IDENTIFIKASI PENYAKIT PADA TANAMAN APEL MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERBASIS CITRA DAUN

SKRIPSI

Oleh:
TAUFIQURRAHMAN IDRUS
NIM. 17650088

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2022

IDENTIFIKASI PENYAKIT PADA TANAMAN APEL MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERBASIS CITRA DAUN

SKRIPSI

Diajukan kepada:
Fakulas Sains dan Teknologi
Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S. Kom)

Oleh:
TAUFIQURRAHMAN IDRUS
NIM. 17650088

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2022

HALAMAN PERSETUJUAN

IDENTIFIKASI PENYAKIT PADA TANAMAN APEL MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERBASIS CITRA DAUN

SKRIPSI

Oleh:
TAUFIQURRAHMAN IDRUS
NIM. 17650088

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal : 24 Juni 2022

Dosen Pembimbing I	Dosen Pembimbing II
Irwan Budi Santoso, M.Kom	Dr. M. Amin Hariyadi
NIP. 19770103 201101 1 004	NIP. 19670018 200501 1 001

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan ST., M.MT., IPM
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

IDENTIFIKASI PENYAKIT PADA TANAMAN APEL MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERBASIS CITRA DAUN

SKRIPSI

Oleh:
TAUFIQURRAHMAN IDRUS
NIM. 17650088

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan
Dinyatakan Diterima Sebagai Salah Satu Persyaratan untuk
Memperoleh Gelar Sarjana Komputer (S. Kom)
Tanggal: 24 Juni 2022

Susunan Dewan Penguji:

Penguji Utama	:	Dr. Totok Chamidy, M. Kom NIP. 19691222 200604 1 001	( )
Ketua Penguji	:	Zainal Abidin, M.Kom NIP. 19760613 200501 1 004	( )
Sekretaris Penguji	:	Irwan Budi Santoso, M.Kom NIP. 19770103 201101 1 004	( )
Anggota Penguji	:	Dr. M. Amin Hariyadi NIP. 19670018 200501 1 001	( )

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan ST., M.MT., IPM
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama : Taufiqurrahman Idrus

Nim : 17650088

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

Judul Skripsi : Identifikasi Penyakit Pada Tanaman Apel Menggunakan Convolutional Neural Network Berbasis Citra Daun

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilan daya, tulisan atau pikiran orang lain yang saya akui menjadi hasil tulisan atau pikiran saya sendiri , kecuali dengan mencantumkan sumber cuplikan atau referensi pada daftar pustaka.

Apabila pada kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan saya.

Malang, 27 Juni 2022

Yang membuat pernyataan,

Taufiqurrahman Idrus

NIM. 17650088

HALAMAN MOTTO

"*Be The Change You Want to See in The World*"
- Mahatma Gandhi

HALAMAN PERSEMBAHAN

Bismillahirrohmanirrohim, Alhamdulillah puji syukur atas nikmat yang telah Allah SWT berikan sehingga penulis dapat menuntaskan skripsi ini untuk menyelesaikan program S1 di kampus Ulul Albab tercinta dengan lancar dan baik pula. Tak lupa pula shalawat serta salam dijunjungkan kepada baginda Rasulullah SAW yang selalu dirindukan dan diidolakan.

Terima Kasih banyak kepada kedua orangtua penulis, teruntuk abah tercinta yaitu Muhammad Idrus yang selalu memberikan dorongan dan motivasi serta mendidik anaknya untuk menjadi pribadi yang Tangguh, disiplin, jujur dan berwibawa serta harus selalu siap dengan kerasnya kehidupan. Teruntuk ummi tercinta Nurbaya yang selalu mengajarkan kasih sayang, penyabar, dan juga rendah hati kepada anaknya. Tak lupa untuk kedua adik Muhammad Izzulhaq Idrus dan Amir Mahmud Idrus semoga selalu tercapai apa yang dicita-citakan dan menjadi kebanggaan untuk keluarga, bangsa dan negara.

Teruntuk seluruh guru dan dosen mulai dari sekolah dasar hingga perguruan tinggi. Terkhusus untuk dosen pembimbingku bapak Irwan Budi Santoso, M.Kom dan Dr. M. Amin Hariyadi yang selalu tulus, ikhlas dan sabar dalam membimbing dan selalu mengarahkan dengan sangat baik. Pesan baik kalian akan selalu kuingat dan tak putus doa untuk seluruh guru dan dosen tercinta hingga akhir hayatku.

Dan teruntuk seluruh rekan Unocore dan rekan tercinta yang tidak bisa penulis sebutkan satu persatu.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Segala puji bagi Allah subhanahu wa ta'ala yang senantiasa memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Tujuan dari penyusunan skripsi ini guna memenuhi salah satu syarat untuk bisa menempuh ujian sarjana komputer pada Fakultas Sains dan Teknologi (FSAINTEK) Program Studi Teknik Informatika di Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang. Di dalam pengerjaan skripsi ini telah melibatkan banyak pihak yang sangat membantu dalam banyak hal. Oleh sebab itu, disini penulis sampaikan rasa terima kasih sedalam-dalamnya kepada :

1. Prof. Dr. H. M. Zainuddin, MA selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Fachrul Kurniawan ST., M.MT., IPM selaku Ketua Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Irwan Budi Santoso, M.Kom. selaku dosen pembimbing I yang selalu memberikan penulis dorongan, motivasi yang sangat kuat dan selalu membantu penulis dalam menyelesaikan permasalahan yang penulis temukan dalam bimbingan untuk menyelesaikan skripsi ini.

5. Dr. M. Amin Hariyadi. selaku dosen pembimbing II yang selalu membantu penulis dalam menyelesaikan kesulitan dan selalu memberikan solusi setiap masalah yang penulis hadapi.
6. Dr. Totok Chamidy, M. Kom dan Bapak Zainal Abidin, M. Kom selaku dosen penguji yang sangat profesional dalam melakukan pengujian terhadap proses ujian skripsi mulai dari seminar proposal hingga sidang skripsi yang berjalan dengan baik.
7. Seluruh jajaran dosen dan staf jurusan teknik informatika UIN Malang baik secara langsung maupun tidak langsung dalam menyelesaikan skripsi ini.
8. Orang tua tercinta serta keluarga yang selalu memberikan semangat dan kasih sayang sehingga penulis dapat menyelesaikan skripsi ini tepat waktu.
9. Rekan-rekan seperjuangan Teknik Informatika 2017 Unocore yang penulis cintai dan penulis banggakan

Malang, 27 Juni 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Idrus, Taufiqurrahman. 2022. **Identifikasi Penyakit Pada Tanaman Apel Menggunakan Convolutional Neural Network Berbasis Citra Daun.** Skripsi. Jurusan Teknik Informatika, Fakultas Sains dan Teknologi. Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Irwan Budi Santoso, M.Kom., (II) Dr. M. Amin Hariyadi.

Kata Kunci: *Apel, Convolutional neural network, Identifikasi Penyakit Tanaman*

Daerah Malang Raya memproduksi apel dalam jumlah yang besar setiap tahunnya. Dengan jumlah produksi yang tinggi tentu saja masalah seperti penurunan hasil panen dan kualitas buah menjadi ancaman yang serius. Salah satu penyebab masalah ini adalah penyakit yang menyerang tanaman apel. Beberapa jenis penyakit pada tanaman apel antara lain adalah: bercak daun, cacar daun dan busuk buah. Penyakit-penyakit ini dapat diidentifikasi gejalanya dengan memeriksa daunnya. Oleh karena itu perlu dibuat sebuah sistem yang dapat mengidentifikasi penyakit pada tanaman apel secara otomatis. Sistem ini akan dibangun dengan mengimplementasikan algoritma *Convolutional Neural Network.* Penelitian ini bertujuan untuk mengukur akurasi, sensitivitas, dan spesifisitas algoritma *Convolutional Neural Network* dalam mengidentifikasi penyakit pada tanaman apel. Algoritma ini dipilih karena dapat melakukan identifikasi citra secara efektif. Total data yang digunakan terdiri dari 1517 citra daun yang sehat dan berpenyakit. Data menggunakan tiga pencahayaan berbeda yaitu lampu kamar, lampu LED dan cahaya matahari. Data terdiri dari empat *class* berbeda yaitu bercak daun, cacar daun, busuk buah beserta apel sehat. Preprocessing dilakukan dengan teknik augmentasi data seperti *rotation, width shift, height shift, shear, zoom, horizontal flip.* Arsitektur CNN yang digunakan terdiri dari 2 *convolutional layer,* 2 *max pooling layer,* 1 *flatten layer*, 1 *dense layer*, dan 1 *output layer.* Pengujian dilakukan dengan metode *K-Fold Cross Validation.* Penelitian ini berhasil memperoleh hasil terbaik saat dilakukan uji coba menggunakan data dengan pencahayaan cahaya matahari yaitu dengan nilai akurasi rata-rata 88.60%, sensitivitas rata-rata 77.20% dan spesifisitas rata-rata 92.40% . Kemudian dilakukan uji coba menggunakan arsitektur VGG16. Diperoleh hasil terbaik 91.50% akurasi, 83.00% sensitivitas, dan 94.33% spesifisitas pada pengujian menggunakan data dengan pencahayaan lampu kamar.

ABSTRACT

Idrus, Taufiqurrahman. 2022. **Identification of Apple Plant Disease Based on Leaf Image Using the Convolutional Neural Network.** Undergraduate Thesis. Informatics Engineering Department, Faculty of Science and Technology. Islamic State University Maulana Malik Ibrahim Malang. Supervisor: (I) Irwan Budi Santoso, M.Kom., (II) Dr. M. Amin Hariyadi.

Keywords*: Apple, Convolutional neural network, Plant disease identification*

Malang Raya area produces apples in large quantities every year. With a high production volume, of course, problems such as decreased yields and fruit quality pose a serious threat. One of the causes of this problem is a disease that attacks apple plants. Several types of diseases in apple plants include: leaf spot, smallpox and fruit rot. Symptoms of these diseases can be identified by examining the leaves. Therefore, it is necessary to create a system that can identify diseases in apple plants automatically. This system will be built by implementing the Convolutional Neural Network algorithm. This study aims to measure the accuracy, sensitivity, and specificity of the Convolutional Neural Network algorithm in identifying diseases in apple plants. This algorithm was chosen because it can identify the image effectively. The total data used consisted of 1517 images of healthy and diseased leaves. The data uses three different lighting, namely room lights, LED lights and sunlight. The data consisted of four different classes, namely leaf spot, smallpox, fruit rot and healthy apples. Preprocessing is done with data augmentation techniques such as rotation, width shift, height shift, shear, zoom, horizontal flip. The CNN architecture used consists of 2 convolutional layers, 2 max pooling layers, 1 flatten layer, 1 dense layer, and 1 output layer. The test was carried out using the K-Fold Cross Validation method. This study succeeded in obtaining the best results when tested using data with sunlight, namely with an average accuracy value of 88.60%, an average sensitivity of 77.20% and an average specificity of 92.40%. Then a trial was conducted using the VGG16 architecture. The best results obtained are 91.50% accuracy, 83.00% sensitivity, and 94.33% specificity in the test using data with room lighting.

