panen yang berimbas pada kerugian para petani padi (Julianto, Sunyoto, & Wibowo, 2022). Berikut adalah beberapa penyakit daun padi yang akan diklasifikasikan pada penelitian ini, yaitu:

2.2.1. Penyakit *Blast*

Penyakit *blast* disebabkan oleh jamur *Pyricularia oryzae Cav*. Penyakit ini merupakan salah satu penyakit utama pada tanaman padi di seluruh dunia

(Pamekas, Zahara, & Sinaga, 2023) Penyakit ini telah telah menyebar hampir di seluruh sektor pertanian di Indonesia. Penurunan kualitas hasil yang disebabkan oleh jamur ini bervariasi dari penurunan kualitas ringan hingga 100% tergantung intensitas penyakit yang menyerang tanaman. Beberapa gejala dari penyakit *blast* pada daun tanaman padi meliputi munculnya bercak-bercak pada daun, bentuk dan warna bercak dapat bervariasi tergantung dengan keadaan sekitarnya, umumnya bercak ini berbentuk elips dengan ujungnya runcing layaknya belah ketupat (Kusumawati & Istiqomah, 2020). Bercak yang telah sampai keadaan parah dapat mencapai diameter 1-1,5 cm dan lebar 0,3-0,5 cm dengan tepi berwarna coklat. Penyakit *blast* dapat dilihat pada Gambar 2.1 di bawah.

Gambar 2.1. Penyakit *Blast*

2.2.2. Penyakit *Tungro*

Penyakit *tungro* merupakan salah satu penyakit penting pada tanaman padi yang menjadi hambatan dalam peningkatan produksi padi di Indonesia. Penyakit

ini disebabkan oleh infeksi dua virus, yaitu *Rice tungro bacilliform virus* (RBTV) dan *Rice tungro spherical virus* (RTSV). Gejala yang terlihat dari tanaman padi yang terkena penyakit ini adalah terjadi perubahan warna daun menguning sampai oranye yang disertai dengan bercak-bercak berwarna coklat pada daun, anakan berkurang, kerdil, dan perkembangan akar terhambat (Mbedo & Anasaga, 2020). Penyakit *tungro* dapat dilihat pada Gambar 2.2 di bawah.

Gambar 2.2. Penyakit *Tungro*

2.2.3. Penyakit *Blight*

Penyakit *blight* termasuk salah satu penyakit penting pada tanaman padi di beberapa negara penghasil padi, termasuk Indonesia. Penyakit ini di sebabkan oleh bakteri *Xanthomonas oryzae pv. Orzae*. Peyakit ini dapat menginfeksi daun padi pada seluruh tahapan pertumbuhan tanaman, dimulai dari pesemaian sampai dengan masa panen. Penyakit ini menginfeksi tanaman padi melalui daun dengan melalui luka daun atau melewati lubang alami dari daun yaitu stomata yang akan

merusak klorofil daun sehingga kemampuan tanaman untuk berfotosintesis berkurang (Laraswati, Ramdan, & Kulsum, 2021). Kehilangan hasil padi yang disebabkan oleh penyakit ini berkisar antara 15-80%. Gejala yang terlihat dari penyakit ini adalah bagian yang terinfeksi akan memiliki bercak keabu-abuan dan lama-lama daun akan mengering dan menggulung, gejala ini akan terus menyebar hingga seluruh daun mengering dan terkadang sampai menjadi pelepah (Laraswati, Ramdan, & Kulsum, 2021). Penyakit *blight* dapat dilihat pada Gambar 3.3 di bawah.

Gambar 2.3. Penyakit *Blight*

2.3. *Deep Learning*

Deep Learning adalah subbidang dari *machine learning* yang menggunakan metode *neural network* yang kompleks (Raup, dkk., 2022). *Neural network* adalah model matematis yang menirukan struktur dan fungsi sistem saraf manusia. Terdapat jutaan neuron yang saling terhubung satu sama lain pada otak manusia

untuk mengirimkan dan memproses informasi, *Deep Learning* juga terbuat dari banyak lapisan neuron yang saling terhubung dalam *Deep Learning*, *neural network* terdiri dari lapisan-lapisan yang disebut *"layer"* yang masing-masing memproses input dan menyampaikan hasilnya ke lapisan berikutnya.

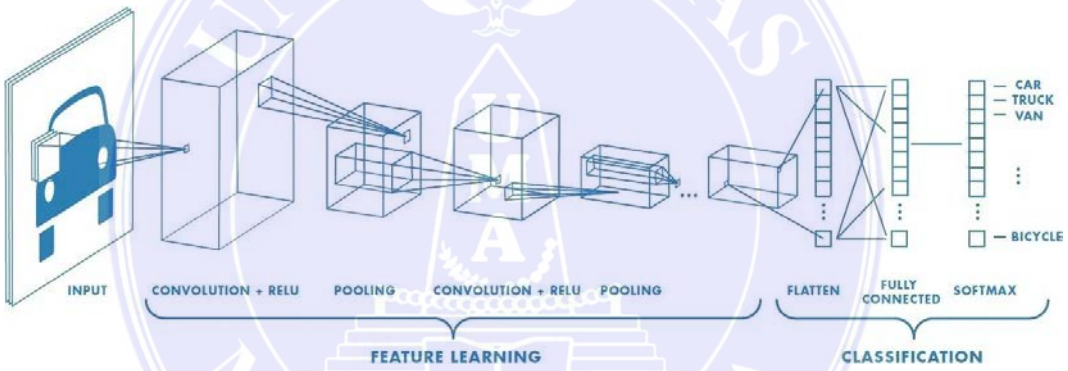
Pemanfaatan dari *Deep Learning* banyak digunakan diberbagai bidang, seperti pada bidang pengolahan citra digital atau *image processing* (Sudalto, 2022). Pengolahan citra digital ini diharapkan dapat membantu manusia untuk mengenali atau mengklasifikasi sebuah objek secara cepat dan tepat. Terdapat banyak algoritma *Deep Learning* yang dapat digunakan untuk melakukan tugas seperti pengenalan object, analisis citra ataupun pembuatan sistem rekomendasi, salah satunya adalah *Convolutional Neural Network* (CNN). Algoritma CNN dianggap mampu memberikan hasil yang signifikan dalam pengenalan citra digital (Nugroho, Fenriana, & Arijanto, 2020).

2.4. *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah jenis algoritma *deep learning* yang merupakan pengembangan dari *multilayer perceptron*. CNN merupakan jenis algoritma yang umum digunakan untuk melakukan klasifikasi ataupun pengenalan object pada data berupa gambar (Kotta, Paseru, & Sumampouw, 2022). Algoritma CNN sering digunakan dalam pengenalan citra karena akurasinya yang tinggi dan sangat baik dalam pengenalan gambar visual (Yuliany, Aradea, & Rachman, 2022). Algoritma CNN dapat menerima input seperti gambar kemudian mengenali objek-objek yang ada pada sebuah gambar, melalui objek-objek tersebut mesin belajar mengenali gambar dan membedakan antara setiap gambar dengan gambar yang lainnya (Iswantoro & Handayani UN, 2022).

Konsep utama dari metode CNN adalah operasi konvolusi yang dilakukan terhadap citra, proses ini akan mengekstraksi setiap fitur pada citra sehingga berdasarkan fitur fitur yang dimiliki oleh sebuah citra metode CNN dapat membedakan dan mengenali satu citra dengan citra yang lainnya. CNN menggunakan arsitektur yang terbentuk dari neuron-neuron yang saling terhubung pada suatu layer (Alamsyah & Pratama, 2020).

Secara umum arsitektur CNN terbagi atas 2 bagian yaitu *feature learning* dan *classification*, setiap bagian terdiri dari banyak lapisan tersembunyi (*hidden layer*). Arsitektur CNN secara umum dapat dilihat pada Gambar 2.4:



Gambar 2.4. Arsitektur *Convolutional Neural Network*
(Tilasefana & Putra, 2023)

Dari Gambar 2.4 di atas dapat dilihat terdapat dua bagian arsitektur, yaitu:

a) *Feature Learning*

Pada *feature learning* ini terjadi proses ekstraksi fitur pada citra, proses ekstraksi ini merupakan perpaduan antara sejumlah *hidden layer* yaitu *convolution layer*, *pooling layer* dan *activation function*.

b) *Classification*

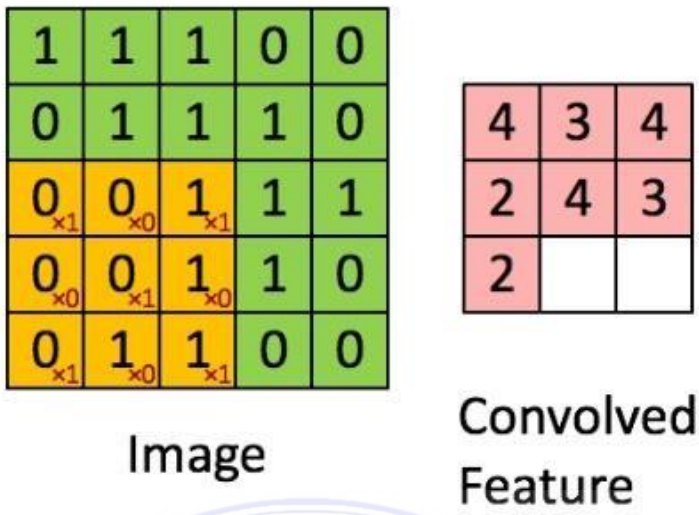
Pada tahapan ini merupakan proses klasifikasi, proses bagian klasifikasi ini terdiri dari *fully connected layer* dan menggunakan

softmax activation function yang keluarannya adalah hasil klasifikasi citra. Pada awal tahapan ini akan dilakukan proses *flatten* yang memiliki arti perataan agar hasil dari operasi konvolusi bisa diproses oleh *fully-connected layer* (Antoko, Ridani, & Minarno, 2021).

2.4.1. *Convolution Layer*

Convolution Layer merupakan lapisan pertama dan juga lapisan yang paling penting pada metode *Convolutional Neural Network* (CNN) (Pratiwi, 2023). Pada lapisan ini akan dilakukan operasi konvolusi pada citra. Operasi konvolusi merupakan proses mengaplikasikan sebuah filter atau *kernel* pada sebuah citra secara bertahap. *Kernel* akan diaplikasikan secara bertahap yang dimulai dari sudut kiri atas dari citra hingga ke kanan bawah. Kernel memiliki ukuran yang beragam, umumnya ukuran kernel yang digunakan adalah 3x3 (Alamsyah & Pratama, 2020).

Tujuan dari dilakukannya operasi konvolusi untuk mengekstraksi fitur dari citra (Nisa, Puspanigrum, & Maulana, 2020). Hasil dari *layer* ini adalah matriks yang berisi nilai hasil konvolusi antar *kernel* yang telah diaplikasikan pada gambar atau biasa disebut dengan *feature map.* Nilai hasil konvolusi diperoleh dengan melakukan operasi *dot* antara nilai input piksel dari citra dengan nilai dari sebuah filter. Operasi konvolusi ini menyebabkan perubahan pada beberapa bagian gambar sehingga bagian yang kurang penting pada gambar akan hilang dan bagian yang penting akan tampak jelas (Fahrezantara, Rizal, & Pratiwi, 2022). Proses dari operasi konvolusi dapat dilihat pada Gambar 2.5 sebagai berikut.



Gambar 2.5. Operasi Konvolusi
(Paliwang, 2020)

2.4.2. *Rectified Linear Unit Activation Function*

Matriks nilai hasil dari operasi konvolusi akan dimasukkan kedalam fungsi *aktivasi Rectified Linear Unit* (ReLU). Beberapa fungsi aktivasi yang sering digunakan adalah ReLU, Sigmoid dan Tanh, namun fungsi aktivasi ReLU sering menjadi pilihan dikarenakan memberikan hasil yang lebih baik dibandingkan fungsi aktivasi lainnya (Antoko, Ridani, & Minarno, 2021).

Setiap nilai yang dimasukkan kedalam fungsi aktivatsi ReLU akan menghasilkan 0 apabila inputnya bernilai negatif. Jika nilai inputan bernilai positif maka fungsi ReLU akan mengembalikan nilai inputan itu sendiri. Fungsi aktivasi ReLU secara matematis adalah sebagai berikut:

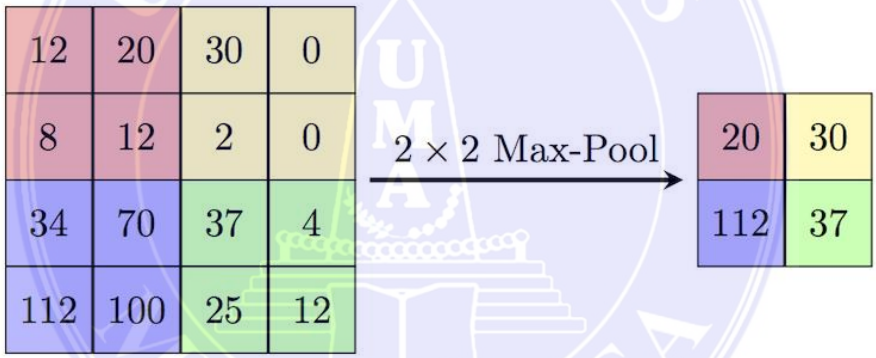
$$\boldsymbol{f(x) = \max(0, x)} \quad \text{.....(2.1)}$$

2.4.3. *Pooling Layer*

Pooling layer merupakan layer setelah *convolutional layer*. Tujuan utama dari layer ini adalah melakukan *down-sampling* yaitu mengurangi dimensi dari

feature map hasil dari operasi konvolusi. Dengan berkurangnya dimensi dari *feature map* maka proses komputasi pada tahapan *classification* akan berjalan lebih cepat (Antoko, Ridani, & Minarno, 2021).

Terdapat berbagai macam filter yang digunakan pada *pooling layer*, umumnya filter yang digunakan adalah filter 2x2. Menggunakan filter berukuran 2x2 dapat mengurangi ukuran *feature maps* hingga 75% dari ukuran aslinya. Proses pooling dapat dilakukan dengan banyak cara, diantaranya adalah *max pooling* yaitu dengan mengambil nilai maksimum dari sub citra dan *average pooling* yaiut dengan mengambil nilai rata-rata dari sub citra (Alamsyah & Pratama, 2020). Proses dari max pooling dapat dilihat pada Gambar 2.6 sebagai berikut.



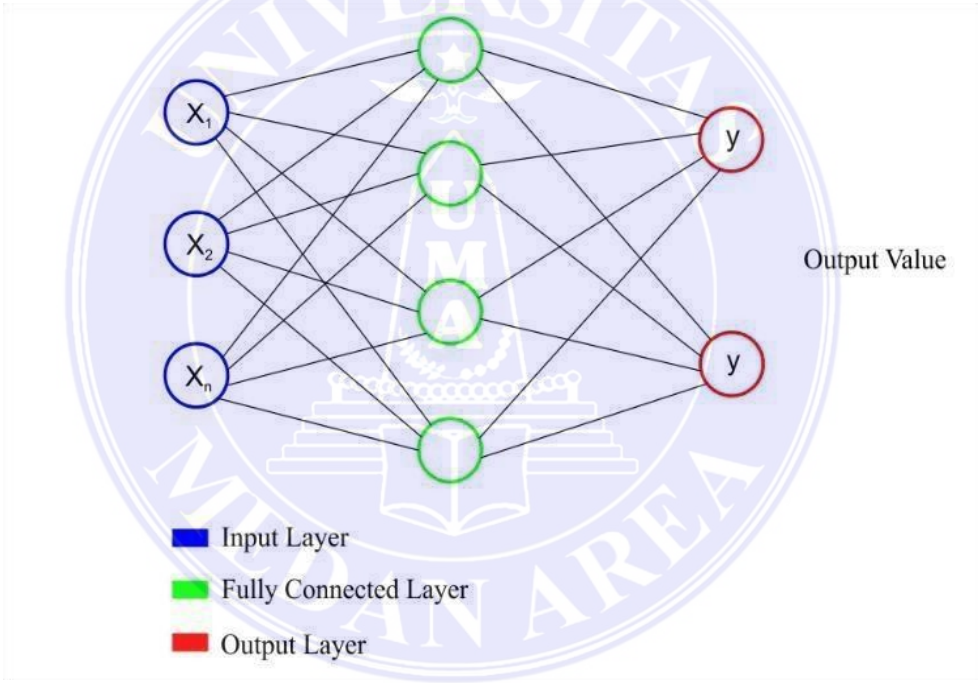
Gambar 2.6. *Max Pooling*
(Antoko, Ridani, & Minarno, 2021)

2.4.4. *Fully Connected Layer*

Feature map yang didapatkan dari operasi konvolusi dan *pooling* masih berbentuk *multidimensional array* sementara *fully connected layer* hanya dapat menerima inputan berupa *singledimensional array*, oleh sebab itu *feature map* harus melewati proses *flatten* untuk mengubah *feature map* menjadi satu dimensi agar dapat digunakan sebagai input dari *fully connected layer* (Bowo, Syahputra, &

Akbar, 2020). *Flattening* adalah teknik yang digunakan untuk merubah vektor yang memiliki ukuran dua dimensi menjadi vektor berukuran satu dimensi (Wicaksono, Andryana, & Benrahman, 2020).