الملخص

ادروس ، توفيق ارحمن. ٢٠٢٢. تحديد المرض في نبات التفاح باستخدام الشبكة العصبية التقوية على أساس الصورة الورقية. قسم الهندسة والمعلوماتية، كلية العلوم والتكنولوجيا، الجامعة مولانا مالك إبراهيم بمالانج. المشرف: (١) اروان بودي سانتوسو (٢) دكتور أمين هريادي

الكلمات الرئيسية: التفاح ، الشبكة العصبية الالتصامية ، تحديد أمراض النبات

تنتج منطقة مالانج رايا التفاح بكميات كبيرة كل عام. مع حجم الإنتاج الكبير ، بطبيعة الحال ، فإن مشاكل مثل انخفاض الغلة وجودة الفاكهة تشكل تهديدًا خطيرًا. أحد أسباب هذه المشكلة هو مرض يهاجم نبات التفاح. عدة أنواع من الأمراض في نبات التفاح تشمل: بقعة الأوراق والجدري وتعفن الفاكهة. يمكن التعرف على أعراض هذه الأمراض بفحص الأوراق. لذلك ، من الضروري إنشاء نظام يمكنه التعرف تلقائيًا على الأمراض في نباتات التفاح. سيتم بناء هذا النظام من خلال تنفيذ خوارزمية الشبكة العصبية التلافيفية. تهدف هذه الدراسة إلى قياس دقة وحساسية ونوعية خوارزمية الشبكة العصبية التلافيفية في التعرف على الأمراض في نبات التفاح. تم اختيار هذه الخوارزمية لأنها تستطيع التعرف على الصورة بشكل فعال. يتكون إجمالي البيانات المستخدمة من1517 صورة لأوراق سليمة ومريضة. تستخدم البيانات ثلاثة إضاءة مختلفة ، وهي أضواء الغرفة وأضواء LED وضوء الشمس. تكونت البيانات من أربع فئات مختلفة ، وهي بقعة الأوراق ، والجدري ، وتعفن الفاكهة ، والتفاح الصحي. تتم المعالجة المسبقة باستخدام تقنيات زيادة البيانات مثل الدوران ، وإزاحة العرض ، وإزاحة الارتفاع ، والقص ، والتكبير ، والوجه الأفقي. تتكون بنية CNNالمستخدمة من طبقتين تلافيفيتين وطبقتين تجميع كحد أقصى وطبقة واحدة مسطحة وطبقة واحدة كثيفة وطبقة إخراج واحدة. تم إجراء الاختبار باستخدام طريقة .K-Fold Cross Validation نجحت هذه الدراسة في الحصول على أفضل النتائج عند اختبارها باستخدام بيانات مع ضوء الشمس ، أي بمتوسط قيمة دقة 88.60٪ ، متوسط حساسية 77.20٪ ومتوسط خصوصية 92.40٪. ثم أجريت تجربة باستخدام هندسة VGG 16أفضل النتائج التي تم الحصول عليها هي دقة 91.50٪ وحساسية 83.00٪ ونوعية 94.33٪ في الاختبار باستخدام البيانات مع إضاءة الغرفة

BAB I

PENDAHULUAN

1.1 Latar Belakang

Apel merupakan merupakan salah satu buah yang banyak diproduksi di provinsi Jawa Timur khususnya daerah Malang Raya (Kabupaten Malang, Kota Batu dan Kota Malang). Berdasarkan data Badan Pusat Statistik (2020) Kabupaten Malang dan Kota Batu berturut-turut memproduksi 1.406.173 kuintal apel dan 505.254 kuintal apel. Dengan jumlah produksi yang tinggi tersebut tentu saja masalah seperti penurunan hasil panen dan kualitas buah menjadi ancaman yang serius. Salah satu penyebab masalah ini adalah penyakit yang menyerang tanaman apel.

Salah satu jenis apel yang diproduksi adalah apel manalagi (*Malus sylvestris*) Beberapa jenis penyakit pada tanaman apel ini antara lain adalah: bercak daun, cacar daun dan busuk buah. Penyakit-penyakit ini dapat diidentifikasi gejalanya dengan memeriksa daunnya. Daun merupakan bagian dari tanaman apel yang paling banyak jumlahnya dan paling mudah untuk diamati.

Dengan mengamati gejala-gejala awal penyakit pada tanaman, penyakit dapat segera diidentifikasi dan dicegah lebih awal. Penyakit pada tanaman umumnya menunjukkan gejala-gejala ringan dan pada skala yang kecil Sebelum mencapai tahap yang lebih parah dan menyebar (Kamilaris dan Prenafeta-Boldú, 2018). Secara tradisional pendeteksian penyakit masih dilakukan oleh ahli pertanian dan botanis, hal ini membutuhkan biaya tambahan, kerugian waktu dan inefisiensi (Priya dkk, 2012).

Bersama dengan perkembangan teknologi, *Deep Learning* telah banyak digunakan dalam berbagai bidang seperti informatika, medis, kelautan, agrikultur dan bisnis (Hidayat dkk, 2019). Salah satu algoritma *Deep Learning* yang populer digunakan untuk pengolahan citra adalah *Convolutional Neural Network* (CNN). CNN Merupakan pengembangan dari *Multi Layer Perceptron* (MLP) yang dirancang untuk mengolah data, salah satunya adalah data citra dua dimensi. CNN digunakan untuk mengklasifikasikan data yang berlabel menggunakan metode pembelajaran yang diawasi (*supervised learning)*, dalam metode pembelajaran yang diawasi terdapat data latih dan data uji (Kamilaris dan Prenafeta-Boldú, 2018). Yang membedakan CNN dengan *Neural Network* biasa adalah pada CNN terdapat lapisan konvolusi dan *pooling layer.* Sebelum input diproses*,* Lapisan konvolusi dan *pooling layer* akan membuat input citra menjadi lebih mudah untuk diproses pada lapisan *fully connected* karena sudah disederhanakan.

Dalam penelitian yang dilakukan oleh Baranwal dkk (2019), mereka melakukan identifikasi pada penyakit apel menggunakan metode CNN. Penyakit yang dapat diidentifikasi antara lain adalah *Apple Scab, Apple black rot, Apple cedar apple rust* beserta mendeteksi apel yang sehat. Namun, data yang digunakan merupakan data dari spesies apel yang ada di luar negeri. Sehingga, kurang tepat jika diaplikasikan di wilayah Indonesia.

Berdasarkan latar belakang di atas, penelitian ini akan mengimplementasikan algoritma CNN untuk mengidentifikasi penyakit pada tanaman apel manalagi. Identifikasi akan dilakukan berdasarkan ciri-ciri yang ada pada daun tanaman apel. Data yang digunakan berupa citra daun apel yang sehat dan berpenyakit.

Diharapkan dengan penelitian ini petani dapat mencegah menurunnya produksi apel yang disebabkan oleh penyakit karena penyakit dapat diidentifikasi lebih awal.

1.2 Pernyataan Masalah

Berdasarkan latar belakang di atas, diperoleh pernyataan masalah untuk penelitian ini yaitu, seberapa tinggi akurasi, sensitivitas, dan spesifisitas metode *Convolutional Neural Network* dalam mengidentifikasi penyakit pada tanaman apel?

1.3 Tujuan Penelitian

Berdasarkan pernyataan masalah di atas, diperoleh tujuan penelitian untuk penelitian ini yaitu, mengukur akurasi, sensitivitas, dan spesifisitas metode *Convolutional Neural Network* dalam mengidentifikasi penyakit pada tanaman apel.

1.4 Batasan Masalah

Batasan masalah dalam penelitian ini antara lain:

1) Data yang digunakan adalah citra daun tanaman apel sebanyak 1517 citra.
2) Identifikasi dilakukan pada spesies apel manalagi (*Malus sylvestris*)
3) Penyakit yang dapat diidentifikasi adalah bercak daun, cacar daun, busuk buah beserta apel yang sehat

1.5 Manfaat Penelitian

Hasil dari penelitian ini diharapkan dapat bermanfaat untuk petani apel agar dapat mengidentifikasi penyakit pada tanaman apel lebih awal dan menghindari kerugian karena gagal panen.

BAB II

STUDI PUSTAKA

Dalam bab ini akan dibahas tentang penelitian terdahulu yang terkait dengan penelitian ini. Selain itu, dalam bab ini akan dijelaskan landasan teori yang digunakan sebagai dasar teori dalam penelitian ini.

2.1 Penelitian Terkait

Dubey dan Jalal. (2016) melakukan penelitian yang mengklasifikasian penyakit pada apel berdasarkan fitur-fitur seperti warna, tekstur, dan bentuk. Penelitian ini dapat mendeteksi penyakit-penyakit apel seperti *Apple Scab, Apple rot, dan Apple blotch* serta mendeteksi apel sehat. Data yang digunakan adalah citra buah apel. Klasifikasi dilakukan dengan mendeteksi bagian buah yang terinfeksi dengan bantuan metode cluster K-means. Kemudian fitur warna, tekstur, dan bentuk dikomputasi dan digabungkan untuk untuk membentuk descriptor tunggal. Langkah akhir adalah multi-class support vector machine digunakan untuk mengklasifikasikan apel berpenyakit dan apel sehat. Akurasi yang diperoleh bervariasi sesuai dengan fitur yang dideteksi, mulai dari 61.25% hingga 100%.

Samajpati dan Degadwala. (2016) melakukan penelitian yang dapat mendeteksi penyakit-penyakit apel seperti Apple scab, Apple rot, dan Apple blotch serta mendeteksi apel sehat. Metodologi pemrosesan citra yang diusulkan menggabungkan keadaan warna dan fitur tekstur yang diekstraksi dari data latih. Kemudian warna dan tekstur digabungkan dan random forest akan mengklasifikasikan penyakit dari apel. Jika terdapat infeksi penyakit maka bagian yang terinfeksi akan disegmentasi menggunakan teknik clustering K-means.

Akurasi yang diperoleh bervariasi sesuai dengan fitur yang dideteksi, mulai dari 30% hingga 100%.

Tabel 2.1 menampilkan perbedaan penelitian ini dengan penelitian yang sudah pernah dilakukan oleh peneliti lain

Tabel 2. 1 Perbedaan dengan penelitian sebelumnya

No.	Peneliti (Tahun)	Metode dan Studi Kasus
1.	Dubey dan Jalal (2016)	- Klasifikasi menggunakan multi-class support vector machine - Penyakit yang dideteksi antara lain blotch, rot, dan scab serta apel sehat - Data berjumlah 320 citra RGB.
2.	Samajpati dan Degadwala (2016)	- Klasifikasi menggunakan random forest classifier - Penyakit yang dideteksi antara lain apple scab, apple rot, dan apple blotch
3.	Baranwal dkk (2019)	- Klasifikasi menggunakan algoritma CNN - Penyakit yang dideteksi antara lain Apple scab, black rot, dan cedar-apple rust serta apel sehat - Data berjumlah 2561 citra RGB
4.	Turkoglu dkk (2019)	- Klasifikasi menggunakan Multi-model LSTM-based Pre-trained Convolutional Neural Networks (MLP-CNNs) - Data berjumlah 1192 citra RGB - Penyakit yang dideteksi antara lain Venturia Inaequalis, Monilinia Laxa, Eriosoma Lanigerum, Aphis pomi Deg
5.	Idrus (2022)	- Klasifikasi menggunakan algoritma CNN - Identifikasi dilakukan pada spesies apel manalagi - Identifikasi dilakukan pada penyakit bercak daun, cacar daun, busuk buah beserta apel sehat - Data berjumlah 1517 citra RGB.

Baranwal dkk. (2019) melakukan penelitian yang dapat mengidentifikasi penyakit pada tanaman apel berdasarkan citra daun apel. Metode yang digunakan adalah Convolutional Neural Network. Data yang digunakan diperoleh dari Plant

Village Dataset. Penyakit yang dapat diidentifikasi antara lain adalah Apple Scab, Apple black rot, Apple cedar apple rust beserta mendeteksi apel yang sehat. Penelitian ini memperoleh akurasi 98.54% (Baranwal dkk, 2019).

Turkoglu dkk. (2019) melakukan penelitian yang mengusulkan metode terbaru yang dinamakan Multi-model LSTM-based Pre-trained Convolutional Neural Networks (MLP-CNNs). Model hybrid yang diusulkan ini dibuat berdasarkan kombinasi dari jaringan LSTM dengan model CNN pasca-latih. Data yang digunakan dalam penelitian ini bersumber dari citra penyakit tanaman apel di Turki yang diambil secara *real time*. Penelitian ini memperoleh hasil akurasi sebesar 99.2%.

2.2 Landasan Teori

Landasan teori adalah dasar-dasar teori dalam penelitian terdahulu yang berkaitan dengan penelitian ini.

2.2.1 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah sebuah algoritma *Deep Learning* yang dapat digunakan untuk melakukan pengenalan citra. CNN memiliki banyak fitur seperti struktur yang sederhana, parameter *training* yang lebih sedikit, dan adaptabilitas (Liu dkk, 2015). Selain untuk pengenalan citra, CNN banyak digunakan untuk analisis suara dan pengenalan pola.

CNN umumnya digunakan untuk mengidentifikasi perpindahan, zoom dan bentuk lain dari distorsi invarian dari grafik dua dimensi. Karena lapisan ekstraksi fitur CNN belajar dari data *training*, ekstraksi fitur secara eksplisit dan secara implisit dari data *training* dapat dihindari saat menggunakan CNN. Selain itu,

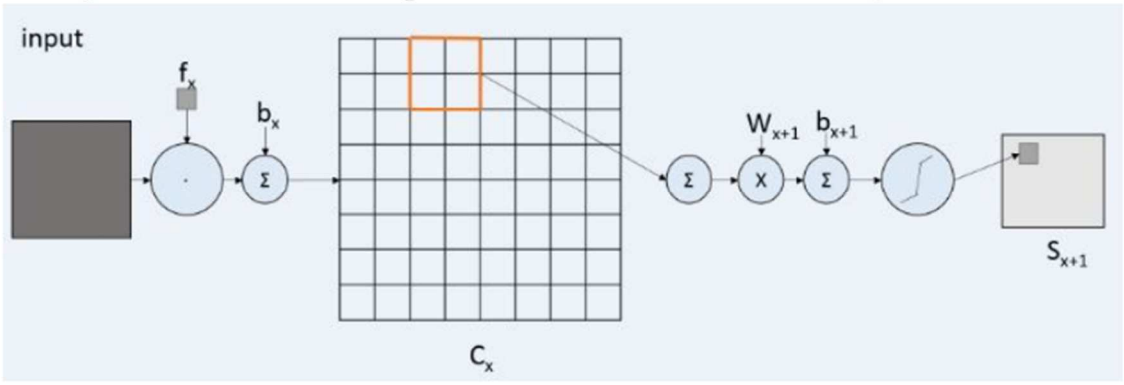
neuron pada bidang *feature map* yang sama memiliki bobot yang sama, sehingga jaringan dapat belajar secara bersamaan. Ini merupakan keuntungan utama dari CNN sehubungan dengan jaringan saraf yang terhubung satu sama lain. Struktur seperti ini yang membuat CNN memiliki keunggulan unik dalam pengenalan suara dan pemrosesan gambar. Bobot yang dibagikan mengurangi kompleksitas jaringan. (Liu dkk, 2015).

2.2.2 Prinsip Kerja CNN

Algoritma CNN adalah *multilayer perceptron* yang didesain khusus untuk identifikasi informasi citra dua dimensi. CNN selalu memiliki lapisan-lapisan seperti lapisan input, lapisan konvolusi, lapisan sampel, dan lapisan output. CNN dapat memiliki lebih dari satu lapisan konvolusi dan lapisan sampel. CNN lebih bebas dari mesin boltzmann, yang mana perlu untuk memproses setiap piksel pada citra digital. Pada CNN, setiap neuron tidak perlu memproses citra global, cukup memproses area lokal dari citra. Selain itu, setiap parameter neuron diatur sama atau disebut dengan *sharing of weight* (beban yang terbagi) yaitu setiap neuron dengan *convolution kernel* yang sama ke citra dekonvolusi. (Liu dkk, 2015).

Algoritma CNN memiliki dua proses utama yaitu konvolusi dan pengambilan sampel. Proses konvolusi: gunakan filter yang dapat dilatih Fx, dekonvolusi citra input (input tahap pertama dinamakan citra input, input setelah lapisan konvolusi adalah fitur citra dari setiap lapisan, yang dinamakan Peta Fitur), lalu tambahkan nilai bias bx, sehingga diperoleh lapisan konvolusi Cx. Proses pengambilan sampel: n piksel dari setiap ketetanggaan melalui langkah-langkah *pooling*, menjadi piksel, dan kemudian dengan pembobotan skalar Wx + 1 berbobot, tambahkan nilai bias

bx + 1, dan kemudian dengan fungsi aktivasi, menghasilkan n kali peta fitur Sx + 1 . (Liu dkk, 2015). Proses ini diilustrasikan pada gambar 2.1



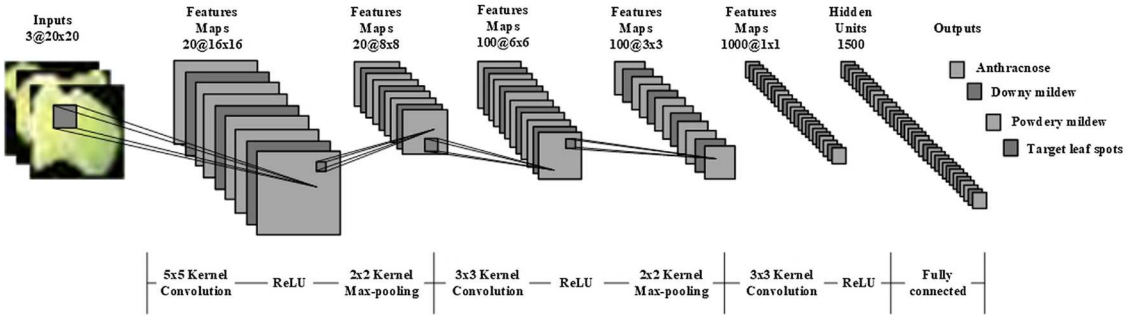
Gambar 2. 1 Proses utama dari CNN (Sumber: Liu dkk, 2015)

Teknologi kunci CNN adalah bidang reseptif lokal, pembagian bobot, *sub sampling* berdasarkan waktu atau ruang, sehingga dapat mengekstrak fitur dan mengurangi ukuran parameter pelatihan. Keuntungan algoritma CNN adalah untuk menghindari ekstraksi fitur eksplisit, dan untuk belajar dari data *training* secara implisit; Bobot neuron yang sama pada permukaan pemetaan fitur, sehingga jaringan dapat belajar secara paralel, mengurangi kompleksitas jaringan; Mengadopsi struktur *sub sampling* berdasarkan waktu atau ruang, dapat mencapai beberapa derajat ketahanan, skala dan perpindahan deformasi; Informasi input dan topologi jaringan bisa sangat cocok, Ini memiliki keunggulan unik dalam pengenalan ucapan dan pemrosesan citra. (Liu dkk, 2015).

2.2.3 Arsitektur CNN

Secara umum, struktur dari CNN terdiri dari dua lapisan. Lapisan pertama adalah lapisan ekstraksi fitur. Input yang ada pada setiap neuron terhubung pada bidang reseptif lokal dari lapisan sebelumnya dan mengekstrak fitur lokalnya. Setelah fitur diekstrak, hubungan posisionalnya dengan fitur lainnya juga

ditentukan. Lapisan kedua adalah lapisan peta fitur. Setiap lapisan komputasi dari CNN terdiri dari sejumlah peta fitur. Setiap peta fitur adalah bidang, berat neuron di bidang tersebut sama. Struktur peta fitur menggunakan fungsi sigmoid sebagai fungsi aktivasi pada CNN, sehingga peta fitur memiliki invariansi pergeseran. Selain itu, karena neuron dalam bidang pemetaan yang sama saling berbagi bobot, jumlah parameter bebas pada CNN menjadi berkurang. Setiap lapisan konvolusi dalam CNN diikuti oleh lapisan komputasi yang digunakan untuk menghitung rata-rata lokal dan ekstrak kedua, struktur ekstraksi dua fitur unik ini mengurangi resolusi. (Liu dkk, 2015). Gambar 2.2 menampilkan contoh arsitektur CNN



Gambar 2. 2 Contoh arsitektur CNN (Sumber: Ma dkk, 2018)

2.2.4 Lapisan Konvolusi

Proses konvolusi merupakan tahapan ekstraksi fitur pada CNN. Dalam proses konvolusi terdapat sebuah *kernel*. *Kernel* adalah filter yang berbentuk matriks yang digunakan untuk mengekstrak fitur yang ada dalam sebuah citra. Adapun rumus yang digunakan untuk melakukan konvolusi terhadap citra dua dimensi adalah sebagai berikut:

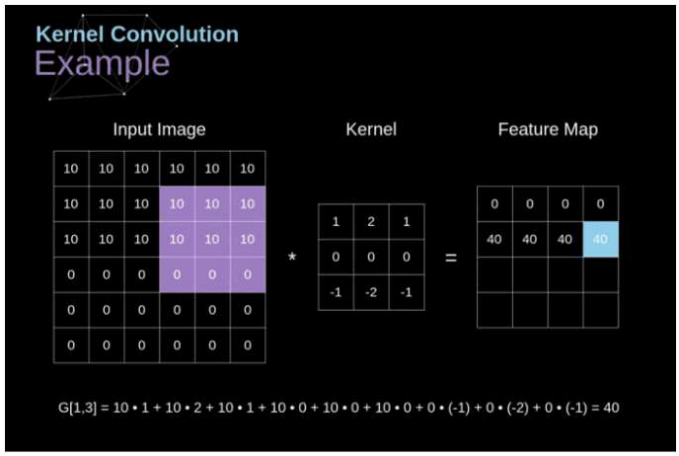
$$G[m,n] = (f * h)[m,n] = \sum_{j}\sum_{k} h[j,k]f[m-j,n-k] \qquad (2.1)$$

Dimana:

f = citra input

h = kernel

m,n = index matriks hasil konvolusi



Gambar 2. 3 konvolusi matriks 2 dimensi (sumber: Skalski, 2019)

Matriks *kernel* bergerak di atas data input dan melakukan operasi *dot product* pada potongan data input yang dilewati oleh *kernel. Kernel* bergerak sesuai dengan jumlah *stride* yang ditentukan. Selanjutnya, hasil perkalian matriks dijumlahkan dan ditempatkan sesuai dengan indeks yang ditentukan dalam *feature map. Output* yang dihasilkan berupa matriks baru yang disebut *feature map* karena berisi fitur-fitur yang ada dalam sebuah citra.