Fully Connected Layer adalah lapisan dimana setiap neuron yang berada pada lapisan sebelumnya terhubung secara keseluruhan dengan lapisan neuron selanjutnya. Nilai-nilai matriks yang sebelumnya telah diperoleh pada tahap *feature learning* akan menjadi inputan pada lapisan ini. Struktur *fully connected layer* dapat dilihat pada Gambar 2.7 sebagai berikut.



Gambar 2.7. *Fully Connected Layer*
(Adithama, Maslim, & Gunawan, 2023)

2.4.5. *Softmax Activation Function*

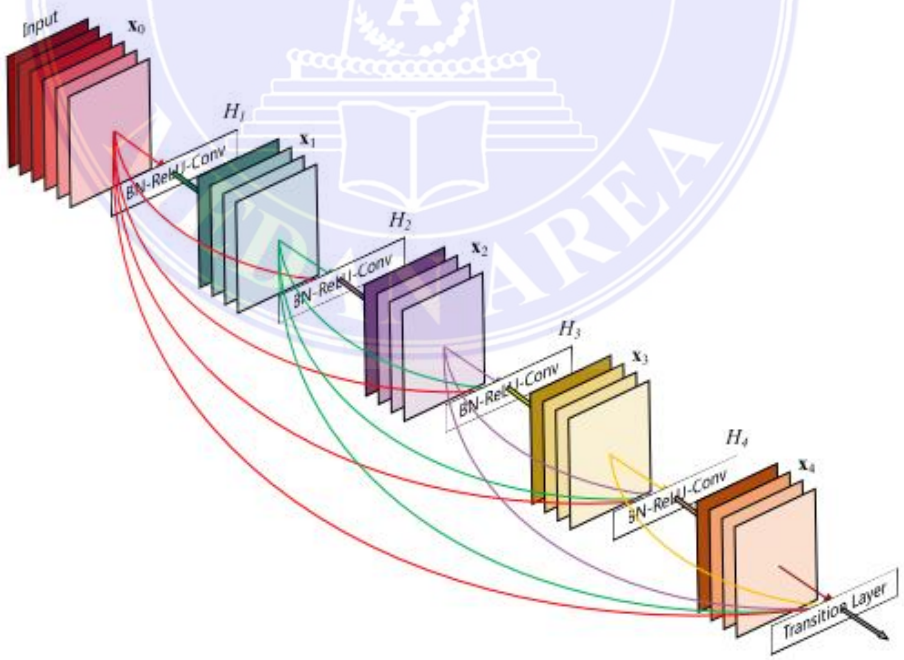
Softmax merupakan fungsi aktivasi yang digunakan untuk mengubah nilai perhitungan klasifikasi menjadi nilai *probability* sehingga nilai perhitungan bisa saling dibandingkan (Dzaky, 2021). Softmax akan melakukan perhitungan

probabilitas atau kemungkinan dari setiap kelas target dari inputan citra yang diberikan. Kelas target yang menghasilkan nilai probabilitas tertinggi menjadi kelas hasil klasifikasi dari input citra yang diberikan. Softmax akan memberikan nilai probabilitas dengan rentang nilai 0 hingga 1 dengan jumlah keseluruhan nilai probabilitas = 1 (Sholawati, Auliasari, & Ariwibisono, 2022)

2.5. *DenseNet-169*

Dense Convolutional Network atau DenseNet awalnya diperkenalkan pada tahun 2017 oleh Gao Huang, Zhuang Liu, Laurens van Der Maaten, dan Kilian Q. Weinberger di konferensi CVPR (*Computer Vision and Pattern Recognition*) pada tahun 2017 dan berhasil mendapatkan penghargaan *Best Paper Award*. DenseNet bekerja dengan cara menghubungkan satu *convolution layer* dengan semua *convolution layer* setelahnya yang menjadikan semua layer dapat mengambil input berupa *feature map* dari semua layer sebelumnya (Fahrezatanra, Rizal, & Pratiwi, 2022). DenseNet memiliki beberapa kelebihan diantaranya yaitu mengurangi *vanishing-gradient problem,* memperkuat *feature propagation,* dan mengurangi jumlah parameter yang digunakan (Putra & Bunyamin, 2020). *Vanishing gradient* adalah keadaan ketika nilai *gradient* yang digunakan untuk memperbarui bobot menyusut seiring waktu sehingga nilai *gradient* bernilai 0 atau mendekati 0 (Syahram, Effendy, & Setyawan, 2021). Feature propagation adalah proses penerusan informasi berupa feature yang telah didapat dari satu lapisan ke lapisan-lapisan berikutnya sebuah jaringan neural networks. DenseNet memiliki beberapa varian yaitu 121, 169, dan 201 dimana angka-angka ini mengartikan kedalaman dari arsitektur DenseNet.

Secara umum arsitektur DenseNet terdiri dari 3 bagian yaitu *dense block layer*, *transition layer*, dan *classification layer* (Ramdan, dkk., 2020). *Dense block* terdiri dari *layer batch normalization*, fungsi aktivasi ReLU, dan *convolutional layer* yang berukuran 1x1 dan 3x3 dengan jumlah yang berbeda-beda pada setiap blok. Sementara *transition layer* terdiri dari *convolutional layer* 1x1 *dan average pooling layer* 2x2. Pada *transition layer* ini dilakukan pengurangan dimensi pada inputan untuk mempercepat proses komputasi*, pooling layer* yang digunakan pada *transition layer* adalah average pooling yang artinya nilai yang diambil pada proses pooling adalah nilai rata-rata dari hasil *convolution layer*. *Classification layer* merupakan lapisan terakhir dari arsitektur DenseNet, lapisan ini terdiri dari *global average layer* dan *fully connected layer* dengan fungsi aktivasi softmax. Arsitektur DenseNet dapat dilihat pada Gambar 2.8 sebagai berikut.



Gambar 2.8. Arsitektur DenseNet
(Putra & Bunyamin, 2020)

2.6. Simulasi Perhitungan dan Cara Kerja Algoritma CNN

Tahapan dari algoritma *Convolutional Neural Network* (CNN) dalam melakukan klasifikasi citra melewati 2 proses, yaitu proses *feature learning* dan proses klasifikasi. CNN pada mulanya akan menerima inputan sebuah citra untuk diklasifikasikan, kemudian citra tersebut akan melewati proses *feature learning* dan proses klasifikasi. Pada tahapan *feature learning* akan dilakukan proses ekstraksi fitur. Pada tahapan klasifikasi, nilai dari fitur-fitur citra yang telah diekstrak sebelumnya untuk mendapatkan nilai probabilitas untuk setiap neuron pada lapisan *output* menggunakan fungsi aktivasi softmax. Neuron dengan nilai probabilitas tertinggi akan menjadi kelas target dari citra yang diinputkan pada CNN.

2.6.1. *Convolution Layer*

Setelah CNN menerima input image kemudian operasi konvolusi akan dilakukan dengan cara mengaplikasikan sebuah filter atau *kernel* pada citra secara bertahap dimulai dari kiri atas sampai dengan kanan bawah dari citra. Umumnya ukuran kernel yang digunakan adalah 3x3. Tujuan dilakukannya operasi konvolusi untuk mengekstraksi fitur-fitur dari sebuah citra. Berikut adalah contoh *input image* yang akan digunakan dan contoh *kernel* yang akan digunakan untuk menunjukkan proses *feature learning* dan klasifikasi dari algoritma CNN.

101	88	72	69	77
107	95	78	73	79
107	97	80	73	77
111	102	82	73	74
125	117	94	78	74

Sampel Pixel Citra

Input Image

Gambar 2.9. Sampel Data Input

1	0	1
0	1	0
1	0	1

Kernel 3x3

Pada pembahasan cara kerja CNN akan digunakan sebagian kecil dari *input image* yaitu sebesar 5 x 5 yang dapat dilihat pada Gambar 2.9. Citra umumnya terdiri atas nilai-nilai piksel yang mewakili warna dari citra tersebut. Nilai-nilai piksel dari sampel *input image* dapat dilihat pada Gambar 2.9. Setelah menerima *input image*, CNN akan melakukan operasi konvolusi menggunakan *kernel*. Adapun operasi konvolusi dapat dilihat pada Gambar 2.10.

101	88	72	69	77
107	95	78	73	79
107	97	80	73	77
111	102	82	73	74
125	117	94	78	74

×

1	0	1
0	1	0
1	0	1

Gambar 2.10. Operasi Konvolusi Menggunakan Kernel 3x3

Kernel akan bergerak dari kiri atas dan akan berpindah sebanyak 1 piksel ke kanan hingga baris pertama selesai diproses kemudian dilanjutkan dengan turun pada baris berikutnya, setiap pergeseran kernel akan dilakukan operasi konvolusi. Konvolusi merupakan penjumlahan dari perkalian setiap nilai pada *kernel* dengan setiap piksel pada citra masukan. Hasil dari operasi konvolusi ini disebut dengan *feature map.* Berikut adalah hasil dari operasi konvolusi.

101	88	72	69	77
107	95	78	73	79
107	97	80	73	77
111	102	82	73	74
125	117	94	78	74

×

1	0	1
0	1	0
1	0	1

=

455		

Gambar 2.11. Operasi Konvolusi 1

(101*1) + (88*0) + (72*1) + (107*0) + (95*1) + (78*0) + (107*1) + (97*0) + (80*1) = 455

101	88	72	69	77
107	95	78	73	79
107	97	80	73	77
111	102	82	73	74
125	117	94	78	74

×

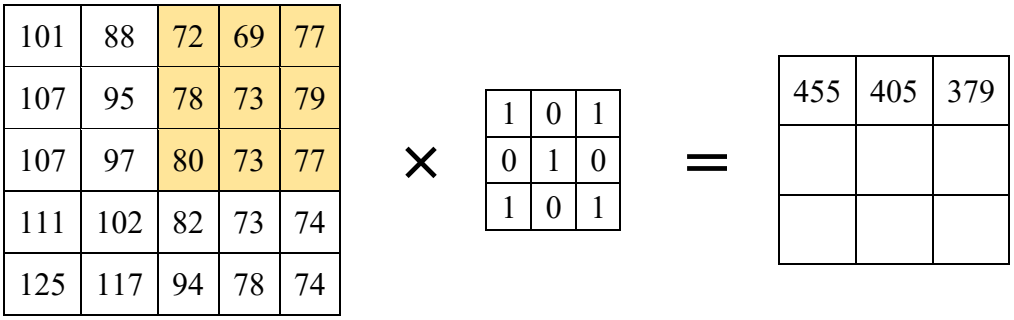
1	0	1
0	1	0
1	0	1

=

455	405	

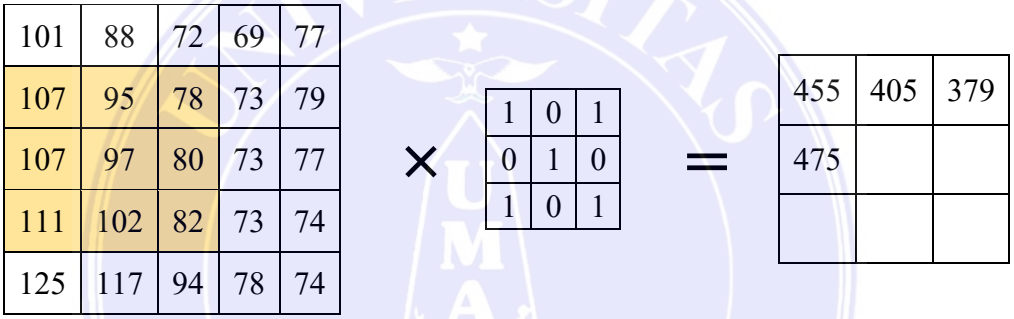
Gambar 2.12. Operasi Konvolusi 2

(88*1) + (72*0) + (69*1) + (95*0) + (78*1) + (73*0) + (97*1) + (80*0) + (73*1) = 405



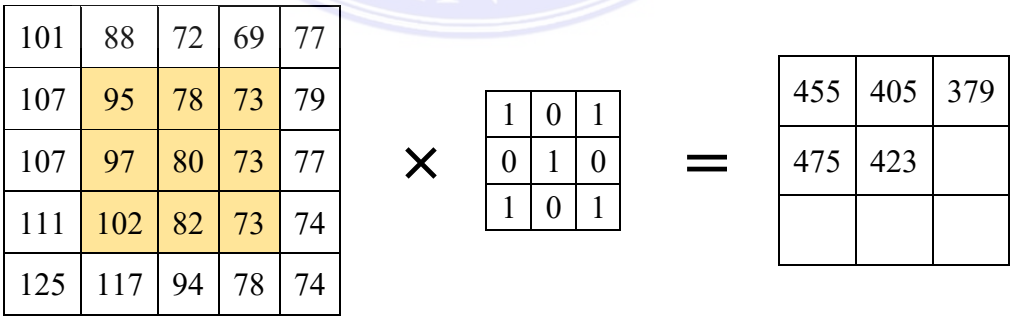
Gambar 2.13. Operasi Konvolusi 3

(72*1) + (69*0) + (77*1) + (78*0) + (73*1) + (79*0) + (80*1) + (73*0) + (77*1) = 379



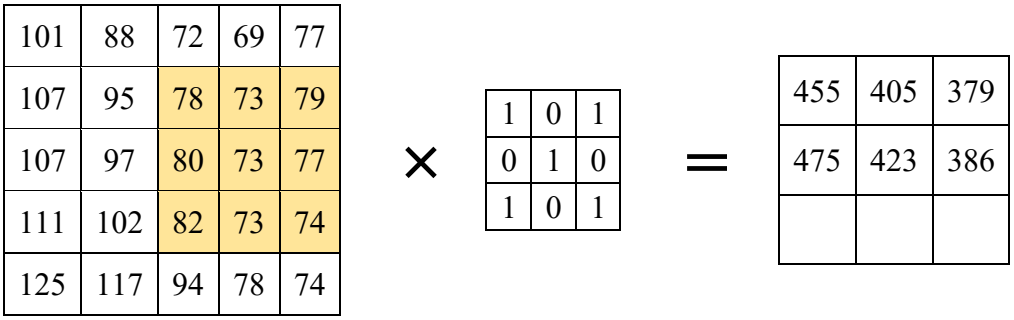
Gambar 2.14. Operasi Konvolusi 4

(107*1) + (95*0) + (78*1) + (107*0) + (97*1) + (80*0) + (111*1) + (102*0) + (82*1) = 475



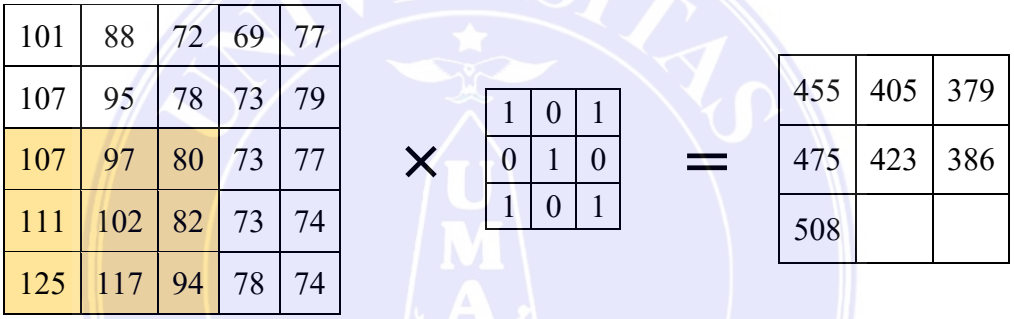
Gambar 2.15. Operasi Konvolusi 5

(95*1) + (78*0) + (73*1) + (97*0) + (80*1) + (73*0) + (102*1) + (82*0) + (73*1) = 423