Dalam penelitian ini data input yang digunakan berupa citra RGB atau dengan kata lain citra 3 dimensi, sehingga perlu dilakukan konvolusi untuk citra 3 dimensi. Proses konvolusi matriks 3 dimensi hampir sama dengan konvolusi untuk matriks 2 dimensi. Yang membedakan adalah dimensi kernel yang digunakan dan *feature map* yang dihasilkan sama sama berdimensi 3. Output ini biasa disebut dengan tensor. Adapun rumus yang digunakan untuk melakukan konvolusi terhadap citra tiga dimensi adalah sebagai berikut:

$$[n, n, n_c] * [f, f, n_c] = \left[\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor, \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor, n_f\right] \quad (2.2)$$

Dimana:

n = ukuran citra

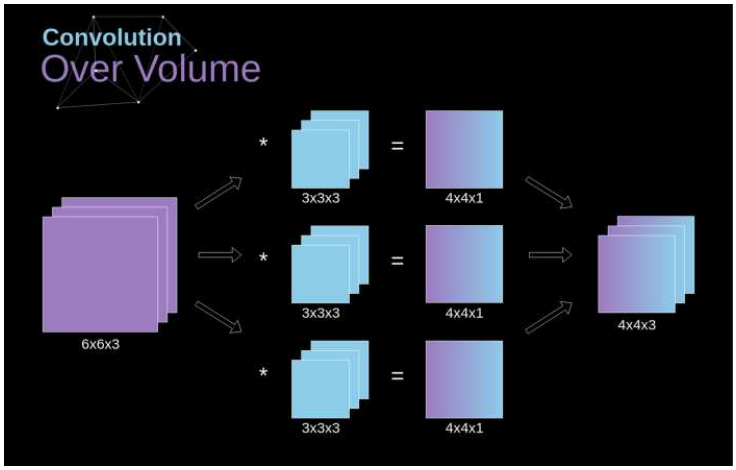
f = ukuran *filter*

n_c = jumlah *channel* pada citra

p = *padding* yang digunakan

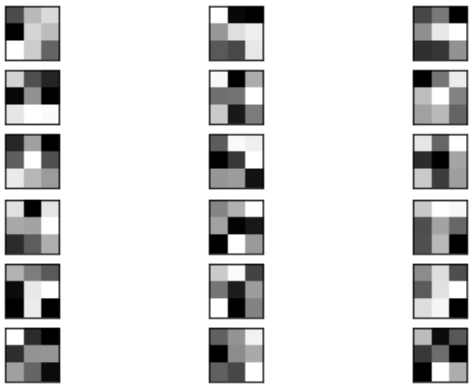
s = *stride* yang digunakan

n_f = jumlah filter



Gambar 2. 4 konvolusi matriks 3 dimensi (sumber: Skalski, 2019)

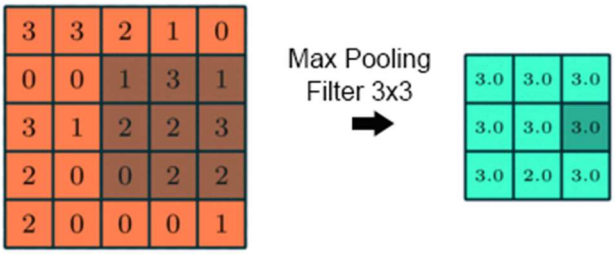
Karena Dalam penelitian ini library yang digunakan adalah Tensorflow, maka dari itu nilai kernel yang akan digunakan tidak perlu diatur oleh peneliti karena Tensorflow akan mengatur secara otomatis nilai dari filter yang akan digunakan dalam proses konvolusi. Berikut adalah contoh kernel yang dibuat secara otomatis oleh Tensorflow:



Gambar 2. 5 Contoh kernel konvolusi

2.2.5 Lapisan *Max Pooling*

Max pooling layer adalah bagian dari ekstraksi fitur pada CNN. *Max pooling* berfungsi untuk mengurangi dimensi dari *feature map* yang akan diproses tanpa meninggalkan informasi yang penting. Hal ini dilakukan agar beban komputasi dapat berkurang. Adapun cara kerja *max pooling layer* adalah dengan mengambil potongan matriks seukuran filter yang sudah ditentukan, kemudian dalam potongan matriks tersebut diambil nilai terbesar. Nilai tersebut kemudian akan menjadi anggota dari matriks *feature map* baru yang dimensinya sudah berkurang. Gambar berikut menampilkan ilustrasi dari proses *max pooling* dengan filter berukuran 3x3.



Gambar 2. 6 Ilustrasi proses *max pooling*

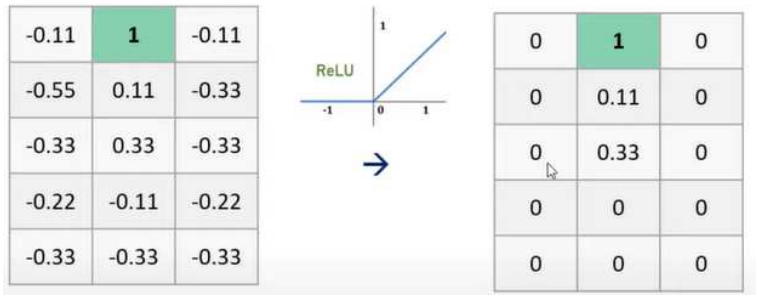
2.2.6 Fungsi Aktivasi ReLU

Selanjutnya setiap nilai yang ada dalam *feature map* ini akan melalui fungsi aktivasi ReLU sebelum menuju lapisan selanjutnya. Cara kerja Fungsi aktivasi ReLU adalah apabila nilai input bernilai negatif maka outputnya berupa 0. Apabila nilai inputnya bernilai positif maka outputnya berupa nilai input itu sendiri. Ilustrasi penggunaan fungsi aktivasi relu ada pada gambar di bawah. Adapun rumus untuk fungsi aktivasi ReLU adalah sebagai berikut:

$$R(z) = \max(0, z) \qquad (2.3)$$

Dimana:

z = nilai input



Gambar 2. 7 Ilustrasi penggunaan fungsi aktivasi ReLU

2.2.7 *Fully Connected Layer*

Kata "*Fully Connected*" menunjukkan bahwa setiap neuron pada setiap neuron pada layer sebelumnya terkoneksi dengan setiap neuron pada layer berikutnya (Primartha, 2018). Tujuan dari *Fully Connected Layer* adalah untuk klasifikasi citra berdasarkan fitur yang telah diekstrak pada *Convolutional layer* dan *Pooling layer*. Fitur-fitur dalam citra digital yang sebelumnya dalam bentuk Matriks akan "diratakan" menjadi bentuk vektor kolom agar dapat menjadi input dari *Fully Connected Layer*. Dalam setiap iterasi *training* akan diaplikasikan

algoritma pembelajaran *Backpropagation.* Fungsi aktivasi yang akan mengklasifikasikan citra sesuai kelasnya adalah fungsi *Softmax.*

2.2.8 Proses *Training* CNN

Proses *training network* pada dasarnya adalah proses untuk memberikan kecerdasan kepada algoritma CNN. Untuk memperoleh kecerdasan maka algoritma memerlukan bahan untuk dipelajari. Bahan belajar disini adalah data latih yang sebelumnya sudah disiapkan. Selama pelatihan berlangsung nilai-nilai *weight* dalam setiap neuron akan diperbaharui sampai ditemukan nilai *weight* yang tepat untuk performa sistem terbaik.

Dalam proses *training* algoritma CNN umumnya digunakan sebuah algoritma optimasi. Algoritma optimasi akan menentukan bagaimana *weight-weight* yang ada dalam CNN akan diperbaharui, sehingga pemilihan algoritma optimasi merupakan hal yang sangat penting. Dalam penelitian ini algoritma optimasi untuk *training* algoritma CNN yang akan digunakan adalah algoritma Adam.

Algoritma Adam merupakan algoritma *stochastic gradient descent* (SGD) yang berdasar pada estimasi adaptif momen orde pertama dan orde kedua. Algoritma Adam menggabungkan properti terbaik dari algoritma AdaGrad (*Adaptive Gradient Algorithm)* dan RMSProp (*Root Mean Square Propagation*) untuk menciptakan algoritma optimasi yang dapat mengatasi masalah gradien dan *noise*. Algoritma Adam cenderung lebih mudah untuk dikonfigurasi karena konfigurasi bawaan dapat bekerja dengan baik untuk kebanyakan kasus.

Proses *training* dalam *deep learning* biasa disebut dengan *backpropagation. Backpropagation* dilakukan dengan menghitung turunan yang akan kemudian digunakan untuk memperbaharui nilai parameter. Proses ini dinamakan *gradient descent.* Dalam *gradient descent* kita akan menilai pengaruh perubahan parameter pada *feature map* yang dihasilkan dan selanjutnya pada hasil akhirnya. Kalkulasi dilakukan dengan menerapkan aturan *chain rule.* Adapun rumus yang digunakan adalah sebagai berikut:

$$dA^{[l]} = \frac{\partial L}{\partial A^{[l]}}\ dZ^{[l]} = \frac{\partial L}{\partial Z^{[l]}}\ dW^{[l]} = \frac{\partial L}{\partial W^{[l]}}\ \ db^{[l]} = \frac{\partial L}{\partial b^{[l]}} \quad (2.4)$$

Dimana:

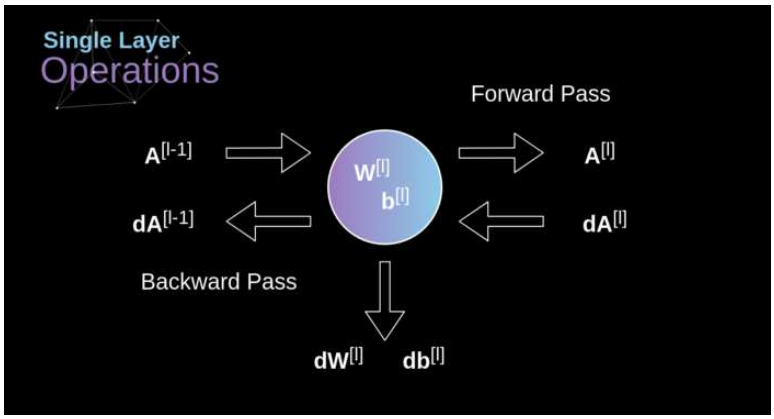
W = *weight (*bobot)

b = bias

L = loss

A = *sum of product*

Z = *total differential*



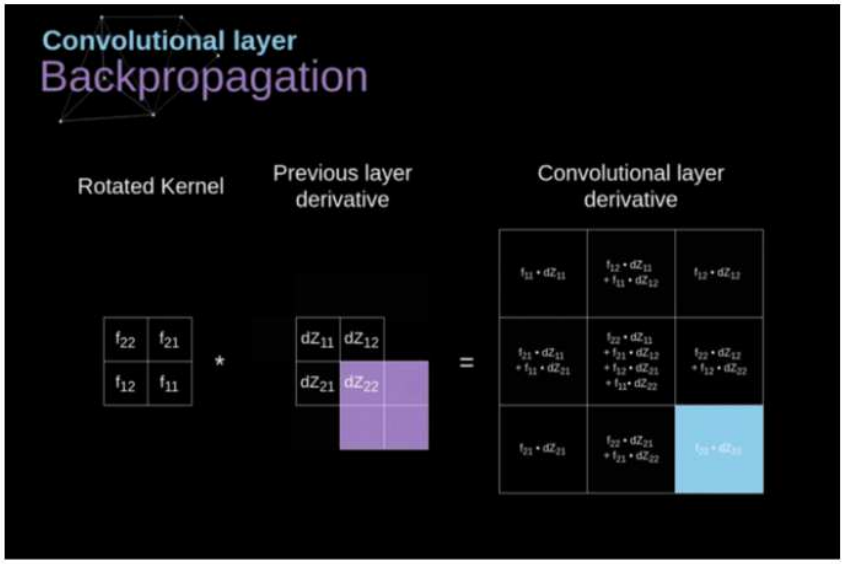
Gambar 2. 8 *Backpropagation* sebuah *neuron* (sumber: Skalski, 2019)