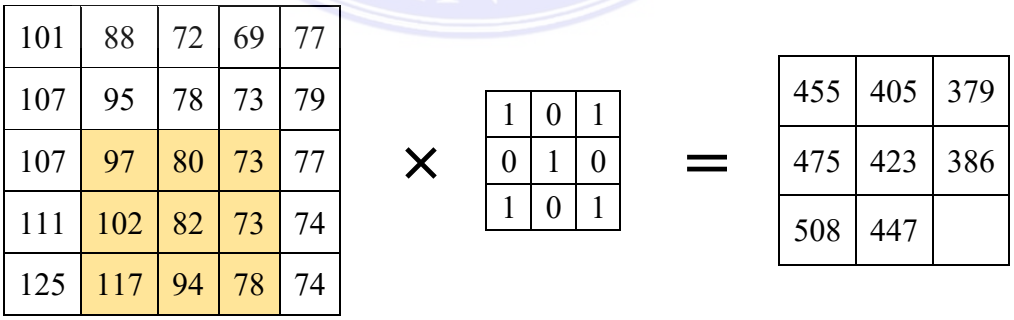
Gambar 2.16. Operasi Konvolusi 6

(78*1) + (73*0) + (79*1) + (80*0) + (73*1) + (77*0) + (82*1) + (73*0) + (74*1) = 386



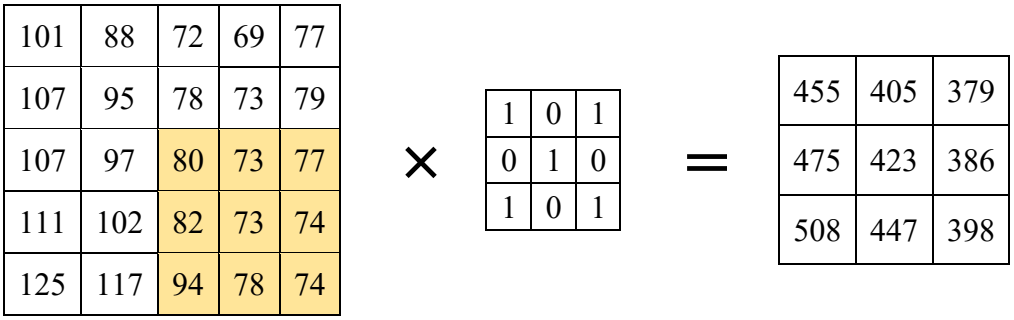
Gambar 2.17. Operasi Konvolusi 7

(107*1) + (97*0) + (80*1) + (111*0) + (102*1) + (82*0) + (125*1) + (117*0) + (94*1) = 508



Gambar 2.18. Operasi Konvolusi 8

(97*1) + (80*0) + (73*1) + (102*0) + (82*1) + (73*0) + (117*1) + (94*0) + (78*1) = 447



Gambar 2.19. Operasi Konvolusi 9

(80*1) + (73*0) + (77*1) + (82*0) + (73*1) + (74*0) + (94*1) + (78*0) + (74*1) = 398

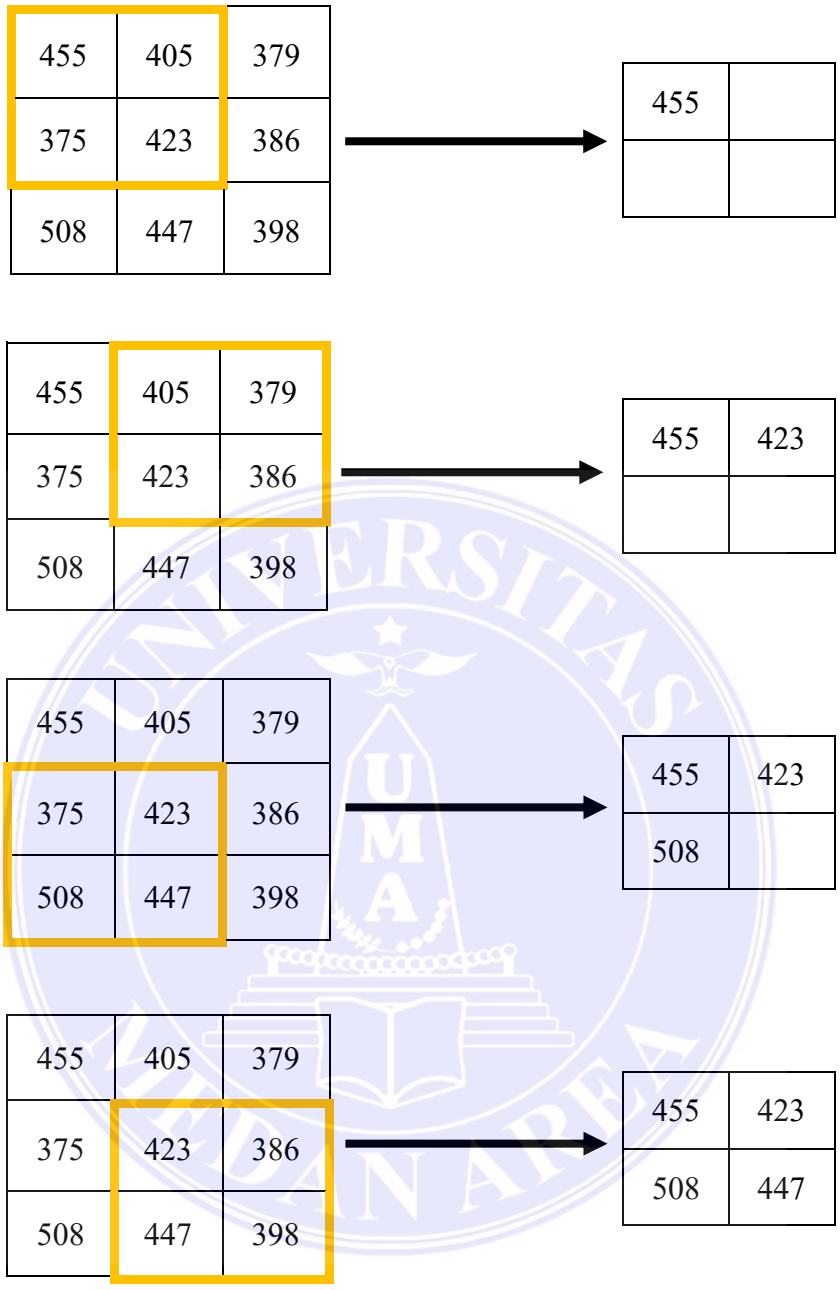
Hasil akhir dari operasi konvolusi dari *input image* terlihat pada Gambar 2.20.

455	405	379
375	423	386
508	447	398

Gambar 2.20. Hasil Operasi Konvolusi

2.6.2. *Max Pooling Layer*

Setelah operasi konvolusi selesai dilakukan pada *convolution layer*, selanjutnya akan dilakukan proses *max pooling* terhadap hasil dari operasi konvolusi sebelumnya. Ukuran *pooling* yang digunakan adalah 2x2. Tujuan *max pooling* untuk mengurangi dimensi dari *feature map* hasil operasi konvolusi. Tahapan dari proses *max pooling* dapat dilihat pada Gambar 2.21.



Gambar 2.21. Proses *Max Pooling*

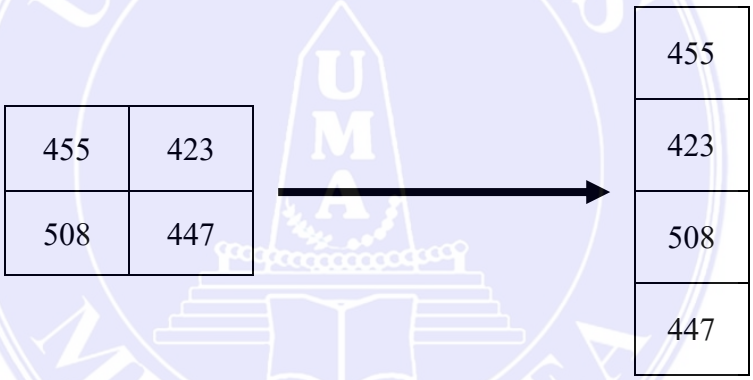
Kernel max pooling akan bergeser dari kiri ke kanan dan mengambil nilai maksimal dari *feature maps* kemudian menyimpannya pada matrik baru. Setelah proses *max pooling* dilakukan maka akan tercipta matrik baru berisi nilai dari proses *max pooling.* Hasil dari proses *max pooling* dapat dilihat pada Gambar 2.22.

455	423
508	447

Gambar 2.22. Hasil *Max Pooling*

2.6.3. *Flatten Layer*

Hasil dari proses *max pooling* masih berbentuk dua dimensi sementara *fully connected layer* membutuhkan inputan berupa 1 dimensi. *Flatten layer* berfungsi untuk mengubah matriks ukuran 2 dimensi menjadi matriks berukuran 1 dimensi. Hasil dari *flatten layer* dapat dilihat pada Gambar 2.23.

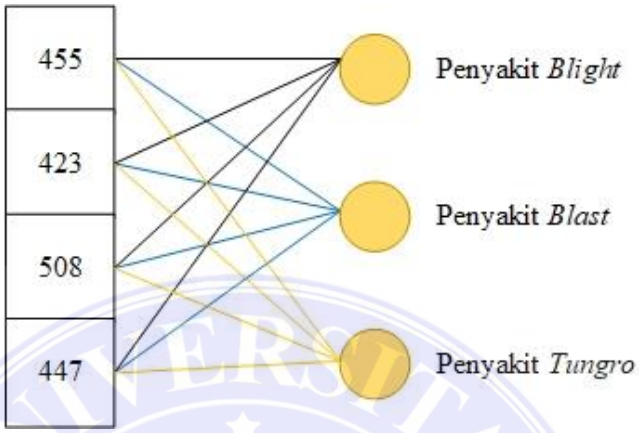


Gambar 2.23. Vektor Hasil *Flattening*

2.6.4. *Fully Connected Layer*

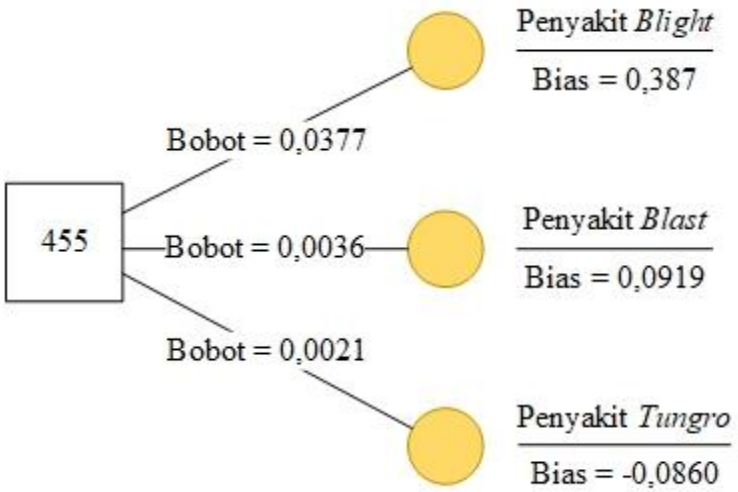
Setelah matriks berhasil menjadi 1 dimensi selanjutnya matriks tersebut akan menjadi inputan dari *fully connected layer. Fully connected layer* adalah lapisan dimana setiap neuron pada lapisan sebelumnya terhubung secara keseluruhan dengan lapisan neuron selanjutnya. Lapisan ini merupakan *hidden layer*, setiap model memiliki minimal 1 *hidden layer* pada *fully connected layer* dimana *layer* ini terdiri dari neuron-neuron yang mewakili masing-masing

kemungkinan kelas atau kelompok yang ada dan lapisan ini akan menentukan kelas dari citra yang diberikan untuk diklasifikasikan. Bentuk dari *fully connected layer* dapat dilihat pada Gambar 2.24.



Gambar 2.24. *Fully Connected Layer*

Pada lapisan ini akan dilakukan perhitungan terhadap nilai piksel dari vektor citra dengan melibatkan bobot dan bias. Setiap jaringan yang terhubung dari sebuah nilai ke neuron pada lapisan selanjutnya memiliki sebuah bobot dan setiap neuron output dari setiap kelas mempunyai bias. Nilai piksel akan dikalikan dengan setiap bobot yang ada, selanjutnya akan dilakukan penjumlahan terhadap seluruh nilai hasil perkalian sebelumnya, kemudian nilai hasil penjumlahan akan ditambahkan dengan nilai bobot, nilai inilah yang kemudian akan dihitung menggunakan *softmax function* untuk mendapatkan nilai probabilitas dengan rentang 0-1. Contoh nilai bobot dan bias pada 1 neuron yang terhubung ke 3 *hidden layer* dapat dilihat pada Gambar 2.25.

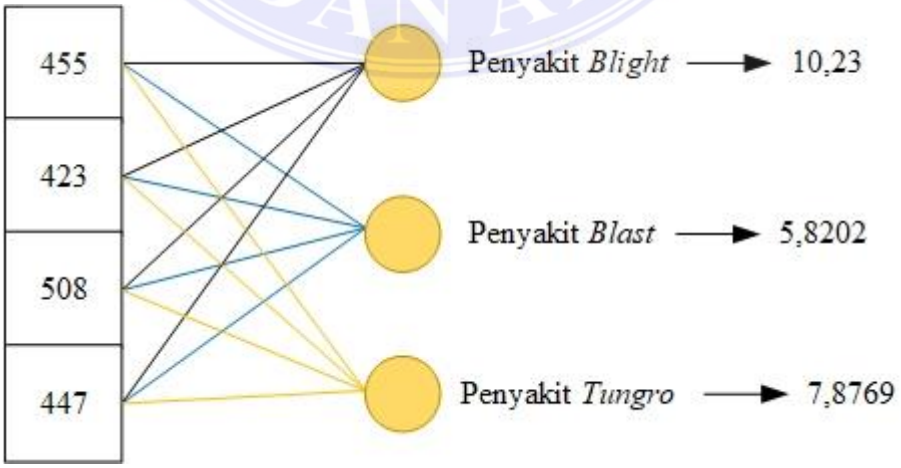


Gambar 2.25. Contoh Bobot dan Bias

Hasil perhitungan dari nilai piksel, bobot, dan bias untuk setiap neuron *output* dapat dilihat sebagai berikut:

- Neuron *Blight* = 455 * 0,0377 + 0,387 = 17,5405
- Neuron *Blast* = 455 * 0,0036 + 0,0919 = 1,7299
- Neuron *Tungro* = 455 * 0,0021 + -0,0860 = 0,8695

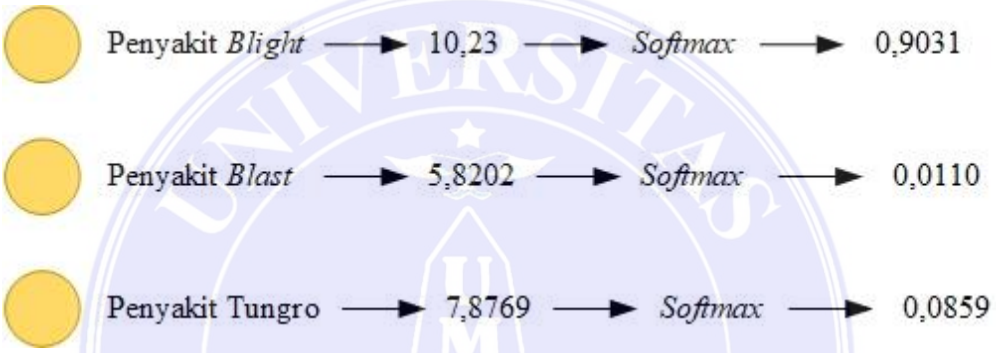
Hasil dari perhitungan setiap vektor piksel terhadap bobot dan bias dapat dilihat pada Gambar 2.26.



Gambar 2.26. Hasil Perhitungan *Fully Connected Layer*

2.6.5. Klasifikasi menggunakan *Softmax Function*

Hasil dari perhitungan neuron-neuron yang terhubung pada *fully connected layer* selanjutnya akan dimasukkan ke dalam *softmax function* sehingga menghasilkan nilai probabilitas dari setiap neuron output pada *hidden layer* terakhir, neuron output yang memiliki nilai probabilitas tertinggi menjadi kelas hasil klasifikasi dari citra yang telah diinputkan. Hasil dari perhitungan *softmax function* dapat dilihat pada Gambar 2.27.