Tugas kita adalah menghitung nilai dW[l] dan db[l] - yang merupakan turunan yang terkait dengan parameter lapisan saat ini, serta nilai dA[l -1] - yang akan

diteruskan ke lapisan sebelumnya. Gambar 3.31 menunjukkan dA[l] sebagai sebuah input. Dimensi tensor dW dan W, db dan b serta dA dan A masing-masing adalah sama. Langkah pertama adalah mendapatkan nilai tengah dZ[l] dengan menerapkan turunan dari fungsi aktivasi kita ke tensor input kita. Menurut aturan rantai, hasil operasi ini akan digunakan nanti. Rumus yang dihasilkan adalah sebagai berikut:

$$dZ^l = dA^{l'} * g'\left(Z^{[l]}\right) \quad (2.5)$$

Selanjutnya adalah melakukan *backpropagation* pada *convolutional layer*. Dalam proses ini digunakan operasi matriks yang disebut konvolusi penuh. Operasi ini diilustrasikan pada gambar berikut.

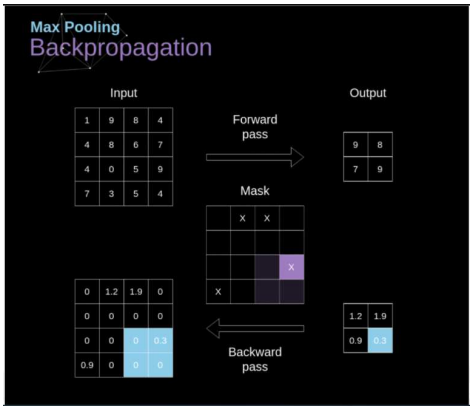


Gambar 2. 9 *Backpropagation convolutional layer* (sumber: Skalski, 2019)

Dalam operasi ini filter dilambangkan dengan W, dan dZ[m,n] adalah skalar yang merupakan turunan parsial yang diperoleh dari lapisan sebelumnya. Dalam proses ini kernel yang digunakan diputar 180 derajat. Rumus untuk proses ini adalah sebagai berikut:

$$dA+= \sum_{m=0}^{n_h} \sum_{n=0}^{n_w} W \cdot dZ[m,n] \quad (2.6)$$

Selanjutnya adalah melakukan proses *backpropagation* pada *max pooling layer*. Dalam lapisan ini tidak ada parameter yang harus diperbaharui. Yang perlu dilakukan hanyalah mendistribusikan gradien dengan tepat. Dalam proses *forward propagation* untuk *max pooling* kita memilih nilai maksimum dari setiap wilayah dan mentransfernya ke lapisan berikutnya. Oleh karena itu jelas bahwa selama *backpropagation*, gradien seharusnya tidak mempengaruhi elemen matriks yang tidak termasuk dalam *forward pass*. Dalam praktiknya, ini dapat dicapai dengan membuat *mask* yang mengingat posisi nilai yang digunakan pada fase pertama, yang nantinya dapat dimanfaatkan untuk mentransfer gradien. Proses ini diilustrasikan dalam gambar berikut.



Gambar 2. 10 *Backpropagation max pooling layer* (sumber: Skalski, 2019)

BAB III

METODE PENELITIAN

3.1 Pengumpulan Data

Data yang diperlukan dalam penelitian ini adalah citra daun tanaman apel. Jenis apel yang diidentifikasi adalah apel manalagi. Data yang digunakan dalam penelitian ini adalah data primer. Data diperoleh melalui survei langsung di kebun apel yang dilakukan oleh peneliti. Pengambilan gambar menggunakan kamera *smartphone*. Berikut adalah detail perangkat yang digunakan:

1) Nama perangkat: Xiaomi Redmi Note 5
2) Spesifikasi kamera: 12 Megapixel (MP), f/2.2, 1/2.9", 1.25µm, PDAF
3) ISO: otomatis
4) *White Balance*: otomatis

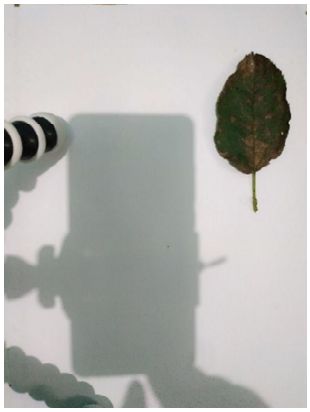
Pengambilan gambar dilakukan dalam tiga kondisi pencahayaan berbeda. Pengukuran tingkat kecerahan menggunakan aplikasi Lux Light Meter Pro yang terhubung dengan sensor cahaya yang ada pada *smartphone.* Berikut adalah jenis pencahayaan yang digunakan:

1) Pencahayaan 1: Dalam ruangan dengan cahaya lampu kamar 15W (100-200 lux).
2) Pencahayaan 2: Dalam ruangan dengan cahaya lampu LED (2000-3000 lux).
3) Pencahayaan 3: Luar ruangan dengan cahaya matahari (lebih dari 30000 lux).

3.1.1 Persiapan Citra Input

Berikut adalah tahapan persiapan citra sampai citra siap digunakan:

1) Daun apel difoto menggunakan kamera *smartphone* dengan bantuan *tripod.* Resolusi awal citra adalah 3000x4000 piksel.



Gambar 3. 1 Citra daun sebelum di-*crop*

2) Hasil foto kemudian di-*crop* pada bagian yang terdapat daun apel. Resolusi citra setelah di-crop beragam mulai 2000x2000 sampai 500x500 piksel seperti gambar di bawah:

Gambar 3. 2 Citra daun setelah di-*crop*

3) Ukuran citra diperkecil menjadi 60x60 piksel untuk menjadi input algoritma CNN.

Gambar 3. 3 Citra daun setelah diperkecil

Dalam penelitian ini citra daun apel dibagi menjadi tiga pencahayaan yang berbeda. Berikut adalah tampilan daun apel dengan tiga jenis pencahayaan berbeda untuk setiap jenis data:

Tabel 3. 1 Perbandingan citra daun

No.	Jenis data	Lampu Kamar	Lampu LED	Cahaya Matahari
1	Apel Sehat			
2	Bercak Daun			
3	Busuk Buah			
4	Cacar Daun			

3.1.2 Jenis Citra Input

Total data yang telah dikumpulkan peneliti berjumlah 1517 file citra RGB dengan format JPG. Dalam penelitian ini terdapat empat jenis atau kelas citra input yaitu bercak daun, cacar daun, busuk buah beserta apel yang sehat. Penyakit dipisahkan berdasarkan ciri-ciri yang ada pada daun. Ciri-ciri dari setiap penyakit diperoleh peneliti melalui wawancara dengan petani apel setempat. Berikut adalah penjelasan dari setiap jenis data yang digunakan dalam penelitian ini:

1) Apel Sehat

Daun apel sehat memiliki ciri-ciri yaitu pada permukaan daun bersih tidak terdapat bintik hitam atau putih. Daun berwarna hijau segar dengan permukaan cenderung mengkilap. Dalam penelitian ini terdapat 200 citra daun apel sehat pada pencahayaan dengan lampu kamar. 200 citra pada pencahayaan dengan lampu LED. 202 citra pada pencahayaan dengan cahaya matahari.

Gambar 3. 4 Daun sehat

2) Bercak Daun

Daun apel dengan penyakit bercak daun memiliki ciri-ciri yaitu pada permukaan daun terdapat bintik-bintik berwarna hitam. Dalam penelitian ini terdapat 88 citra daun apel dengan penyakit bercak daun pada pencahayaan

dengan lampu kamar. 90 citra pada pencahayaan dengan lampu LED. 77 citra pada pencahayaan dengan cahaya matahari.

Gambar 3. 5 Daun dengan penyakit bercak daun

3) Cacar Daun

Daun apel dengan penyakit cacar daun memiliki ciri-ciri yaitu pada permukaan daun terdapat bintik-bintik berwarna putih. Dalam penelitian ini terdapat 80 citra daun apel dengan penyakit cacar daun pada pencahayaan dengan lampu kamar. 80 citra pada pencahayaan dengan lampu LED. 92 citra pada pencahayaan dengan cahaya matahari.

Gambar 3. 6 Daun dengan penyakit cacar daun

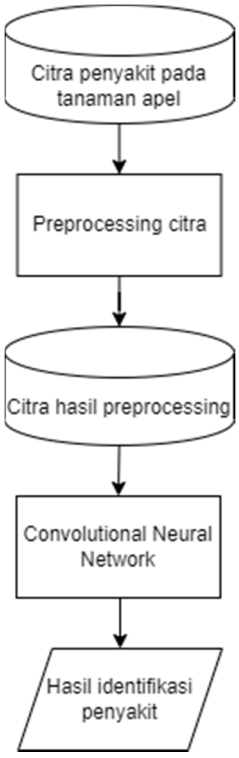
4) Busuk Buah

Daun apel dengan penyakit busuk buah memiliki ciri-ciri yaitu sebagian permukaan daun menjadi berwarna kuning atau menjadi kehitaman. Dalam penelitian ini terdapat 135 citra daun apel dengan penyakit busuk buah pada pencahayaan dengan lampu kamar. 135 citra pada pencahayaan dengan lampu LED. 138 citra pada pencahayaan dengan cahaya matahari.

Gambar 3. 7 Daun dengan penyakit busuk buah

3.2 Desain Sistem

Desain sistem dalam penelitian ini terdiri dari citra penyakit pada tanaman apel, *preprocessing* citra, citra hasil *preprocessing,* algoritma CNN dan hasil identifikasi penyakit. Berikut adalah desain sistem untuk penelitian ini.



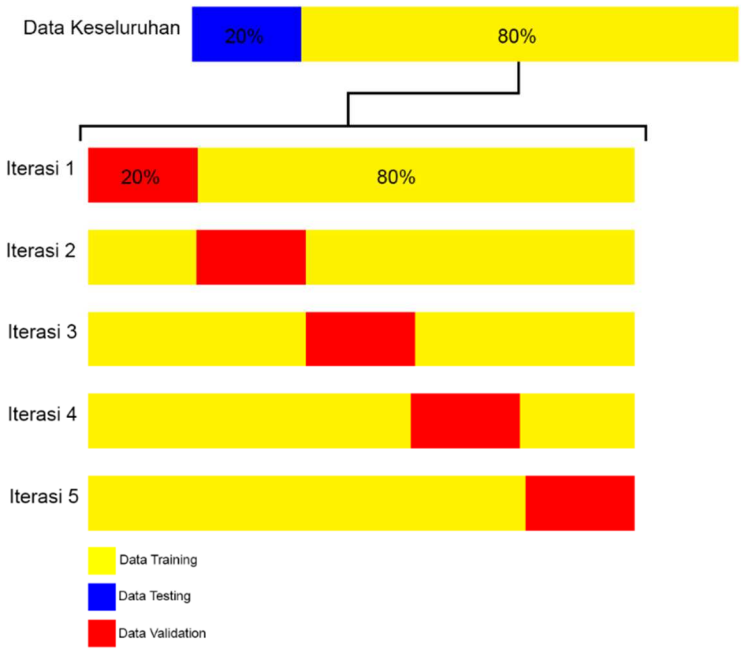
Gambar 3. 8 Desain sistem

3.2.1 Input Citra

Tahap pertama adalah menyiapkan data yang akan menjadi input pada sistem. Data yang sudah dikumpulkan pada sub bab 3.1 adalah data yang akan

menjadi input pada sistem yang akan dibuat. Data ini akan dibagi menjadi tiga yaitu data *training* , data *validation* dan data *testing*. Sebelumnya data sudah dibagi secara manual menjadi 5 *fold*er sesuai jumlah *fold* yang ditentukan. Data kemudian disimpan dalam format .rar untuk mempermudah proses ekstraksi data. Untuk ekstraksi file dengan format .rar, Python membutuhkan *package* seperti pyunpack dan patool. Selanjutnya setiap citra input diberikan label sesuai *class*-nya. Setelah itu citra akan dinormalisasi dengan cara membagi nilai setiap piksel dengan angka 255.

Dalam penelitian ini metode pengujian yang digunakan adalah K-*fold cross validation* dengan K=5. Maka dari itu, perbandingan data latih dan data uji adalah 80:20. Data *training* yang sudah dipisahkan dari data *testing* kemudian dibagi lagi menjadi data *validation* dan data *training*. Data validasi berjumlah 20% dari jumlah data latih. Berikut adalah diagram pembagian data untuk penelitian ini.



Gambar 3. 9 Diagram pembagian data

Pada data dengan pencahayaan lampu kamar terdapat 503 citra. *Class* apel sehat terdiri dari 128 data *training,* 40 data *test,* 32 data *validation. Class* bercak daun terdiri dari 57 data *training,* 17 data *test,* 14 data *validation. Class* busuk buah terdiri dari 87 data *training,* 27 data *test,* 21 data *validation. Class* cacar daun terdiri dari 52 data *training,* 16 data *test,* 12 data *validation.* Total pembagian data pada pencahayaan ini adalah 324 data *training,* 100 data *test,* 79 data *validation.*

Pada data dengan pencahayaan lampu LED terdapat 505 citra. *Class* apel sehat terdiri dari 128 data *training,* 40 data *test,* 32 data *validation. Class* bercak daun terdiri dari 58 data *training,* 18 data *test,* 14 data *validation. Class* busuk buah terdiri dari 87 data *training,* 27 data *test,* 21 data *validation. Class* cacar daun terdiri dari 52 data *training,* 16 data *test,* 12 data *validation.* Total pembagian data pada pencahayaan ini adalah 324 data *training,* 100 data *test,* 79 data *validation.*

Pada data dengan pencahayaan cahaya matahari terdapat 509 citra. *Class* apel sehat terdiri dari 130 data *training,* 40 data *test,* 32 data *validation. Class* bercak daun terdiri dari 49 data *training,* 15 data *test,* 13 data *validation. Class* busuk buah terdiri dari 89 data *training,* 29 data *test,* 22 data *validation. Class* cacar daun terdiri dari 60 data *training,* 18 data *test,* 14 data *validation.* Total pembagian data pada pencahayaan ini adalah 324 data *training,* 100 data *test,* 79 data *validation.*

3.2.2 *Image Preprocessing*

Untuk memperoleh performa yang optimal, algoritma CNN memerlukan data dalam jumlah yang besar. Salah satu cara untuk menambah jumlah data adalah dengan teknik augmentasi data. Dengan augmentasi data, data dapat diperbanyak tanpa merusak data tersebut. Dalam penelitian ini teknik augmentasi data yang akan digunakan adalah *rotation, width shift, height shift, shear, zoom, horizontal flip.* Berikut adalah penjelasan dari setiap teknik augmentasi data yang digunakan:

a. Rotation

Teknik ini akan melakukan rotasi pada citra. Derajat rotasi ditentukan secara acak sesuai dengan rentang nilai (*range)* derajat yang ditentukan. Dalam penelitian digunakan rotation_range = 25. Berikut adalah contoh penerapan dari teknik *rotation:*

Gambar 3. 10 Hasil augmentasi dengan teknik *rotation*

b. Width Shift

Teknik ini akan melakukan pergeseran pada citra secara horizontal atau dari kanan ke kiri. Rentang nilai (*range)* pergeseran ditentukan berdasarkan

persentase dari total lebar citra. Dalam penelitian ini citra input berukuran 60 x 60 piksel dan width_shift_range = *0.1*. Artinya pergeseran akan dilakukan -10% sampai +10% dari lebar gambar atau -6px sampai +6px. Citra akan digeser secara acak sesuai dengan rentang nilai tersebut. Apabila nilai yang terpilih bernilai positif maka citra akan bergeser ke kanan dan apabila nilai yang terpilih bernilai negatif maka citra akan bergeser ke kiri. Berikut adalah contoh penerapan dari teknik *width shift:*

Gambar 3. 11 Hasil augmentasi dengan teknik *width shift*

c. Height Shift

Teknik ini akan melakukan pergeseran pada citra secara vertikal atau dari atas ke bawah. Rentang nilai (*range)* pergeseran ditentukan berdasarkan persentase dari total tinggi citra. Dalam penelitian ini citra input berukuran 60 x 60 piksel dan width_shift_range = *0.1*. Artinya pergeseran akan dilakukan -10% sampai +10% dari tinggi gambar atau -6px sampai +6px.. Citra akan digeser secara acak sesuai dengan rentang nilai tersebut. Apabila nilai yang terpilih bernilai positif maka citra akan bergeser ke atas dan apabila nilai yang terpilih bernilai negatif maka citra akan bergeser ke bawah. Berikut adalah contoh penerapan dari teknik *height shift:*

Gambar 3. 12 Hasil augmentasi dengan teknik *height shift*

d. Shear

Teknik ini akan memiringkan bentuk dari citra. Dalam penelitian ini digunakan intensitas kemiringan 0.2 atau shear_range = *0.2*. Berikut adalah contoh penerapan dari teknik *shear:*

Gambar 3. 13 Hasil augmentasi dengan teknik *shear*

e. Zoom

Teknik ini akan memperbesar citra secara acak sesuai dengan rentang nilai (*range)* yang ditentukan. Dalam penelitian ini digunakan zoom_range = 0.2 yang artinya citra akan diperbesar 0% sampai 20%. Berikut adalah contoh penerapan dari teknik *zoom:*

Gambar 3. 14 Hasil augmentasi dengan teknik *zoom*

f. Horizontal Flip

Teknik ini akan membalikkan citra secara horizontal. citra yang akan dibalik akan dipilih secara acak. Berikut adalah contoh penerapan dari teknik *horizontal flip:*

Gambar 3. 15 Hasil augmentasi dengan teknik horizontal *flip*

3.2.3 Implementasi CNN

Dalam penelitian ini tahapan implementasi CNN akan dilakukan dalam bahasa pemrograman Python versi 3.7.12 dengan *flatform* TensorFlow versi 2.7.0. *framework* yang digunakan adalah Keras versi 2.7.0. Proses *training* dan *testing* akan dilakukan pada aplikasi Google Colaboratory. Google Colaboratory adalah aplikasi web yang dikembangkan oleh Google untuk keperluan *machine learning, data science* dan analisis data. Dengan Google Colaboratory kita dapat menjalankan

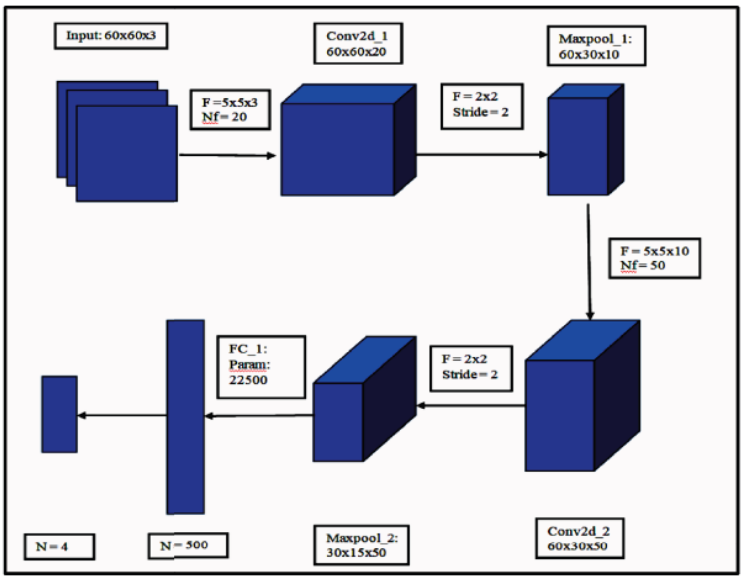
kode yang ditulis dalam bahasa Python melalui browser dan menggunakan tenaga komputasi *cloud* yang disediakan oleh Google sehingga tidak diperlukan komputer lokal dengan tenaga komputasi tinggi dan hanya memerlukan koneksi internet yang baik.

Dalam implementasi CNN terdapat beberapa komponen dan tahapan yang perlu diperhatikan. Ini meliputi: lapisan konvolusi, fungsi aktivasi ReLU, lapisan *max pooling,* arsitektur CNN, ekstraksi fitur pada CNN, identifikasi pada CNN dan proses training CNN.

3.2.3.1 Arsitektur CNN

Arsitektur CNN yang digunakan dalam penelitian ini adalah arsitektur yang digunakan dalam penelitian Baranwal dkk (2019). Arsitektur ini memiliki pola yang sama dengan arsitektur LeNet-5. Arsitektur ini dipilih karena memiliki jumlah parameter yang lebih sedikit jika dibandingkan dengan arsitektur popular seperti AlexNet (Baranwal dkk, 2019).

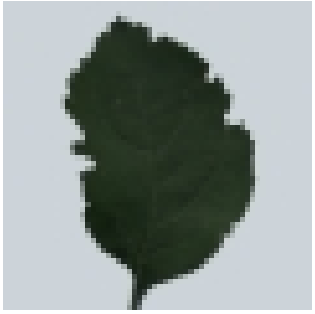
Arsitektur ini terdiri dari 7 *layer* atau lapisan. Lapisan-lapisan ini antara lain 2 *convolutional layer,* 2 *max pooling layer,* 1 *flatten layer,* 1 *dense layer,* dan 1 *output layer.* Arsitektur CNN terbagi menjadi 2 bagian yaitu bagian ekstraksi fitur dan bagian identifikasi. Bagian Identifikasi dilakukan oleh *convolutional layer* dan *max pooling layer.* Sementara bagian klasifikasi lakukan oleh *flatten layer, dense layer,* dan *output layer.* Ketiga lapisan ini biasa disebut dengan *fully connected layer.* Berikut adalah arsitektur CNN yang digunakan dalam penelitian ini:



Gambar 3. 16 Arsitektur CNN (sumber: Baranwal dkk, 2019)

3.2.3.2 Ekstraksi Fitur Pada CNN

Sebelum Masuk ke proses ekstraksi fitur CNN, pertama-tama dilakukan inisialisasi model CNN yang akan dibuat. Model yang digunakan adalah model Sequential karena setiap layer akan dijalankan secara linear atau secara berurutan. Gambar di bawah merupakan salah satu citra yang akan menjadi citra input. Citra ini memiliki label "apel_sehat (1).jpg". Citra ini berukuran 60x60x3 piksel. Citra ini akan digunakan sebagai contoh untuk menampilkan proses ekstraksi fitur pada CNN.