Gambar 2.27. Hasil *Softmax Function*

2.7. Confusion Matrix

Salah satu tahapan penting adalah pengukuran kinerja dari suatu sistem klasifikasi. Kinerja dari sistem klasifikasi dapat diketahui dari hasil sistem ketika mengklasifikasikan data. Confusion Matrix dapat digunakan sebagai metode untuk mengukur kinerja atau performa dari suatu metode klasifikasi (Wahid, Mustamin, & Lawi, 2021). Confusion matrix memiliki informasi yang membandingkan hasil klasifikasi yang dilakukan sistem dengan hasil klasifikasi seharusnya.

Confusion matrix memiliki tabel dengan empat istilah sebagai representasi hasil proses klarifikasi. Istilah pada confusion matrix merupakan gabungan berbeda dari nilai prediksi dan nilai aktual (Agustin, Eviyanti, & Azizah, 2023). Keempat istilah tersebut diantaranya adalah:

- *True Positive* (TP), terjadi jika model memprediksi data berada di kelas positif dan data memang berada di kelas positif.
- *True Negative* (TN), terjadi jika model memprediksi data berada di kelas negatif dan data memang berada di kelas negatif.
- *False Positive* (FP), terjadi jika model memprediksi berada di kelas positif dan ternyata data berada di kelas negatif.
- *False Negative* (FN), terjadi jika model memprediksi data berada di kelas negatif dan ternyata data berada di kelas positif.

Tabel 2.1 Matrix Confusion

		Prediksi	
		Positif	Negatif
Aktual	Positif	*True Positive (TP)*	*False Negative (FN)*
	Negatif	*False Positive (FP)*	*True Negative (TN)*

2.8. Penelitian Terdahulu

Penelitian terdahulu memaparkan tentang penelitian yang telah dilakukan oleh peneliti lain tentang kasus-kasus yang berkaitan dengan penelitian saat ini. Penelitian terdahulu menjadi bahan referensi dan kajian dalam melakukan penelitian ini. Berikut adalah penelitian terdahulu yang berkaitan dengan klasifikasi penyakit pada daun tanaman dan arsitektur DenseNet-169 serta menjadi acuan dalam melakukan penelitian ini:

Tabel 2.2 Penelitian Terdahulu

No	Judul	Peneliti & Tahun	Keterangan
1	Deteksi Penyakit Daun pada Tanaman Padi Menggunakan Algoritma *Decision Tree*, *Random Forest*, *Naïve Bayes*, SVM dan KNN	(Purnamawati, Nugroho, Putri, & Hidayat, 2020)	Deteksi klasifikasi pada hama daun padi menggunakan dataset yang berasal dari UCI *Machine Learning Repository* dengan dataset *Rice Leaf Diseases* dengan tiga kelas yaitu *Bacterial leaf Blight*, *Brown* spot, dan *Leaf smut* dengan masing-masing kelas memiliki 40 gambar dengan format gambar jpg. Pengujian dilakukan dengan 5 algoritma yang berbeda yaitu *Decision Tree*, *Random Forest*, *Naïve Bayes*, SVM, KNN. Dari perbandingan kelima algoritma tersebut ditemukan KNN menjadi metode yang terbaik dengan nilai akurasi 87%.
2	Klasifikasi Citra Penyakit Daun Tanaman Padi Menggunakan CNN dengan Arsitektur *VGG-19*	(Shinta, Irsyad, Yanto, Sanjaya, 2023)	Menggunakan data penyakit daun padi yang diperoleh dari *Kaggle Repository* dan terbagi atas 4 jenis kelas penyakit daun padi yaitu *blast, brown spot, leaf smut,* dan *healthy* sebanyak 440 citra. Augmentasi citra dilakukan terhadap dataset sehingga mendapatkan total citra hasil augmentasi sebanyak 1320 citra. Resize dilakukan untuk mengubah ukuran citra yang diperkecil menjadi 224 x 224 piksel Penelitian ini membandingan hasil akurasi pengujian dari model yang menggunakan augmentasi data dan tanpa augmentasi data. Hasil pengujian

			menunjukkan akurasi tertinggi dengan menggunakan augmentasi data adalah 94.31%, sedangkan akurasi tertinggi tanpa augmentasi data yang diperoleh adalah 93.18%
3	Identifikasi Penyakit Tanaman Padi Melalui Citra Daun Menggunakan *DenseNet 201*	(Sitompul, Okprana, & Prasetio, 2022)	Menggunakan data penyakit daun padi yang diperoleh dari *Kaggle Repository* yang terbagi atas 4 jenis penyakit yaitu *healthy* sebanyak 1,488 citra, *brown spot* sebanyak 523 citra, *hispa* sebanyak 565 citra dan *leaf blast* sebanyak 779 citra dengan format gambar jpg. Ukuran citra kemudian direduksi menjadi 224*224 piksel, dataset dibagi menjadi tiga bagian yaitu data training, data validasi dan data testing dengan rasio 70% untuk training, 20% untuk testing dan 10% untuk validasi. Augmentasi selanjutnya dilakukan untuk memperbanyak citra. Hasil akurasi sebesar 92.59% diperoleh untuk data training dan 82.99% diperoleh pada data testing.
4	*Classification of Face Mask Detection Using Transfer Learning Model DenseNet-169*	(Putri, Sudrajad, & Nastiti, 2022)	Penelitian ini melakukan klasifikasi apakah seseorang menggunakan masker atau tidak. Dataset yang digunakan berasal dari situs *Kaggle* dan *GitHub*. Dataset terbagi atas 2 kelas yaitu citra seseorang menggunakan masker dan tidak menggunakan masker. Dataset yang berasal dari *Kaggle* berjumlah 1376 citra dan dataset yang berasal dari *GitHub*

			berjumlah 7553 citra. Pembagian data dilakukan dengan rasio 80% untuk data training, 10% untuk data validasi dan 10% untuk data testing. Beberapa tahapan augmentasi citra dilakukan yaitu diantaranya *rescale*, *rotation*, *width shift*, *height shift*, *shear*, *zoom*, *horizontal flip*, dan *fill mode.* Dataset kemudian dilatih menggunakan arsitektur DenseNet-169 dan berhasil mendapatkan akurasi sebesar 96%.
5	Pemanfaatan *Convolutional Neural Network* Dalam Klasifikasi Penyakit Tanaman Singkong Menggunakan Arsitektur DenseNet	(Fahrezantara, Rizal, & Pratiwi, 2022)	Klasifikasi penyakit pada singkong menggunakan dataset *Cassava Disease Classification* yang berasal dari *Tensorflow Datasets*. Dataset terdiri atas 9430 gambar dengan pembagian data latih sebanyak 80% dari total dataset. Terdapat lima kelas pada dataset yang akan diklasifikasikan yaitu *Cassava Brown Streak Disease* (CBSD), *Cassava Mosaic Disease* (CMD), *Cassava Bacterial Bligh* (CBB), *Cassava Green Mite* (CGM) dan daun sehat. Skenario pengujian dilakukan menggunakan arsitektur DenseNet-169 dengan nilai *learning rate* 0,01; 0,001; dan 0,0001. Optimizer yang digunakan untuk masing-masing *learning rate* adalah Adam, RMSProp, dan SGD. Hasil penelitian menunjukkan *learning rate* 0,0001 dengan optimizer RMSProp menghasilkan akurasi tertinggi yaitu

			akurasi training sebesar 98,12% dan akurasi validasi sebesar 78,88%.
6	*COVID-19 detection on Chest X-ray images: A comparison of CNN architectures and ensembles*	(Breve, 2022)	Penelitian ini memaparkan klasifikasi terhadap citra x-ray apakah seseorang memiliki penyakit *COVID-19* atau tidak. Dataset yang digunakan adalah COVIDX8B dengan jumlah data sebanyak 16,352 data. Pengujian dilakukan menggunakan 21 arsitektur CNN yang berbeda untuk mengetahui arsitektur yang terbaik. 21 arsitektur tersebut diantarany adalah DenseNet-169, EfficientNetB2, InceptionResNetV2, InceptionV3, MobileNet, EfficientNetB0, EfficientNetb3, DenseNet201, ResNet152V2, ResNet152, DenseNet121, Xception, VGG19, EfficientNetB1, ResNet50, VGG16, ResNet101V2, MobileNetV2, ResNet101, DenseNet-169 Mendapatkan hasil tertinggi dengan nilai akurasi sebesar 98.15%

BAB III
METODOLOGI PENELITIAN

3.1. Alat dan Bahan Penelitian

Pada penelitian ini dibutuhkan peralatan dan bahan pendukung agar penelitian dapat berjalan dengan baik. Peralatan dan bahan terbagi atas perangkat keras dan perangkat lunak. Perangkat keras dan perangkat lunak yang digunakan diantaranya adalah:

3.1.1 Perangkat Keras

Tabel 3.1 Perangkat Keras

No	Perangkat Keras	Deskripsi
1	Device	Laptop Acer E5-552G
2	Processor	AMD A10-8700P @1.80 GHz
3	SSD	120 GB
4	RAM	12 GB

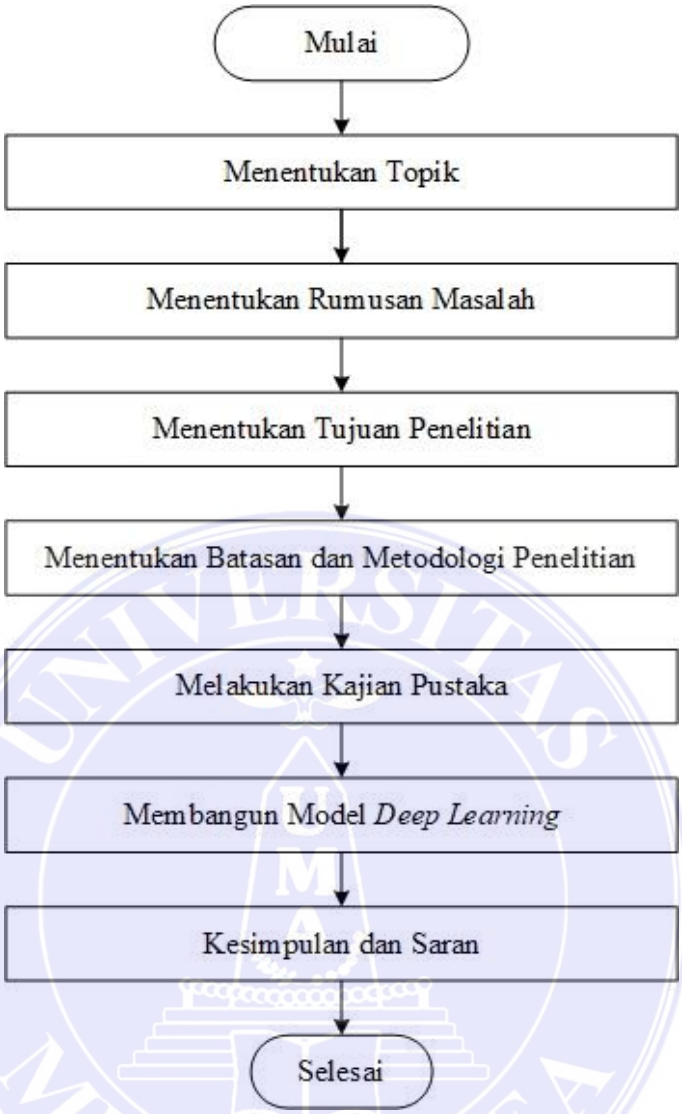
3.1.2 Perangkat Lunak

Tabel 3.2 Perangkat Lunak

No	Perangkat Lunak	Deskripsi
1	Windows 10 Pro 64-bit	Sistem Operasi
2	Google Colab	– Disk: 107 GB – RAM: 12 GB
3	Python	Bahasa Pemrograman
4	TensorFlow	Library

3.2. Alur Metodologi Penelitian

Secara keseluruhan alur dari penelitian ini dimulai dengan menentukan topik pembahasan penelitian kemudian menentukan permasalahan topik tersebut, dilanjutkan dengan menentukan tujuan dari penelitian serta menentukan batasan masalah untuk mambatasi ruang lingkup penelitian, selanjutnya menentukan metodologi penelitian yang akan digunakan dalam menyelesaikan permasalahan penelitian. Selanjutnya dilakukan kajian pustaka untuk mengetahui teori-teori terhadap objek-objek penelitian. Dilanjutkan dengan pembuatan model *deep learning* menggunakan arsitektur *DenseNet-169*. Diakhiri dengan kesimpulan dan saran atas penelitian yang telah dilakukan. Diagram alur metodologi penelitian ini dapat dilihat pada Gambar 3.1.

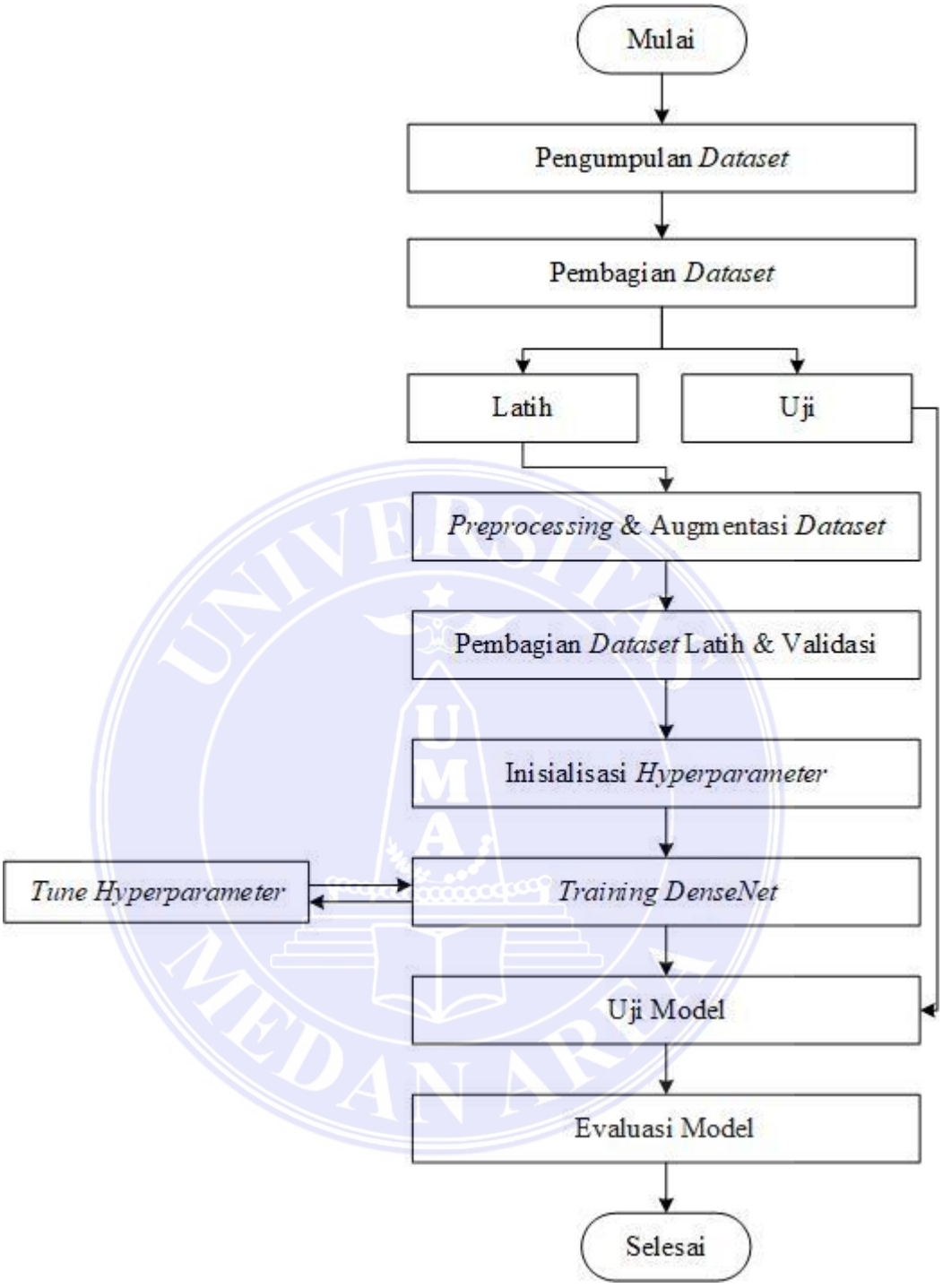


Gambar 3.1. Alur Metodologi Penelitian

3.3. Alur Perancangan Model *DenseNet-169*

Alur merancang model *deep learning* diawali dengan mengumpulkan *dataset* yang dibutuhkan, dilanjutkan dengan dilakukan tahap pembagian *dataset* menjadi data *training*, data *validation*, dan data *testing*. Selanjutnya tahap *preprocessing* dan augmentasi *dataset* dilakukan, *preprocessing* dilakukan dengan tujuan mempersiapkan *dataset* sesuai kebutuhan, sementara augmentasi dilakukan dengan tujuan memperbanyak jumlah *dataset* yang ada. Dilanjutkan dengan

pemanggilan model *DenseNet-169* menggunakan library *TensorFlow*. Selanjutnya tahap inisialisasi *hyperparameter* dilakukan, penentuan *hyperparameter* akan mempengaruhi bagaimana sebuah model dilatih, kemudian model akan dilatih menggunakan *dataset* yang telah dibagi sebelumnya. Setelah model selesai dilatih maka akan dilakukan tahap pengujian terhadap model. Tahap terakhir adalah melakukan evaluasi kinerja model dalam melakukan klasifikasi untuk mengetahui akurasi yang dihasilkan. Diagram alur perancangan model *deep learning* dapat dilihat pada Gambar 3.2.



Gambar 3.2. Alur Perancangan Model

3.4. Pengumpulan *Dataset*

Jenis data yang digunakan dalam penelitian merupakan data berjenis sekunder dalam bentuk citra yang diperoleh melalui *website* Kaggle. Kaggle

merupakan sebuah *website* repositori data berbasis *cloud* untuk menyimpan data. *Keyword* yang digunakan untuk mendapatkan data citra dari penyakit daun tanaman padi adalah *Leaf Rice Disease.* Dataset ini terbagi atas 3 jenis kategori daun yaitu daun dengan kategori *blast* sebanyak 80 citra, daun dengan kategori *blight* sebanyak 80 citra, dan daun dengan kategori *tungro* sebanyak 80 citra dengan masing-masing citra menggunakan format warna *RGB.* Semua data citra berukuran 1440 x 1920 piksel dengan format gambar jpg. Data berasal dari Sulawesi Tenggara, Indonesia. Adapun sampel data gambar dan jumlah data dari 3 jenis penyakit ditunjukkan pada gambar dan tabel berikut.

(a) (b) (c)

Gambar 3.3. Contoh Dataset Penyakit Pada Daun Padi
(a) Blight, (b) Blast, (c) Tungro

Tabel 3.3 Jumlah *Dataset*

Data Penyakit *Blast*	80
Data Penyakit *Blight*	80
Data Penyakit *Tungro*	80
Total	240

3.5. Pembagian *Dataset* Pengujian

Dataset yang telah ada selanjutnya akan dibagi menjadi data latih, data validasi, dan data uji. Pembagian data latih dan validasi digunakan untuk melatih model dalam mengklasifikasikan citra. Data latih berfungsi untuk melatih model dan data uji berfungsi untuk melakukan pengujian ketika model telah selesai melakukan pelatihan. Pada tahapan ini akan dilakukan pembagian data untuk dataset pengujian, kemudian sisa *dataset* akan diperbanyak melalui proses augmentasi, hasil *dataset* dari proses augmentasi ini selanjutnya dibagi atas data latih dan data validasi. *Data* uji akan berjumlah 32 citra dari setiap kelas data sehingga total data uji akan berjumlah 96 yang bersumber dari 32 data penyakit *blast,* 32 penyakit *blight,* dan 32 penyakit *tungro.* Pengambilan data uji akan dilakukan secara acak. Adapun pembagian data dapat dilihat pada tabel 3.4 berikut.

Tabel 3.4 Pembagian *Dataset* Latih & *Dataset* Uji

Jenis Data	Jumlah Data
Data Latih	144
Data Uji	96

3.6. *Augmentation & Dataset Preprocessing*

Jumlah *dataset* yang sedikit akan mempengaruhi akurasi dari model ketika melakukan pelatihan. Semakin banyak *dataset* yang digunakan maka kualitas model akan semakin baik dalam mengklasifikasikan penyakit pada daun tanaman padi. Augmentasi data dapat dilakukan untuk menambah jumlah *dataset.* Augmentasi data merupakan teknik memperbanyak jumlah citra dengan melakukan proses modifikasi pada citra. Augmentasi data dilakukan menggunakan *library*

TensorFlow yaitu *ImageDataGenerator*. *ImageDataGenerator* memiliki beberapa parameter yang dapat digunakan untuk melakukan augmentasi data. Penjelasan setiap parameter dapat dilihat pada tabel berikut.

Tabel 3.5 Parameter *ImageDataGenerator*

NO	Parameter	Keterangan
1	*Rotation Range*	Merotasi atau melakukan perputaran pada citra
2	*Width Shift Range*	Melakukan pergeseran gambar secara horizontal
3	*Height Shift Range*	Melakukan pergeseran gambar secara vertikal
4	*Shear Range*	Mengubah kemiringan gambar
5	*Zoom Range*	Melakukan pembesaran gambar
6	*Fill Mode*	Mengisi ruang kosong pada gambar
7	*Horizontal Flip*	Membalik gambar secara horizontal
8	*Vertical Flip*	Membalik gambar secara vertikal

Dataset Preprocessing merupakan tahap pengolahan *dataset* untuk mendapatkan spesifikasi *dataset* yang diinginkan sebelum *dataset* digunakan dalam proses *training* model. *Dataset Preprocessing* mencakup proses seperti pembersihan data, pengubahan data, dan pengurangan dimensi data. *Dataset Preprocessing* sangat penting untuk mendapatkan kinerja model yang baik. Tahapan *preprocessing* pada penelitian ini adalah *resize image.*

Resize merupakan proses mengubah ukuran citra secara horizontal maupun vertikal. Proses resize bertujuan untuk mengurangi waktu komputasi pada saat proses pelatihan model serta menyeragamkan ukuran dari citra. Pada penelitian ini

ukuran citra adalah 1140 * 1920 piksel. Citra akan di *resize* menjadi 224 x 224 piksel.

3.7. Pembagian *Dataset* Pelatihan & *Dataset* Validasi

Dataset pelatihan yang berjumlah 144 citra dengan masing-masing kelas memiliki sebanyak 48 citra akan diperbanyak melalui proses augmentasi hingga akhirnya keseluruhan *dataset* akan berjumlah sebanyak 3744 citra dengan masing-masing kelas memiliki 1248 citra. Adapun jumlah keseluruhan citra setelah melalui proses augmentasi dapat dilihat pada tabel berikut.

Tabel 3.6 Jumlah *Dataset* Setelah Augmentasi

Data Penyakit *Blast*	1248
Data Penyakit *Blight*	1248
Data Penyakit *Tungro*	1248
Total	3744

Dataset dari setiap kelas ini selanjutnya akan dibagi atas data latih dan data validasi menggunakan rasio pembagian 80:20. Data latih akan berjumlah 998 citra dari setiap kelas sehingga total data latih akan berjumlah 2994 citra yang bersumber dari 998 data penyakit *blast,* 998 penyakit *blight,* dan 998 penyakit *tungro.* Sementara data validasi akan berjumlah 250 citra dari setiap kelas sehingga total data latih akan berjumlah 750 citra yang bersumber dari 250 data penyakit *blast,* 250 penyakit *blight,* dan 250 penyakit *tungro.* Adapun pembagian *dataset* uji dan *dataset* latih dapat dilihat pada tabel berikut.

Tabel 3.7 Pembagian *Dataset* Latih & *Dataset* Validasi

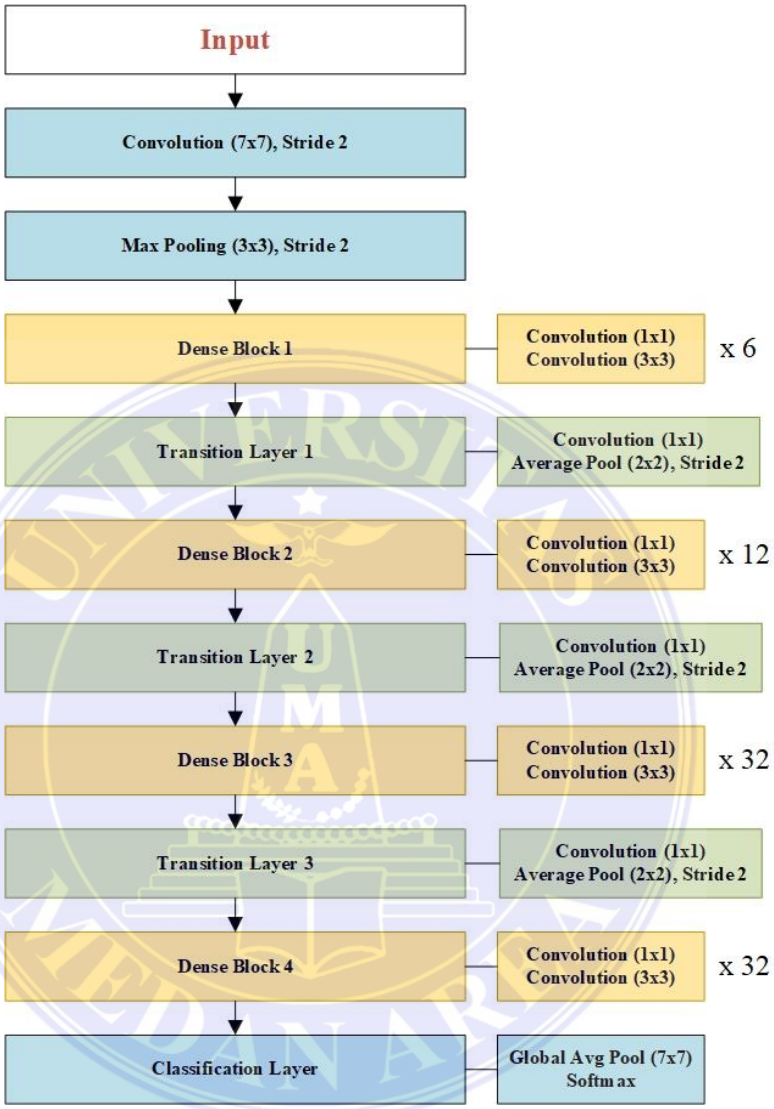
Jenis Data	Jumlah Data
Data Latih	2994
Data Validasi	750

3.8. Perancangan Arsitektur *DenseNet-169*

DenseNet-169 adalah *Dense Convolutional Neural Network* yang memiliki kedalaman lapisan sebanyak 169 lapisan. Setiap lapisan pada blok *dense* terhubung dengan lapisan berikutnya. Arsitektur *DenseNet-169* diawali dengan lapisan *convolutional* dan lapisan *pooling*. *DenseNet-169* terdiri dari 4 blok *dense* yang diakhiri dengan lapisan *transition*. Pada lapisan terakhir terdapat fungsi *softmax* untuk menghitung probabilitas dari masing-masing kelas target. Pada penelitian ini *DenseNet-169* akan digunakan melalui library *TensorFlow.*

Pada lapisan *convolutional* akan terjadi beberapa proses diantaranya *Batch Normalization* (BN), *Rectified Linear Unit* (ReLU), dan konvolusi. Selanjutnya pada setiap blok *dense* terdiri atas 2 konvolusi dengan ukuran 1x1 dan 3x3. Pada *DenseNet-169* terdapat 4 blok *dense* yang terdiri atas 6 lapisan, 12 lapisan, 32 lapisan, dan 32 lapisan serta setiap lapisannya terdiri atas 2 konvolusi dengan ukuran 1x1 dan 3x3. Terdapat lapisan *transition* yang terdiri atas konvolusi 1x1 dan *average pooling* dengan *stride* 2, lapisan *transition* ini terletak setelah blok *dense* yang pertama, kedua, dan ketiga. Blok *dense* keempat terhubung dengan lapisan klasifikasi yang menggunakan fungsi *softmax*. Angka 169 pada densenet mengartikan kedalaman dari arsitektur tersebut yang berasal dari 6+12+32+32 = 82 lapisan yang terdiri atas 2 konvolusi yaitu 1x1 dan 3x3, maka 82*2=164 lapisan konvolusi, lapisan *transition* sebanyak 3, lapisan klasifikasi sebanyak 1 dan lapisan *convolutional* pada awal arsitektur sebanyak 1 sehingga total lapisan adalah

164+3+1+1 = 169. Detail dari arsitektur *DenseNet-169* dapat dilihat pada gambar berikut.



Gambar 3.4. Arsitektur DenseNet-169

3.9. Inisialisasi *Hyperparamater*

Hyperparameter adalah parameter yang ditentukan sebelum model memulai pelatihan. *Hyperparameter* mempengaruhi bagaimana sebuah model *machine learning* mempelajari data. Pengunaan *hyperparameter* yang tepat dapat meningkatkan kinerja model *machine learning* baik dari akurasi model ataupun

waktu pembelajaran model. *Hyperparameter-hyperparameter* yang digunakan pada penelitian ini adalah sebagai berikut.

Tabel 3.8 *Hyperparameter*

Hyperparameter	**Nilai**
Epoch	40
Batch Size	64
Optimizer	SGD
Learning Rate	0.001

Berdasarkan tabel 3.6 di atas diketahui *hyperparameter* yang digunakan pada proses penelitian ini adalah *Epoch*, *Batch Size*, *Optimizer*, dan *Learning Rate*. Untuk *hyperparameter Epoch* nilai yang yang digunakan adalah 40. Nilai *Batch Size* yang digunakan adalah 64. Nilai *Learning Rate* yang digunakan adalah 0,001. *Optimizer* yang digunakan SGD. Pemilihan keseluruhan nilai *hyperparameter* ini berdasarkan dari penelitian terhadap *DenseNet-169* ketika arsitektur ini diciptakan pada tahun 2018.

3.10. *Tune Hyperparamater*

Hyperparameter adalah parameter yang ditentukan sebelum model memulai pelatihan. *Hyperparameter* mempengaruhi bagaimana sebuah model *machine learning* dilatih yang berdampak pada kinerja model yang dihasilkan. Pengunaan *hyperparamter* yang tepat dapat meningkatkan kinerja model *machine learning*. Proses untuk menemukan nilai *hyperparameter* yang optimal disebut

dengan *hyperparameter tuning*. *Hyperparameter tuning* merupakan tahapan yang sangat penting dalam mengoptimalkan kinerja dari algoritma *machine learning*.

Hyperparameter tuning dapat dilakukan secara manual dengan menguji kumpulan *hyperparameter* yang telah ditentukan sebelumnya secara satu persatu. Pada penelitian ini, *tuning hyperparameter* akan dilakukan secara otomatis menggunakan metode *grid search* dan *random search*. Metode *grid search* akan mencoba seluruh kemungkinan kombinasi *hyperparameter* yang telah ditentukan sebelumnya. Sementara *random search* hanya akan mencoba beberapa kombinasi *hyperparameter* yang dipilih secara acak. *Hyperparameter* beserta nilainya yang akan dilakukan proses *tuning* dilihat pada tabel berikut.

Tabel 3.9 *Tuning Hyperparameter*

Hyperparameter	**Nilai**
Epoch	10, 25, 50, 75
Batch Size	16, 32, 64
Optimizer	SGD, RMSProp, Adagrad, Adadelta, Adam, Adamax, Nadam
Learning Rate	0.001, 0.01, 0.1, 0.2, 0.3

Berdasarkan tabel 3.7 di atas *hyperparameter* yang akan di*tuning* adalah *epoch*, *batch size*, *optimizer* dan *learning rate*. Nilai *epoch* yang digunakan adalah 10, 25, 50, 75. Nilai *batch size* yang digunakan adalah 16, 32, dan 64. *Optimizer* yang digunakan adalah Adam, SGD, RMSProp, Adagrad, Adadelta, Adam, Adamax, dan Nadam. Nilai *learning rate* yang digunakan adalah 0.001, 0.01, dan 0.1, 0.2, dan 0.3. Nilai-nilai dari *hyperparameter-hyperparameter* hasil tuning

metode *grid search* dan *random search* akan digunakan untuk melakukan pelatihan model untuk mendapatkan model dengan kinerja terbaik.