Gambar 3. 17 Contoh citra input

Tabel 3.2 akan menampilkan representasi gambar di atas dalam bentuk matriks. Karena keterbatasan tempat maka hanya akan ditampilkan 15 baris dan kolom pertama pada *channel* warna merah. Perhitungan tetap dilakukan dengan melibatkan seluruh anggota matriks

Tabel 3. 2 Representasi citra input dalam bentuk matriks

151	152	153	153	150	150	150	150	151	150	153	151	154	154	156
153	151	152	153	150	150	149	150	151	151	151	153	153	155	155
151	149	150	150	152	150	149	150	149	150	151	151	151	152	153
150	148	151	150	149	150	149	151	150	149	150	151	150	151	152
149	149	149	150	150	150	150	149	149	150	150	152	151	151	151
150	148	149	150	151	150	150	149	150	150	150	151	152	152	155
150	149	149	148	148	149	148	149	150	151	150	151	154	155	154
150	150	150	149	149	151	150	150	150	151	150	150	152	155	155
148	148	149	150	149	150	151	150	150	149	151	151	152	154	154
149	149	149	150	150	151	150	151	151	151	151	151	152	153	155
148	150	151	151	150	151	152	151	152	152	151	151	152	152	154
149	149	151	151	150	151	153	153	154	153	153	153	154	156	157
150	151	151	152	154	155	154	154	155	156	156	153	153	156	157
150	151	151	150	153	154	154	153	156	157	155	154	154	157	156
152	154	153	152	154	155	154	154	155	156	154	154	153	157	157

Sebelum dilakukan perhitungan, dilakukan normalisasi terhadap citra input terlebih dahulu. Normalisasi dilakukan dengan membagi setiap nilai piksel dengan 255. Berikut adalah citra input setelah dilakukan normalisasi

Tabel 3. 3 Citra input setelah normalisasi

0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59

1. *Convolutional layer* 1

Tahap pertama adalah citra input akan melalui *convolutional layer* 1. Lapisan ini memiliki kernel berukuran 5x5 dengan stride=1 dan filter berjumlah 20 buah. Input pada lapisan ini adalah citra input berukuran 60x60x3. Jenis padding yang digunakan adalah ‘same’ artinya dimensi tinggi dan lebar output akan sama dengan dimensi input. Hal ini dilakukan dengan menambahkan angka 0 pada baris dan kolom baru di sekeliling citra. Berikut adalah matriks citra setelah proses padding.

Tabel 3. 4 Matriks input setelah proses *padding*

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59

Sebagai contoh, berikut adalah kernel yang digunakan pada lapisan ini

Tabel 3. 5 Contoh filter untuk *convolutional layer 1*

-0.1305493600	0.0512389700	0.0077721600
0.0365313000	-0.0629468600	0.0986095700
-0.0515038200	0.0329678400	0.0965579800

Berikut adalah contoh perhitungan konvolusi untuk satu elemen matriks yang melewati lapisan ini:

G[1,1] = (0 * -0.1305493600) + (0 * 0.0512389700) + (0 * 0.0512389700) + (0 * 0.0365313000) + (0.59 * -0.0629468600) + (0.59 * 0.0986095700) + (0 * -0.0515038200) + (0.59 * 0.0329678400) + (0.59 * 0.0965579800)

G[1,1] = 0.10

Berikut adalah hasil konvolusi secara keseluruhan:

Tabel 3. 6 Matriks output setelah proses konvolusi 1

0.10	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Selanjutnya matriks akan melewati fungsi aktivasi ReLU

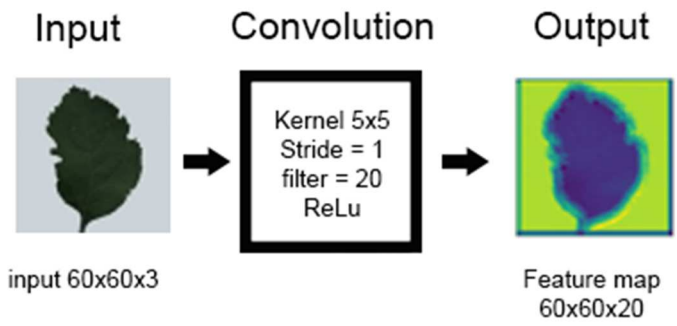
R(z) = max(0, 0.10) = 10

Berikut adalah matriks citra setelah melalui fungsi aktivasi ReLU :

Tabel 3. 7 Matriks output setelah melewati fungsi aktivasi ReLU

0.10	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Dalam lapisan ini output yang dihasilkan berupa *feature map* berukuran 60x60x20. Gambar di bawah Menampilkan proses konvolusi pada lapisan ini. Output *Feature map* lapisan ini selanjutnya akan menjadi input untuk lapisan *max pooling layer* 1.



Gambar 3. 18 Hasil konvolusi pada lapisan konvolusi 1

2. *Max pooling layer* 1

Lapisan selanjutnya adalah *max pooling layer* 1. Lapisan ini memiliki filter berukuran 2x2 dengan stride=2. Input lapisan ini adalah *feature map* berukuran 60x60x20 yang dihasilkan oleh lapisan sebelumnya. Berikut adalah contoh proses yang terjadi dalam lapisan *max pooling*

Tabel 3. 8 Proses max pooling

0.10	0.09	0.09	0.09	0.09	0.09
0.13	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05

G[1,1] = max(0.10, 0.90, 0.09, 0.13, 0.05, 0.05, 0.13, 0.05, 0.05)

G[1,1] = 0.13

G[1,2] = max(0.09, 0.09, 0.09, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05)

G[1,2] = 0.09

G[2,1] = max(0.13, 0.05, 0.05, 0.13, 0.05, 0.05, 0.13, 0.05, 0.05)

G[2,1] = 0.13

G[2,2] = max(0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05)

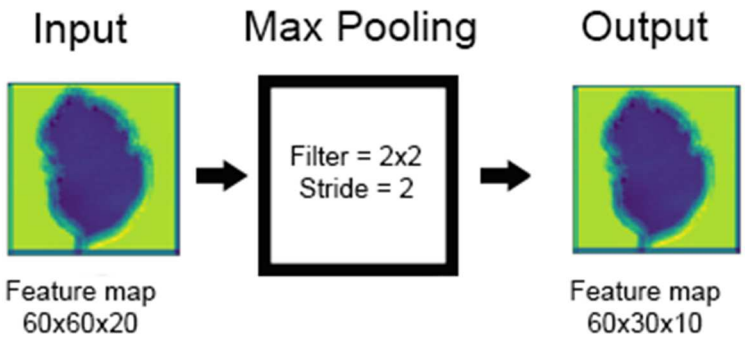
G[2,2] = 0.05

Secara keseluruhan, berikut adalah hasil yang diperoleh dari lapisan *max pooling* 1

Tabel 3. 9 Matriks output setelah proses max pooling 1

0.13	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Output yang dihasilkan lapisan ini berupa *feature map* berukuran 60x30x10. Gambar di bawah Menampilkan proses *pooling* pada lapisan ini. Selanjutnya *feature map* ini akan menjadi input untuk *convolutional layer* 2.



Gambar 3. 19 Output *Feature map* dari *max pooling layer* 1

3. *Convolutional layer* 2

Lapisan selanjutnya adalah *convolutional layer* 2. Lapisan ini memiliki kernel berukuran 5x5 dengan stride=1 dan filter berjumlah 50 buah. Input lapisan ini adalah *feature map* berukuran 60x30x10 yang dihasilkan oleh lapisan sebelumnya. Sebelum dilakukan konvolusi, *feature map* akan diberikan *padding* terlebih dahulu. Berikut adalah matriks citra setelah proses padding.

Tabel 3. 10 Matriks input setelah proses padding

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0.13	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0	0.13	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Sebagai contoh, berikut adalah kernel yang digunakan pada lapisan ini

Tabel 3. 11 Contoh filter untuk convolutional layer 2

-0.1155515800	0.0588773800	0.0421962400
-0.1181599100	-0.1125107000	-0.0538383200
-0.0085613300	0.0767164400	0.1069133400

Berikut adalah contoh perhitungan konvolusi untuk satu elemen matriks yang melewati lapisan ini:

G[1,1] = (0 * -0.1155515800) + (0 * 0.0588773800) + (0 * 0.0421962400) + (0 * -0.1181599100) + (0.13 * -0.1125107000) + (0.06 * -0.0538383200) + (0 * -0.0085613300) + (0.13 * 0.0767164400) + (0.05 * 0.1069133400)

G[1,1] = -1.14

Berikut adalah hasil konvolusi secara keseluruhan:

Tabel 3. 12 Matriks output setelah proses konvolusi 2

0.00	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02
0.01	-0.02	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
0.01	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01

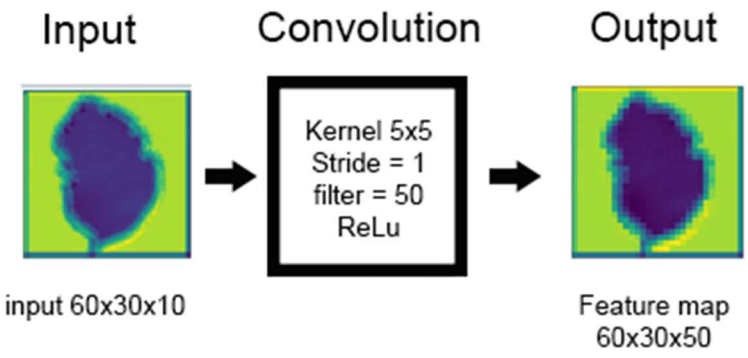
Selanjutnya matriks akan melewati fungsi aktivasi ReLU

Berikut adalah matriks citra setelah melalui fungsi aktivasi ReLU :

Tabel 3. 13 Matriks output setelah melewati fungsi aktivasi ReLU

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Dalam lapisan ini output yang dihasilkan berupa *feature map* berukuran 60x30x50. Gambar di bawah Menampilkan proses konvolusi pada lapisan ini. Output *Feature map* lapisan ini selanjutnya akan menjadi input untuk lapisan *max pooling layer* 2.