3.11. Evaluasi Model

Tahapan terakhir setelah model dilatih adalah melakukan evaluasi. Evaluasi dilakukan untuk mengetahui bagaimana kinerja model dalam mengklasifikasikan penyakit daun tanaman padi. Salah satu metode untuk mengukur kinerja dari suatu model klasifikasi adalah menggunakan nilai yang diperoleh berdasarkan *confusion matrix*. Kinerja dari model dapat diketahui berdasarkan *accuracy*, *precision*, *recall*, dan *f1-score*. Pengukuran kinerja dapat dilakukan berdasarkan rumus berikut.

a) *Accuracy*

Merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{..... (3.1)}$$

b) *Precision*

Merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif.

$$Precision = \frac{TP}{TP+FP} \quad \text{..... (3.2)}$$

c) *Recall*

Merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif.

$$Recall = \frac{TP}{TP+FN} \quad \text{..... (3.3)}$$

d) *F1-Score*

Merupakan perbandingan rata-rata *precision* dan *recall*.

$$F1\text{-}Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad \text{..... (3.4)}$$

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil penelitian ini dapat disimpulkan bahwa implementasi arsitektur DenseNet-169 untuk mengklasifikasikan penyakit pada daun tanaman padi bekerja dengan baik. Arsitektur *DenseNet-169* memperoleh hasil terbaik dengan menggunakan model hasil *tuning grid search* dengan menggunakan nilai hyperparameter *epoch* sebesar 25, *batch size* 32, *optimizer* Adam, *dan learning rate* sebesar 0.001. Hasil pengujian dari model ini mendapatkan nilai *precision* 93%, *recall* 93%, *f1-score* 93%, dan *accuracy* 93%.

5.2. Saran

Adapun saran untuk pengembangan terkait penelitian ini adalah sebagai berikut:

1. Menambahkan lebih banyak jumlah dan variasi *dataset* terkait penyakit pada daun tanaman padi atau dapat menggunakan data primer untuk melakukan proses pelatihan, proses validasi, dan proses uji.
2. Melakukan pemgembangan aplikasi lebih lanjut seperti melakukan *deployment* model ke dalam aplikasi berbasis website ataupun android untuk mempermudah pengunaan dari model yang telah dibuat.

DAFTAR PUSTAKA

Afis Julianto, Andi Sunyoto, & Ferry Wahyu Wibowo. (2022). Optimasi Hyperparameter Convolutional Neural Network Untuk Klasifikasi Penyakit Tanaman Padi. *TEKNIMEDIA: Teknologi Informasi Dan Multimedia*, *3*(2), 98–105. https://doi.org/10.46764/teknimedia.v3i2.77

Agustiani, S., Tajul Arifin, Y., Junaidi, A., Khotimatul Wildah, S., & Mustopa, A. (2022). Klasifikasi Penyakit Daun Padi menggunakan Random Forest dan Color Histogram. *Jurnal Komputasi*, *10*(1). https://doi.org/10.23960/komputasi.v10i1.2961

Agustin, E., Eviyanti, A., & Lutvi Azizah, N. (2023). Deteksi Penyakit Epilepsi Melalui Sinyal EEG Menggunakan Metode DWT dan Extreme Gradient Boosting. *Jurnal Media Informatika Budidarma*, *7*(1), 117–127. https://doi.org/10.30865/mib.v7i1.5412

Alamsyah, D., & Pratama, D. (2020). Implementasi Convolutional Neural Networks (CNN) untuk Klasifikasi Ekspresi Citra Wajah pada FER-2013 Dataset. *(JurTI) Jurnal Teknologi Informasi*, *4*(2), 350–355.

Ananta Dwi Prayoga Alwy, M Syahid Nur Wahid, Bukhari Naufal Nur Ag, & M Miftach Fakhri. (2023). Klasifikasi Penyakit Pada Padi Dengan Ekstraksi Fitur LBP dan GLCM. In *Journal of Deep Learning, Computer Vision and Digital Image Processing*. https://doi.org/10.61255/decoding.v1i1.51

Anggiratih, E., Siswanti, S., Octaviani, S. K., & Sari, A. (2021). Klasifikasi Penyakit Tanaman Padi Menggunakan Model Deep Learning Efficientnet B3 dengan Transfer Learning. *Jurnal Ilmiah SINUS*, *19*(1), 75. https://doi.org/10.30646/sinus.v19i1.526

Bowo, T. A., Syaputra, H., & Akbar, M. (2020). Penerapan Algoritma Convolutional Neural Network Untuk Klasifikasi Motif Citra Batik Solo. *Journal of Software Engineering Ampera*, *1*(2), 82–96. https://doi.org/10.51519/journalsea.v1i2.47

Breve, F. A. (2020). *COVID-19 detection on Chest X-ray images: A comparison of*

CNN architectures and ensembles. *January*.

Dwi Antoko, T., Azhar Ridani, M., & Eko Minarno, A. (2021). Klasifikasi Buah Zaitun Menggunakan Convolution Neural Network. *Komputika : Jurnal Sistem Komputer*, *10*(2), 119–126. https://doi.org/10.34010/komputika.v10i2.4475

Dzaky, A. T. R. (2021). Deteksi Penyakit Tanaman Cabai Menggunakan Metode Convolutional Neural Network. *E-Proceeding of Engineering*, *8*(2), 3040–3055.

Eka Kusumawati, D., & Istiqomah, I. (2020). Potensi Agensia Hayati Dalam Menekan Laju Serangan Penyakit Blas (Pyricularia Oryzae) Pada Tanaman Padi. *VIABEL: Jurnal Ilmiah Ilmu-Ilmu Pertanian*, *14*(2), 1–13. https://doi.org/10.35457/viabel.v14i2.1235

Eros Fikri Syahram. (2021). Sun Position Forecasting Menggunakan Metode RNN – LSTM Sebagai Referensi Pengendalian Daya Solar Cell. *Jurnal JEETech*, *2*(2), 65–77. https://doi.org/10.48056/jeetech.v2i2.169

Fahrezantara, A., Elektro, F. T., Telkom, U., Rizal, S., Elektro, F. T., Telkom, U., Kumalasari, N., Pratiwi, C., Elektro, F. T., & Telkom, U. (2022). *Pemanfaatan Convolutional Neural Network Dalam Klasifikasi Penyakit Tanaman Singkong Menggunakan Arsitektur Densenet Use Of Convolutional Neural Networks On Classifying Cassava Diseases With Densenet Architecture*. *8*(6), 3332–3338.

Hawari, F. H., Fadillah, F., Alviandi, M. R., & Arifin, T. (2022). Klasifikasi Penyakit Tanaman Padi Menggunakan Algoritma Cnn (Convolutional Neural Network). *Jurnal Responsif : Riset Sains Dan Informatika*, *4*(2), 184–189. https://doi.org/10.51977/jti.v4i2.856

Iswantoro, D., & Handayani UN, D. (2022). Klasifikasi Penyakit Tanaman Jagung Menggunakan Metode Convolutional Neural Network (CNN). *Jurnal Ilmiah Universitas Batanghari Jambi*, *22*(2), 900. https://doi.org/10.33087/jiubj.v22i2.2065

Khoiruddin, M., Junaidi, A., & Saputra, W. A. (2022). Klasifikasi Penyakit Daun Padi Menggunakan Convolutional Neural Network. *Journal of Dinda : Data Science, Information Technology, and Data Analytics*, *2*(1), 37–45. https://doi.org/10.20895/dinda.v2i1.341

Kotta, C. R., Paseru, D., & Sumampouw, M. (2022). Implementasi Metode Convolutional Neural Network untuk Mendeteksi Penyakit pada Citra Daun Tomat. *Jurnal_Pekommas_Vol._7_No*, *2*, 123–132.

Kusuma Putra, A., & Bunyamin, H. (2020). Pengenalan Simbol Matematika dengan Metode Convolutional Neural Network (CNN). *Jurnal Strategi*, *2*(November), 426.

Lidya Fankky Oktavia Putri, Ahmad Junjung Sudrajad, & Vinna Rahmayanti Setyaning Nastiti. (2022). Classification of Face Mask Detection Using Transfer Learning Model DenseNet169. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *6*(5), 790–796. https://doi.org/10.29207/resti.v6i5.4336

Lu, J., Tan, L., & Jiang, H. (2021). Review on convolutional neural network (CNN) applied to plant leaf disease classification. *Agriculture (Switzerland)*, *11*(8), 1–18. https://doi.org/10.3390/agriculture11080707

Nisa', C., Puspaningrum, E. Y., & Maulana, H. (2020). Penerapan Metode Convolutional Neural Network untuk Klasifikasi Penyakit Daun Apel pada Imbalanced Data. *Prosiding Seminar Nasional Informatika Bela Negara*, *1*, 169–175. https://doi.org/10.33005/santika.v1i0.46

Noprisson, H. (2022). Fine-Tuning Model Transfer Learning VGG16 Untuk Klasifikasi Citra Penyakit Tanaman Padi. *JSAI (Journal Scientific and Applied Informatics)*, *5*(3), 244–249. https://doi.org/10.36085/jsai.v5i3.3609

Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (CNN) Pada Ekspresi Manusia. *Algor*, *2*(1), 12–21.

Ollivia, A., & Pratiwi, C. (2023). Klasifikasi Jenis Anggur Berdasarkan Bentuk Daun Menggunakan Convolutional Neural Network Dan K-Nearest Neighbor.

Jurnal Ilmiah Teknik Informatika Dan Komunikasi, *3*(2), 201–224.

Paraijun, F., Aziza, R. N., & Kuswardani, D. (2022). Implementasi Algoritma Convolutional Neural Network Dalam Mengklasifikasi Kesegaran Buah Berdasarkan Citra Buah. *Kilat*, *11*(1), 1–9. https://doi.org/10.33322/kilat.v10i2.1458

Purnamawati, A., Nugroho, W., Putri, D., & Hidayat, W. F. (2020). Deteksi Penyakit Daun pada Tanaman Padi Menggunakan Algoritma Decision Tree, Random Forest, Naïve Bayes, SVMdan KNN. *InfoTekJar: Jurnal Nasional Informatika Dan Teknologi Jaringan*, *5*(1), 212–215. https://doi.org/10.30743/infotekjar.v5i1.2934

Ramdan, A., Zilvan, V., Suryawati, E., Pardede, H. F., & Rahadi, V. P. (2020). Tea clone classification using deep CNN with residual and densely connections. *Jurnal Teknologi Dan Sistem Komputer*, *8*(4), 289–296. https://doi.org/10.14710/jtsiskom.2020.13768

Raup, A., Ridwan, W., Khoeriyah, Y., Supiana, S., & Zaqiah, Q. Y. (2022). Deep Learning dan Penerapannya dalam Pembelajaran. *JIIP - Jurnal Ilmiah Ilmu Pendidikan*, *5*(9), 3258–3267. https://doi.org/10.54371/jiip.v5i9.805

Rusman, M. A. A., Darsono, & Antriyandarti, E. (2023). " Akselerasi Hasil Penelitian dan Optimalisasi Tata Ruang Agraria untuk Mewujudkan Pertanian Berkelanjutan " Analisis Forecasting Produksi Padi di Indonesia. *Seminar Nasional Dalam Rangka Dies Natalis Ke-47 UNS Tahun 2023*, *7*(1), 728–739.

Saga, A. M. A. J. P. A. (2020). Peta Penyebaran Dan Citra Foto Udara Penyakit Tungro Pada Tanaman Padi Di Desa Marapokot Kecamatan Aesesa Kabupaten Nagekeo. *AGRICA*, *Vol. 13 No. 2 (2020): December*, 102–116. http://e-journal.uniflor.ac.id/index.php/Agr/article/view/746/754

Saputra, R. A., Suharyanto, Wasiyanti, S., Saefudin, D. F., Supriyatna, A., & Wibowo, A. (2020). Rice Leaf Disease Image Classifications Using KNN Based on GLCM Feature Extraction. *Journal of Physics: Conference Series*, *1641*(1). https://doi.org/10.1088/1742-6596/1641/1/012080

Saputra, R. A., Wasiyanti, S., Supriyatna, A., & Saefudin, D. F. (2021). Penerapan Algoritma Convolutional Neural Network Dan Arsitektur MobileNet Pada Aplikasi Deteksi Penyakit Daun Padi. *Swabumi*, *9*(2), 184–188. https://doi.org/10.31294/swabumi.v9i2.11678

Septian, M. R. D., Paliwang, A. A. A., Cahyanti, M., & Swedia, E. R. (2020). Penyakit Tanaman Apel Dari Citra Daun Dengan Convolutional Neural Network. *Sebatik*, *24*(2), 207–212. https://doi.org/10.46984/sebatik.v24i2.1060

Shinta, R., Irsyad, M., Yanto, Fe., & Sanjaya, S. (2023). Klasifikasi Citra Penyakit Daun Tanaman Padi Menggunakan CNN dengan Arsitektur VGG-19. *Jurnal Sains Dan Informatika*, *9*(1), 37–45. https://doi.org/10.22216/jsi.v9i1.2175

Sholawati, M., Auliasari, K., & Ariwibisono, F. (2022). Pengembangan Aplikasi Pengenalan Bahasa Isyarat Abjad Sibi Menggunakan Metode Convolutional Neural Network (Cnn). *JATI (Jurnal Mahasiswa Teknik Informatika)*, *6*(1), 134–144. https://doi.org/10.36040/jati.v6i1.4507

Sitompul, P., Okprana, H., Prasetio, A., & Artikel, G. (2022). Identifikasi Penyakit Tanaman Padi Melalui Citra Daun Menggunakan DenseNet 201 Identification of Rice Plant Diseases Through Leaf Image Using DenseNet 201 Article Info ABSTRAK. *JOMLAI: Journal of Machine Learning and Artificial Intelligence*, *1*(2), 143–150. https://doi.org/10.55123/jomlai.v1i2.889

Sudalto. (2022). Implementasi Convolutional Neural Network untuk Klasifikasi Citra Soil Nikel dan Non Nikel. *G-Tech: Jurnal Teknologi Terapan*, *6*(1), 85–90. https://doi.org/10.33379/gtech.v6i1.1273

Tilasefana, R. A., & Putra, R. E. (2023). Penerapan Metode Deep Learning Menggunakan Algoritma CNN Dengan Arsitektur VGG NET Untuk Pengenalan Cuaca. *Journal of Informatics and Computer Science (JINACS)*, *05*(1), 48–57.

Ulfah Nur Oktaviana, Ricky Hendrawan, Alfian Dwi Khoirul Annas, & Galih Wasis Wicaksono. (2021). Klasifikasi Penyakit Padi berdasarkan Citra Daun Menggunakan Model Terlatih Resnet101. *Jurnal RESTI (Rekayasa Sistem*

Dan Teknologi Informasi), *5*(6), 1216–1222. https://doi.org/10.29207/resti.v5i6.3607

Wahid islahfahri, M., mustamin akbar, S., & Lawi, A. (2021). Identifikasi Dan Klasifikasi Citra Penyakit Daun Tomat Menggunakan Arsitektur Inception V4. *Onferensi Nasional Ilmu Komputer*, *5*(2019), 257–264.

Wicaksono, G., & Andryana, S. (2020). Aplikasi Pendeteksi Penyakit Pada Daun Tanaman Apel Dengan Metode Convolutional Neural Network. *JOINTECS (Journal of Information Technology and Computer Science)*, *5*(1), 9–16.

Yuliany, S., Aradea, & Andi Nur Rachman. (2022). Implementasi Deep Learning pada Sistem Klasifikasi Hama Tanaman Padi Menggunakan Metode Convolutional Neural Network (CNN). *Jurnal Buana Informatika*, *13*(1), 54–65. https://doi.org/10.24002/jbi.v13i1.5022

LAMPIRAN

1. Hasil Pengecekan Plagiasi Turnitin

turnitin

Similarity Report ID: oid:29477:51057927

PAPER NAME	AUTHOR
1.pdf	Muhammad Yazid Abud Asseweth
WORD COUNT	**CHARACTER COUNT**
9462 Words	56387 Characters
PAGE COUNT	**FILE SIZE**
63 Pages	1.1MB
SUBMISSION DATE	**REPORT DATE**
Feb 6, 2024 3:05 PM GMT+7	Feb 6, 2024 3:06 PM GMT+7

19% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 16% Internet database
- 5% Publications database
- Crossref database
- Crossref Posted Content database
- 12% Submitted Works database

Excluded from Similarity Report

- Small Matches (Less then 10 words)

Summary

2. Surat Keputusan Pembimbing Tugas Akhir

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

Nomor : 729/FT.6/01.10/X/2023 16 Oktober 2023
Lamp : -
H a l : **Perubahan Judul Tugas Akhir**

Yth, Pembimbing Tugas Akhir
Muhathir, ST, M. Kom
di
Tempat

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

N a m a : Muhammad Yazid Abud Asseweth
N P M : 188160054
Jurusan : Teknik Informatika

Maka dengan hormat kami mengharapkan kesediaan saudara :

Muhathir, ST, M. Kom **(Sebagai Pembimbing)**

Adapun Tugas Akhir Skripsi berjudul :

"Klasifikasi Penyakit pada Daun Tanaman Padi menggunakan Arsitektur *DenseNet-169*".

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

Demikian kami sampaikan, atas kesediaan saudara diucapkan terima kasih.

Dekan,

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

3. Surat Izin Melakukan Penelitian dan Pengambilan Data Tugas Akhir

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

Nomor : 750 /FT.6/01.10/X/2023 23 Oktober 2023
Lamp : -
H a l : **Penelitian Dan Pengambilan Data Tugas Akhir**

Yth. Kepala Dinas Pertanian Provinsi Sumatera Utara
Jln. A.H Nasution No.6
Di
Medan

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

NO	N A M A	N P M	PRODI
1	Muhammad Yazid Abud Asseweth	188160054	Teknik Informatika

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

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

Klasifikasi Penyakit pada Daun Tanaman Padi menggunakan Arsitektur *DenseNet-169*

Atas perhatian dan kerja sama yang baik diucapkan terima kasih.

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

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

4. Surat Selesai Melakukan Penelitian

PEMERINTAH PROVINSI SUMATERA UTARA
DINAS KETAHANAN PANGAN, TANAMAN PANGAN DAN HORTIKULTURA
UPTD. PERLINDUNGAN TANAMAN PANGAN, HORTIKULTURA
DAN PENGAWASAN MUTU KEAMANAN PANGAN
Jalan Jenderal Besar Dr. Abdul Haris Nasution No. 4 Pangkalan Masyhur Medan 20143
Telepon (061) 7880230, Telefax (061) 7880230 Email : uptptphmedan@gmail.com

Medan, 01 Nopember 2023

Nomor : 521.4/4081/UPTD. PTPH dan PMKP
Lampiran : -
Perihal : **Penelitian Dan Pengambilan Data Tugas Akhir Telah Selesai**

Yth.
Dekan Fakultas Teknik
Universitas Medan Area
di -
Medan

Menindaklanjuti Surat kami No: 521.4/4051/UPTD. PTPH dan PMKP tanggal 30 Oktober 2023 perihal Permohonan Penelitian Dan Pengambilan Data Tugas Akhir di di UPTD. Perlindungan Tanaman Pangan, Hortikultura dan Pengawasan Mutu Keamanan Pangan Dinas Ketahanan Pangan, Tanaman Pangan dan Hortikultura Provinsi Sumatera Utara, yang telah dilaksanakan pada tanggal 31 Oktober 2023. Bersama ini kami sampaikan Mahasiswa dimaksud atas nama sebagai berikut :

NAMA	NPM	Prodi
Muhammad Yazid Abud Asseweth	188160054	Teknik Informatika

Telah selesai melaksanakan Penelitian Dan Pengambilan Data Tugas Akhir di UPTD. Perlindungan Tanaman Pangan, Hortikultura dan Pengawasan Mutu Keamanan Pangan Dinas Ketahanan Pangan, Tanaman Pangan dan Hortikultura Provinsi Sumatera Utara.

Demikian kami sampaikan, atas perhatian dan kerjasamanya diucapkan terima kasih.

Kepala UPTD. Perlindungan Tanaman Pangan,
Hortikultura dan Pengawasan Mutu
Keamanan Pangan

Marino, SP, MM
Pembina
NIP.196901131993031004

Tembusan :
1. Bapak Kepala Dinas Ketahanan Pangan, Tanaman Pangan dan Hortikultura Provsu

5. Source Code

a. Proses Augmentasi Dataset

```
import os
import shutil
import cv2

from google.colab import files
from google.colab import drive

drive.mount('/content/drive')

!cp
/content/drive/MyDrive/SKRIPSI/2024/splitted_dataset_original.z
ip /content/

! unzip splitted_dataset_original.zip

dataset_path = '/content/training'
blast_original_dataset_path = os.path.join(dataset_path,
"blast")
blight_original_dataset_path = os.path.join(dataset_path,
"blight")
tungro_original_dataset_path = os.path.join(dataset_path,
"tungro")

augmented_dataset_path = '/content/augmented_dataset'
if os.path.exists(augmented_dataset_path):
  shutil.rmtree(augmented_dataset_path);

os.mkdir(augmented_dataset_path)

blast_augmented_path = os.path.join(augmented_dataset_path,
'blast')
blight_augmented_path = os.path.join(augmented_dataset_path,
'blight')
tungro_augmented_path = os.path.join(augmented_dataset_path,
'tungro')

os.mkdir(blast_augmented_path)
os.mkdir(blight_augmented_path)
os.mkdir(tungro_augmented_path)

# copying base blast to blast augmented folder before adding
augmented images
for fileName in os.listdir(blast_original_dataset_path):
  image = os.path.join(blast_original_dataset_path, fileName)
  destination_path = os.path.join(blast_augmented_path,
fileName)
  shutil.copy(image, destination_path)

# copying base blight to blight augmented folder before adding
augmented images
for fileName in os.listdir(blight_original_dataset_path):
  image = os.path.join(blight_original_dataset_path, fileName)
```

```
    destination_path = os.path.join(blight_augmented_path,
fileName)
    shutil.copy(image, destination_path)

# copying base tungro to tungro augmented folder before adding
augmented images
for fileName in os.listdir(tungro_original_dataset_path):
    image = os.path.join(tungro_original_dataset_path, fileName)
    destination_path = os.path.join(tungro_augmented_path,
fileName)
    shutil.copy(image, destination_path)

print(f"Terdapat {len(os.listdir(blast_augmented_path))} citra
penyakit blast dalam folder augmented dataset kategori blast.")
print(f"Terdapat {len(os.listdir(blight_augmented_path))} citra
penyakit blight dalam folder augmented dataset kategori
blight.")
print(f"Terdapat {len(os.listdir(tungro_augmented_path))} citra
penyakit tungro dalam folder augmented dataset kategori
tungro.")

base_dataset_path = "/content/base_dataset"
os.mkdir(base_dataset_path)

blast_parent_path = os.path.join(base_dataset_path, "blast")
blight_parent_path = os.path.join(base_dataset_path, "blight")
tungro_parent_path = os.path.join(base_dataset_path, "tungro")

blast_child_path = os.path.join(blast_parent_path, "blast")
blight_child_path = os.path.join(blight_parent_path, "blight")
tungro_child_path = os.path.join(tungro_parent_path, "tungro")

os.mkdir(blast_parent_path)
os.mkdir(blight_parent_path)
os.mkdir(tungro_parent_path)

os.mkdir(blast_child_path)
os.mkdir(blight_child_path)
os.mkdir(tungro_child_path)

# copying base blight, blast, & tungro to separate dir of each
their own class so augmentation will not generate all of 3
class
for fileName in os.listdir(blast_original_dataset_path):
    image = os.path.join(blast_original_dataset_path, fileName)
    destination_path = os.path.join(blast_child_path, fileName)
    shutil.copy(image, destination_path)

for fileName in os.listdir(blight_original_dataset_path):
    image = os.path.join(blight_original_dataset_path, fileName)
    destination_path = os.path.join(blight_child_path, fileName)
    shutil.copy(image, destination_path)

for fileName in os.listdir(tungro_original_dataset_path):
    image = os.path.join(tungro_original_dataset_path, fileName)
    destination_path = os.path.join(tungro_child_path, fileName)
    shutil.copy(image, destination_path)
```

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=0.2,
    shear_range=0.2,
    fill_mode = 'nearest')

batches = datagen.flow_from_directory(
    blast_parent_path,
    target_size=(224, 224),
    class_mode='categorical',
    shuffle=False,
    batch_size=48,
    save_prefix='aug',
    save_to_dir=blast_augmented_path)

i = 1
for batch, label in batches:
  i += 1
  if i > 25:
    break

from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=0.2,
    shear_range=0.2,
    fill_mode = 'nearest')

batches = datagen.flow_from_directory(
    blight_parent_path,
    target_size=(224, 224),
    class_mode='categorical',
    shuffle=False,
    batch_size=48,
    save_prefix='aug',
    save_to_dir=blight_augmented_path)

i = 1
for batch, label in batches:
  i += 1
  if i > 25:
    break

from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rotation_range=20,
    width_shift_range=0.2,
```

```
        height_shift_range=0.2,
        horizontal_flip=True,
        vertical_flip=True,
        zoom_range=0.2,
        shear_range=0.2,
        fill_mode = 'nearest')

    batches = datagen.flow_from_directory(
        tungro_parent_path,
        target_size=(224, 224),
        class_mode='categorical',
        shuffle=False,
        batch_size=48,
        save_prefix='aug',
        save_to_dir=tungro_augmented_path)

    i = 1
    for batch, label in batches:
      i += 1
      if i > 25:
        break

    print(f"Terdapat {len(os.listdir(blast_augmented_path))} citra
    penyakit blast dalam folder augmented dataset kategori blast.")
    print(f"Terdapat {len(os.listdir(blight_augmented_path))} citra
    penyakit blight dalam folder augmented dataset kategori
    blight.")
    print(f"Terdapat {len(os.listdir(tungro_augmented_path))} citra
    penyakit tungro dalam folder augmented dataset kategori
    tungro.")

    !zip -r augmented_dataset.zip /content/augmented_dataset/

    !cp augmented_dataset.zip /content/drive/MyDrive/SKRIPSI/2024/
```

b. Proses Resizing Dataset

```
    import os
    import shutil
    import cv2

    from google.colab import files
    from google.colab import drive

    drive.mount('/content/drive')

    !cp /content/drive/MyDrive/SKRIPSI/2024/augmented_dataset.zip
    /content/

    resized_dataset_path = '/content/resized_dataset'
    os.mkdir(resized_dataset_path)

    resized_blast_path = os.path.join(resized_dataset_path, 'blast')
    resized_blight_path = os.path.join(resized_dataset_path,
    'blight')
```

```
resized_tungro_path = os.path.join(resized_dataset_path,
'tungro')

os.mkdir(resized_blast_path)
os.mkdir(resized_blight_path)
os.mkdir(resized_tungro_path)

# extract augmented file
!unzip /content/augmented_dataset.zip

source_path = '/content/content/augmented_dataset'

source_path_blast = os.path.join(source_path, "blast");
source_path_blight = os.path.join(source_path, "blight");
source_path_tungro = os.path.join(source_path, "tungro");

print(f"Terdapat {len(os.listdir(source_path_blast))} citra
penyakit blast.")
print(f"Terdapat {len(os.listdir(source_path_blight))} citra
penyakit blight.")
print(f"Terdapat {len(os.listdir(source_path_tungro))} citra
penyakit tungro.")

# resize blast
for image in os.listdir(source_path_blast):
  imageFile = os.path.join(source_path_blast, image)
  img = cv2.imread(imageFile)
  resized_img = cv2.resize(img, (224, 224))
  save_path = os.path.join(resized_blast_path, image)
  cv2.imwrite(save_path, resized_img)

# resize blight
for image in os.listdir(source_path_blight):
  imageFile = os.path.join(source_path_blight, image)
  img = cv2.imread(imageFile)
  resized_img = cv2.resize(img, (224, 224))
  save_path = os.path.join(resized_blight_path, image)
  cv2.imwrite(save_path, resized_img)

# resize tungro
for image in os.listdir(source_path_tungro):
  imageFile = os.path.join(source_path_tungro, image)
  img = cv2.imread(imageFile)
  resized_img = cv2.resize(img, (224, 224))
  save_path = os.path.join(resized_tungro_path, image)
  cv2.imwrite(save_path, resized_img)

print(f"Terdapat {len(os.listdir(resized_blast_path))} citra
penyakit blast pada folder resized.")
print(f"Terdapat {len(os.listdir(resized_blight_path))} citra
penyakit blast pada folder resized.")
print(f"Terdapat {len(os.listdir(resized_tungro_path))} citra
penyakit blast pada folder resized.")

!zip -r /content/resized_dataset.zip /content/resized_dataset/
```

```
!cp                                    /content/resized_dataset.zip
/content/drive/MyDrive/SKRIPSI/2024/
```

c. Proses Pembagian Dataset

```
from google.colab import files
from google.colab import drive

drive.mount('/content/drive')

!cp /content/drive/MyDrive/SKRIPSI/2024/resized_dataset.zip
/content/
! unzip resized_dataset.zip

import os
import shutil

source_path = '/content/content/resized_dataset'

source_path_blast = os.path.join(source_path, "blast");
source_path_blight = os.path.join(source_path, "blight");
source_path_tungro = os.path.join(source_path, "tungro");

print(f"Terdapat {len(os.listdir(source_path_blast))} citra
penyakit blast.")
print(f"Terdapat {len(os.listdir(source_path_blight))} citra
penyakit blight.")
print(f"Terdapat {len(os.listdir(source_path_tungro))} citra
penyakit tungro.")

splitted_dataset_path =
'/content/splitted_dataset_training_validation'

if os.path.exists(splitted_dataset_path):
  shutil.rmtree(splitted_dataset_path);

training_dir = os.path.join(splitted_dataset_path, 'training')
validation_dir = os.path.join(splitted_dataset_path,
'validation')
test_dir = os.path.join(splitted_dataset_path, 'test')

os.makedirs(training_dir)
os.makedirs(validation_dir)
os.makedirs(test_dir)

blast_training_dir = os.path.join(training_dir, 'blast')
blight_training_dir = os.path.join(training_dir, 'blight')
tungro_training_dir = os.path.join(training_dir, 'tungro')

blast_validation_dir = os.path.join(validation_dir, 'blast')
blight_validation_dir = os.path.join(validation_dir, 'blight')
tungro_validation_dir = os.path.join(validation_dir, 'tungro')

blast_test_dir = os.path.join(test_dir, 'blast')
```

```
blight_test_dir = os.path.join(test_dir, 'blight')
tungro_test_dir = os.path.join(test_dir, 'tungro')

os.makedirs(blast_training_dir)
os.makedirs(blight_training_dir)
os.makedirs(tungro_training_dir)

os.makedirs(blast_validation_dir)
os.makedirs(blight_validation_dir)
os.makedirs(tungro_validation_dir)

os.makedirs(blast_test_dir)
os.makedirs(blight_test_dir)
os.makedirs(tungro_test_dir)

# menampilkan folder traing dan validation yang telah dibuat

for rootdir, dirs, files in os.walk(splitted_dataset_path):
    for subdir in dirs:
        print(os.path.join(rootdir, subdir))

from sklearn.model_selection import train_test_split

train_blast_images, validation_blast_images =
train_test_split(os.listdir(source_path_blast), test_size = 250)
train_blight_images, validation_blight_images =
train_test_split(os.listdir(source_path_blight), test_size =
250)
train_tungro_images, validation_tungro_images =
train_test_split(os.listdir(source_path_tungro), test_size =
250)

print(len(train_blast_images))
print(len(train_blight_images))
print(len(train_tungro_images))

print(len(validation_blast_images))
print(len(validation_blight_images))
print(len(validation_tungro_images))

# Pindahkan file validation ke folder validationnya masing-
masing
for file in validation_blast_images:
  shutil.copy(os.path.join(source_path_blast, file),
os.path.join(blast_validation_dir, file))

for file in validation_blight_images:
  shutil.copy(os.path.join(source_path_blight, file),
os.path.join(blight_validation_dir, file))

for file in validation_tungro_images:
  shutil.copy(os.path.join(source_path_tungro, file),
os.path.join(tungro_validation_dir, file))

# Pindahkan file train ke folder trainingnya masing-masing
for file in train_blast_images:
```

```
    shutil.copy(os.path.join(source_path_blast, file),
os.path.join(blast_training_dir, file))

for file in train_blight_images:
    shutil.copy(os.path.join(source_path_blight, file),
os.path.join(blight_training_dir, file))

for file in train_tungro_images:
    shutil.copy(os.path.join(source_path_tungro, file),
os.path.join(tungro_training_dir, file))

print(f"Terdapat {len(os.listdir(blast_training_dir))} citra
penyakit blast dalam folder training kategori blast.")
print(f"Terdapat {len(os.listdir(blight_training_dir))} citra
penyakit blight dalam folder training kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_training_dir))} citra
penyakit tungro dalam folder training kategori tungro.")

print(f"Terdapat {len(os.listdir(blast_validation_dir))} citra
penyakit blast dalam folder validation kategori blast.")
print(f"Terdapat {len(os.listdir(blight_validation_dir))} citra
penyakit blight dalam folder validation kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_validation_dir))} citra
penyakit tungro dalam folder validation kategori tungro.")

print(f"Terdapat {len(os.listdir(blast_test_dir))} citra
penyakit blast dalam folder test kategori blast.")
print(f"Terdapat {len(os.listdir(blight_test_dir))} citra
penyakit blight dalam folder test kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_test_dir))} citra
penyakit tungro dalam folder test kategori tungro.")

# hasil akhir pada folder test setiap kelas terdapat 32 file
(40% dari 80 gambar), sementara folder validasi masih kosong
karena gambar akan diaugmentasi kemudian baru displit
shutil.make_archive('/content/splitted_dataset_training_validati
on', 'zip', '/content/splitted_dataset_training_validation')

!cp splitted_dataset_training_validation.zip
/content/drive/MyDrive/SKRIPSI/


! unzip splitted_dataset_training_validation.zip
```

d. Klasifikasi Dataset Testing Menggunakan Model Default DenseNet-169

```
from google.colab import drive

drive.mount('/content/drive')

!cp /content/drive/MyDrive/dataset_final.zip /content/
! unzip dataset_final.zip -d dataset

import os
import tensorflow as tf
```

```
import keras
import shutil

from keras.preprocessing.image import ImageDataGenerator
from keras.applications import DenseNet169
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import LearningRateScheduler

dataset_path = '/content/dataset'

os.makedirs(dataset_path)

training_dir = os.path.join(dataset_path, 'training')
validation_dir = os.path.join(dataset_path, 'validation')
test_dir = os.path.join(dataset_path, 'test')

blast_training_dir = os.path.join(training_dir, 'blast')
blight_training_dir = os.path.join(training_dir, 'blight')
tungro_training_dir = os.path.join(training_dir, 'tungro')

blast_validation_dir = os.path.join(validation_dir, 'blast')
blight_validation_dir = os.path.join(validation_dir, 'blight')
tungro_validation_dir = os.path.join(validation_dir, 'tungro')

blast_test_dir = os.path.join(test_dir, 'blast')
blight_test_dir = os.path.join(test_dir, 'blight')
tungro_test_dir = os.path.join(test_dir, 'tungro')

# menampilkan folder traing dan validation yang telah dibuat

for rootdir, dirs, files in os.walk(dataset_path):
    for subdir in dirs:
        print(os.path.join(rootdir, subdir))

print(f"Terdapat {len(os.listdir(blast_training_dir))} citra
penyakit blast dalam folder training kategori blast.")
print(f"Terdapat {len(os.listdir(blight_training_dir))} citra
penyakit blight dalam folder training kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_training_dir))} citra
penyakit tungro dalam folder training kategori tungro.")

print(f"Terdapat {len(os.listdir(blast_validation_dir))} citra
penyakit blast dalam folder validation kategori blast.")
print(f"Terdapat {len(os.listdir(blight_validation_dir))} citra
penyakit blight dalam folder validation kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_validation_dir))} citra
penyakit tungro dalam folder validation kategori tungro.")

print(f"Terdapat {len(os.listdir(blast_test_dir))} citra
penyakit blast dalam folder test kategori blast.")
print(f"Terdapat {len(os.listdir(blight_test_dir))} citra
penyakit blight dalam folder test kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_test_dir))} citra
penyakit tungro dalam folder test kategori tungro.")

train_datagen = ImageDataGenerator(
    rescale=1./255,)
```

```
val_datagen = ImageDataGenerator(
    rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    training_dir,
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    validation_dir,
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical')

pre_trained_model = DenseNet169(include_top=False,
weights="imagenet", input_shape=(224, 224, 3))

model = tf.keras.models.Sequential([
    pre_trained_model,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    # tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])

for layer in pre_trained_model.layers:
    layer.trainable = False

model.compile(optimizer=tf.optimizers.SGD(),
              loss='categorical_crossentropy',
              metrics = ['accuracy'])

history = model.fit(train_generator,
                    validation_data=validation_generator,
                    epochs=40,
                    verbose=2)

model.save("model_default.h5")
print("Saved model to disk")

!zip -r /content/model_default.zip /content/model_default.h5

!cp /content/model_default.zip
/content/drive/MyDrive/skripsi/2024/Hasil_Model_Default

import keras
from matplotlib import pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
```

```
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.savefig('accuracy_default.png')
plt.show()

!cp /content/accuracy_default.png
/content/drive/MyDrive/skripsi/2024/Hasil_Model_Default

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.savefig('loss_default.png')
plt.show()

!cp /content/loss_default.png
/content/drive/MyDrive/skripsi/2024/Hasil_Model_Default

score = model.evaluate(validation_generator)
print('Val loss:', score[0])
print('Val accuracy:', score[1])

test_datagen = ImageDataGenerator(
    rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=64,
    shuffle=False,
    class_mode='categorical')

import itertools

#Plot the confusion matrix. Set Normalize = True/False
def plot_confusion_matrix(cm, classes, normalize=True,
title='Confusion matrix', cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.figure(figsize=(10,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
    thresh = cm.max() / 2.
```

```
        for i, j in itertools.product(range(cm.shape[0]),
    range(cm.shape[1])):
            plt.text(j, i, cm[i, j],
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else
    "black")
        plt.tight_layout()
        plt.ylabel('True label')
        plt.xlabel('Predicted label')

    # Some reports
    from sklearn.metrics import classification_report,
    confusion_matrix
    import numpy as np

    classes = list(train_generator.class_indices.keys())

    #Confution Matrix and Classification Report
    Y_pred = model.predict(test_generator)#, nb_test_samples //
    BATCH_SIZE, workers=1)
    y_pred = np.argmax(Y_pred, axis=1)
    target_names = classes

    #Confution Matrix
    cm = confusion_matrix(test_generator.classes, y_pred)
    plot_confusion_matrix(cm, target_names, normalize=False,
    title='Confusion Matrix')
    print('Classification Report')
    print(classification_report(test_generator.classes, y_pred,
    target_names=target_names))

    !cp                  /content/loss_default_80_20_percobaan_1.png
    /content/drive/MyDrive/SKRIPSI/default/plot_hasil_percobaan_1/
```

e. Klasifikasi Dataset Testing Menggunakan Model DenseNet-169 Grid Search

```
from google.colab import drive

drive.mount('/content/drive')

!cp /content/drive/MyDrive/dataset_final.zip /content/
! unzip dataset_final.zip -d dataset

import os
import tensorflow as tf
import keras
import shutil

from keras.preprocessing.image import ImageDataGenerator
from keras.applications import DenseNet169
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import LearningRateScheduler

dataset_path = '/content/dataset'
```

```
training_dir = os.path.join(dataset_path, 'training')
validation_dir = os.path.join(dataset_path, 'validation')
test_dir = os.path.join(dataset_path, 'test')

blast_training_dir = os.path.join(training_dir, 'blast')
blight_training_dir = os.path.join(training_dir, 'blight')
tungro_training_dir = os.path.join(training_dir, 'tungro')

blast_validation_dir = os.path.join(validation_dir, 'blast')
blight_validation_dir = os.path.join(validation_dir, 'blight')
tungro_validation_dir = os.path.join(validation_dir, 'tungro')

blast_test_dir = os.path.join(test_dir, 'blast')
blight_test_dir = os.path.join(test_dir, 'blight')
tungro_test_dir = os.path.join(test_dir, 'tungro')

# menampilkan folder traing dan validation yang telah dibuat

for rootdir, dirs, files in os.walk(dataset_path):
    for subdir in dirs:
        print(os.path.join(rootdir, subdir))

print(f"Terdapat {len(os.listdir(blast_training_dir))} citra
penyakit blast dalam folder training kategori blast.")
print(f"Terdapat {len(os.listdir(blight_training_dir))} citra
penyakit blight dalam folder training kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_training_dir))} citra
penyakit tungro dalam folder training kategori tungro.")

print(f"Terdapat {len(os.listdir(blast_validation_dir))} citra
penyakit blast dalam folder validation kategori blast.")
print(f"Terdapat {len(os.listdir(blight_validation_dir))} citra
penyakit blight dalam folder validation kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_validation_dir))} citra
penyakit tungro dalam folder validation kategori tungro.")

print(f"Terdapat {len(os.listdir(blast_test_dir))} citra
penyakit blast dalam folder test kategori blast.")
print(f"Terdapat {len(os.listdir(blight_test_dir))} citra
penyakit blight dalam folder test kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_test_dir))} citra
penyakit tungro dalam folder test kategori tungro.")

train_datagen = ImageDataGenerator(
    rescale=1./255,)

val_datagen = ImageDataGenerator(
    rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    training_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
```

```
        validation_dir,
        target_size=(224, 224),
        batch_size=32,
        class_mode='categorical')

    pre_trained_model = DenseNet169(include_top=False,
    weights="imagenet", input_shape=(224, 224, 3))

    model = tf.keras.models.Sequential([
        pre_trained_model,
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(256, activation='relu'),
        tf.keras.layers.Dense(128, activation='relu'),
        # tf.keras.layers.Dropout(0.4),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(3, activation='softmax')
    ])

    for layer in pre_trained_model.layers:
        layer.trainable = False

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

    history = model.fit(train_generator,
                        validation_data=validation_generator,
                        epochs=25,
                        verbose=2)

    model.save("model_grid_search.h5")
    print("Saved model to disk")

    !zip -r /content/model_grid_search.zip
    /content/model_grid_search.h5

    !cp /content/model_grid_search.zip
    /content/drive/MyDrive/skripsi/2024/Hasil_Model_Grid_Search

    import keras
    from matplotlib import pyplot as plt

    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'val'], loc='upper left')
    plt.savefig('accuracy_grid_search.png')
    plt.show()

    !cp /content/accuracy_grid_search.png
    /content/drive/MyDrive/skripsi/2024/Hasil_Model_Grid_Search

    plt.plot(history.history['loss'])
```

```
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.savefig('loss_grid_search.png')
plt.show()

!cp /content/loss_grid_search.png
/content/drive/MyDrive/skripsi/2024/Hasil_Model_Grid_Search

score = model.evaluate(validation_generator)
print('Val loss:', score[0])
print('Val accuracy:', score[1])

test_datagen = ImageDataGenerator(
    rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=64,
    shuffle=False,
    class_mode='categorical')

score = model.evaluate(test_generator)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

import itertools

#Plot the confusion matrix. Set Normalize = True/False
def plot_confusion_matrix(cm, classes, normalize=True,
title='Confusion matrix', cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.figure(figsize=(10,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else
"black")
    plt.tight_layout()
```

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

    # Some reports
    from sklearn.metrics import classification_report,
    confusion_matrix
    import numpy as np

    classes = list(train_generator.class_indices.keys())

    #Confution Matrix and Classification Report
    Y_pred = model.predict(test_generator)#, nb_test_samples //
    BATCH_SIZE, workers=1)
    y_pred = np.argmax(Y_pred, axis=1)
    target_names = classes

    #Confution Matrix
    cm = confusion_matrix(test_generator.classes, y_pred)
    plot_confusion_matrix(cm, target_names, normalize=False,
    title='Confusion Matrix')
    print('Classification Report')
    print(classification_report(test_generator.classes, y_pred,
    target_names=target_names))

    !cp                  /content/loss_default_80_20_percobaan_1.png
    /content/drive/MyDrive/SKRIPSI/default/plot_hasil_percobaan_1/
```

f. Klasifikasi Dataset Testing Menggunakan Model DenseNet-169 Random Search

```
    from google.colab import drive

    drive.mount('/content/drive')

    !cp /content/drive/MyDrive/dataset_final.zip /content/
    ! unzip dataset_final.zip -d dataset

    import os
    import tensorflow as tf
    import keras
    import shutil

    from keras.preprocessing.image import ImageDataGenerator
    from keras.applications import DenseNet169
    from sklearn.model_selection import train_test_split
    from tensorflow.keras.callbacks import LearningRateScheduler

    dataset_path = '/content/dataset'

    training_dir = os.path.join(dataset_path, 'training')
    validation_dir = os.path.join(dataset_path, 'validation')
    test_dir = os.path.join(dataset_path, 'test')

    blast_training_dir = os.path.join(training_dir, 'blast')
    blight_training_dir = os.path.join(training_dir, 'blight')
```

```
tungro_training_dir = os.path.join(training_dir, 'tungro')

blast_validation_dir = os.path.join(validation_dir, 'blast')
blight_validation_dir = os.path.join(validation_dir, 'blight')
tungro_validation_dir = os.path.join(validation_dir, 'tungro')

blast_test_dir = os.path.join(test_dir, 'blast')
blight_test_dir = os.path.join(test_dir, 'blight')
tungro_test_dir = os.path.join(test_dir, 'tungro')

# menampilkan folder traing dan validation yang telah dibuat

for rootdir, dirs, files in os.walk(dataset_path):
    for subdir in dirs:
        print(os.path.join(rootdir, subdir))

print(f"Terdapat {len(os.listdir(blast_training_dir))} citra
penyakit blast dalam folder training kategori blast.")
print(f"Terdapat {len(os.listdir(blight_training_dir))} citra
penyakit blight dalam folder training kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_training_dir))} citra
penyakit tungro dalam folder training kategori tungro.")

print(f"Terdapat {len(os.listdir(blast_validation_dir))} citra
penyakit blast dalam folder validation kategori blast.")
print(f"Terdapat {len(os.listdir(blight_validation_dir))} citra
penyakit blight dalam folder validation kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_validation_dir))} citra
penyakit tungro dalam folder validation kategori tungro.")

print(f"Terdapat {len(os.listdir(blast_test_dir))} citra
penyakit blast dalam folder test kategori blast.")
print(f"Terdapat {len(os.listdir(blight_test_dir))} citra
penyakit blight dalam folder test kategori blight.")
print(f"Terdapat {len(os.listdir(tungro_test_dir))} citra
penyakit tungro dalam folder test kategori tungro.")

train_datagen = ImageDataGenerator(
    rescale=1./255,)

val_datagen = ImageDataGenerator(
    rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    training_dir,
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    validation_dir,
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical')

pre_trained_model = DenseNet169(include_top=False,
weights="imagenet", input_shape=(224, 224, 3))
```

```
model = tf.keras.models.Sequential([
    pre_trained_model,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    # tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])

for layer in pre_trained_model.layers:
    layer.trainable = False

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

history = model.fit(train_generator,
                    validation_data=validation_generator,
                    epochs=75,
                    verbose=2)

model.save("model_random_search.h5")
print("Saved model to disk")

!zip -r /content/model_random_search.zip
/content/model_random_search.h5

!cp /content/model_random_search.zip
/content/drive/MyDrive/skripsi/2024/Hasil_Model_Random_Search

import keras
from matplotlib import pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.savefig('accuracy_random_search.png')
plt.show()

!cp /content/accuracy_random_search.png
/content/drive/MyDrive/skripsi/2024/Hasil_Model_Random_Search

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.savefig('loss_random_search.png')
plt.show()
```

```
!cp /content/loss_random_search.png
/content/drive/MyDrive/skripsi/2024/Hasil_Model_Random_Search

score = model.evaluate(validation_generator)
print('Val loss:', score[0])
print('Val accuracy:', score[1])

test_datagen = ImageDataGenerator(
    rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=64,
    shuffle=False,
    class_mode='categorical')

score = model.evaluate(test_generator)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

import itertools

#Plot the confusion matrix. Set Normalize = True/False
def plot_confusion_matrix(cm, classes, normalize=True,
title='Confusion matrix', cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.figure(figsize=(10,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else
"black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Some reports
from sklearn.metrics import classification_report,
confusion_matrix
import numpy as np
```

```
classes = list(train_generator.class_indices.keys())

#Confution Matrix and Classification Report
Y_pred = model.predict(test_generator)#, nb_test_samples //
BATCH_SIZE, workers=1)
y_pred = np.argmax(Y_pred, axis=1)
target_names = classes

#Confution Matrix
cm = confusion_matrix(test_generator.classes, y_pred)
plot_confusion_matrix(cm, target_names, normalize=False,
title='Confusion Matrix')
print('Classification Report')
print(classification_report(test_generator.classes, y_pred,
target_names=target_names))

!cp                    /content/loss_default_80_20_percobaan_1.png
/content/drive/MyDrive/SKRIPSI/default/plot_hasil_percobaan_1/
```