Gambar 3. 20 Hasil konvolusi pada lapisan konvolusi 2

4. *Max pooling layer* 2

Lapisan selanjutnya adalah *max pooling layer* 2. Lapisan ini memiliki filter berukuran 2x2 dengan stride=2. Input lapisan ini adalah *feature map* berukuran 60x30x50 yang dihasilkan oleh lapisan sebelumnya. Berikut adalah contoh proses yang terjadi dalam lapisan *max pooling*

Tabel 3. 14 Proses max pooling 2

0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00

G[1,1] = max(0, 0, 0, 0.01, 0, 0, 0.01, 0, 0)

G[1,1] = 0.01

G[1,2] = max(0, 0, 0, 0, 0, 0, 0, 0 , 0, 0)

G[1,2] = 0

G[2,1] = max(0.01, 0, 0, 0.01, 0, 0, 0.01, 0, 0)

G[2,1] = 2.02

G[2,2] = max(0, 0, 0, 0, 0, 0, 0, 0 , 0, 0)

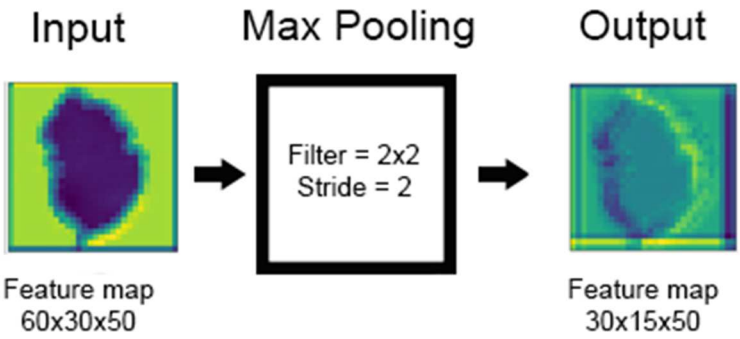
G[2,2] = 0

Secara keseluruhan, berikut adalah hasil yang diperoleh dari lapisan *max pooling* 2

Tabel 3. 15 Matriks output setelah proses max pooling 2

0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Output yang dihasilkan lapisan ini berupa *feature map* berukuran 30x15x50. Gambar di bawah Menampilkan proses *pooling* pada lapisan ini.

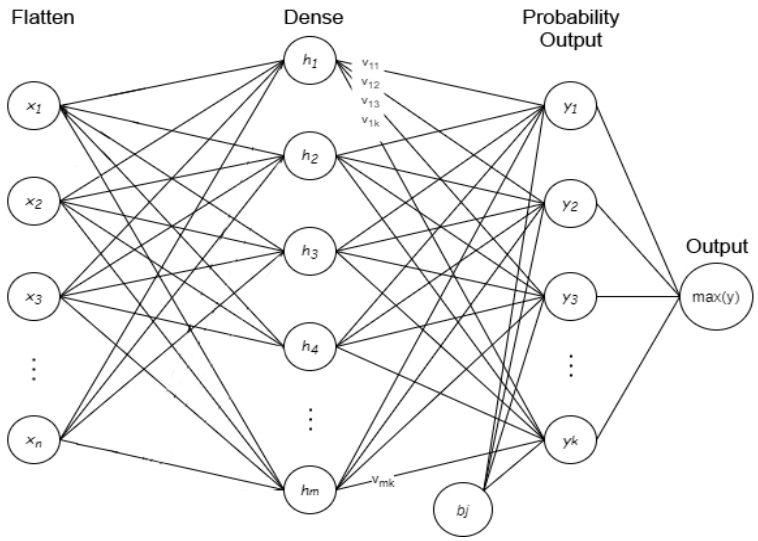


Gambar 3. 21 Output *Feature map* dari *max pooling layer* 2

Lapisan ini merupakan lapisan terakhir dalam tahap ekstraksi fitur milik CNN, tahap selanjutnya adalah tahap klasifikasi yang dilakukan oleh *fully connected layer.*

3.2.3.3 Identifikasi Pada CNN

Arsitektur *Neural Network fully connected layer* untuk klasifikasi pada CNN ditampilkan pada gambar berikut.



Gambar 3. 22 Arsitektur NN

Arsitektur NN pada penelitian ini terdiri dari *flatten layer*, *dense layer*, dan *output layer.* Lapisan pertama adalah *flatten layer. Flatten layer* akan mengubah *feature map* berukuran 30x15x50 menjadi vektor kolom berukuran 1x22500. Vektor ini kemudian akan menjadi input untuk *dense layer 1*. *Dense layer 1* terdiri dari 500 *neurons* dengan fungsi aktivasi ReLU. Output layer terdiri dari 4 *neurons* dengan fungsi aktivasi Softmax.

Setiap *node* dalam *flatten layer* akan terhubung dengan node yang ada pada *dense layer* 1. Setiap *node* yang ada pada *dense layer* terdapat bobot (*weight)* yang disimbolkan dengan *h.* Selanjutnya nilai input akan dikalikan dengan nilai bobot

pada sebuah *node*. Hasil perkalian ini selanjutnya dijumlahkan dengan nilai bias dan kemudian dimasukkan ke dalam fungsi aktivasi *Rectified Linear Unit* (ReLU).

Setelah operasi ini dilakukan, hasilnya akan dimasukkan ke dalam fungsi *softmax*. Fungsi aktivasi ini akan memberikan vektor probabilitas dari 4 kemungkinan hasil identifikasi. Nilai tertinggi dalam vektor probabilitas ini kemudian akan menjadi output dari klasifikasi yang dilakukan. Berdasarkan penjelasan di atas, berikut adalah rumus-rumus yang digunakan:

$$h_m = ReLU\left(\sum_{i=1}^{n} x_i . w_i + b_i\right) \quad (3.1)$$

$$yk = Softmax\left(\sum_{j=1}^{n} h_j . v_j + b_j\right) \quad (3.2)$$

Di mana:

x_i : *Neuron* pada *input layer*

w_i : *Weight*/bobot pada *input layer*

b_i : *Bias* pada *input layer*

h_j : *Neuron* pada *hidden layer*

v_j : *Weight*/bobot pada *hidden layer*

b_j : *Bias* pada *hidden layer*

yk : *Neuron* pada *output layer*

ReLU : Fungsi aktivasi ReLU

Softmax : Fungsi Aktivasi Softmax

Berikut adalah rumus dari fungsi aktivasi Softmax:

$$Softmax(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \quad (3.3)$$

Berdasarkan penjelasan di atas, berikut adalah rangkuman arsitektur CNN yang akan dibuat:

Tabel 3. 16 Rangkuman Arsitektur CNN

Karakteristik	**Spesifikasi**
Dimensi citra *input*	60 x 60 x 3
Jumlah *convolution layer*	2
Jumlah *pooling layer*	2
Jumlah *hidden layer*	2
Fungsi aktivasi	- ReLU (Digunakan pada *convolution layer* dan *hidden layer*) - Softmax (Digunakan pada *output layer)*
Learning rate	0.001
Jumlah *Output*	1

3.2.3.4 Proses *Training* CNN

Dalam penelitian ini *loss function* yang digunakan adalah *categorical cross-entropy*. *Loss function* ini digunakan karena klasifikasi yang dilakukan dalam penelitian ini bersifat *multi-label classification. Pada Loss function* ini, komponen vektor atau class output bersifat independen. Artinya, nilai loss yang diperoleh setiap vektor output CNN tidak dipengaruhi oleh komponen lain. Dalam proses *training* pada penelitian ini *parameter learning* yang digunakan yaitu: *learning rate* = 0.001, *epoch* = 200, *batch size* = 8.

BAB IV

UJI DAN PEMBAHASAN

Pada Bab ini akan dijelaskan tahapan uji coba beserta pembahasannya berdasarkan tujuan penelitian yang ada pada pada Bab 1. Kemudian hasil dari penelitian akan dipaparkan sesuai dengan tahapan uji coba yang ditentukan, yaitu menghitung akurasi, sensitivitas, dan spesifisitas.

4.1 Skenario Uji Coba

Metode pengujian yang akan digunakan dalam penelitian ini adalah *K-Fold Cross Validation*. Cara kerja metode ini adalah dengan membagi data menjadi beberapa grup sesuai dengan nilai K yang ditentukan. Setiap grup terdiri dari data latih, data validasi dan data uji. Dalam penelitian ini nilai K yang digunakan adalah K=5.

Selanjutnya dilakukan penentuan *ground truth* atau kebenaran mutlak yang akan dijadikan sebagai acuan untuk membandingkan hasil prediksi yang dilakukan oleh sistem. Penentuan *ground truth* dilakukan dengan wawancara terhadap petani apel setempat. Peneliti memperlihatkan daun apel kepada petani apel kemudian petani apel menentukan daun tersebut termasuk dalam *class* yang mana. Setelah itu dilakukan pengambilan gambar daun. File citra akan diberikan label sesuai dengan kelasnya masing-masing. Kemudian file citra akan dipisahkan dalam folder yang diberi nama sesuai dengan kelasnya. Berdasarkan penjelasan di atas, maka nama file dan nama folder kelasnya akan dijadikan sebagai *ground truth* untuk pengujian pada penelitian ini.

Berdasarkan penjelasan di atas, maka langkah-langkah dalam melakukan uji coba identifikasi penyakit pada tanaman apel adalah sebagai berikut:

1. Menyiapkan data file citra beserta labelnya masing-masing sesuai dengan kelasnya. Kelas dalam penelitian ini berjumlah empat kelas yaitu: bercak daun, cacar daun, busuk buah beserta apel sehat. Data ini sebelumnya sudah dipisahkan menjadi data *training,* data *validation,* dan data *testing*.
2. Data *training* selanjutnya akan melalui proses augmentasi data sebelum masuk ke dalam tahap *training*.
3. Selanjutnya tahap *training, validation* dan *testing* akan dilakukan sebanyak 5 *fold*.
4. Dalam setiap *fold*, performa sistem akan dihitung sesuai dengan metrik yang sudah ditentukan. Dalam penelitian ini performa sistem yang diukur adalah nilai akurasi, sensitivitas, dan spesifisitas. Sistem tidak langsung memberikan nilai dari metrik tersebut, melainkan akan memberikan nilai-nilai seperti: *true positives* (TP)*, true negatives* (TN)*, false positives* (FP)*, false negatives* (FN). Selanjutnya nilai-nilai ini akan digunakan untuk mendapatkan nilai akurasi, sensitivitas, dan spesifisitas melalui perhitungan sesuai rumus yang ada. Berikut adalah rumus-rumus yang digunakan:

$$\text{Akurasi} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100\% \quad (4.1)$$

$$\text{Sensitivitas} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\% \quad (4.2)$$

$$\text{Spesifisitas} = \frac{\text{TN}}{\text{TN} + \text{FP}} \times 100\% \quad (4.3)$$

Dimana:

TP: Citra dengan output penyakit i, benar diprediksi oleh sistem masuk ke penyakit I

TN: Citra dengan output bukan penyakit i, benar diprediksi oleh sistem tidak masuk ke penyakit I

FP: Citra dengan output bukan bukan penyakit i, ternyata sistem memprediksi masuk ke penyakit I

FN: Citra dengan output penyakit i, ternyata sistem memprediksi tidak masuk ke penyakit I

5. Setelah 5 *fold* selesai, hasil pengujian dari setiap *fold* akan dikumpulkan kemudian dihitung nilai rata-rata akurasi, sensitivitas dan spesifisitas dari setiap *fold* yang dilakukan. Berikut adalah rumus yang digunakan dalam perhitungan ini:

$$\overline{\text{Akurasi}} = \frac{\sum f_i x_i}{\sum f_i} \quad (4.4)$$

$$\overline{\text{Sensitivitas}} = \frac{\sum f_i x_i}{\sum f_i} \quad (4.5)$$

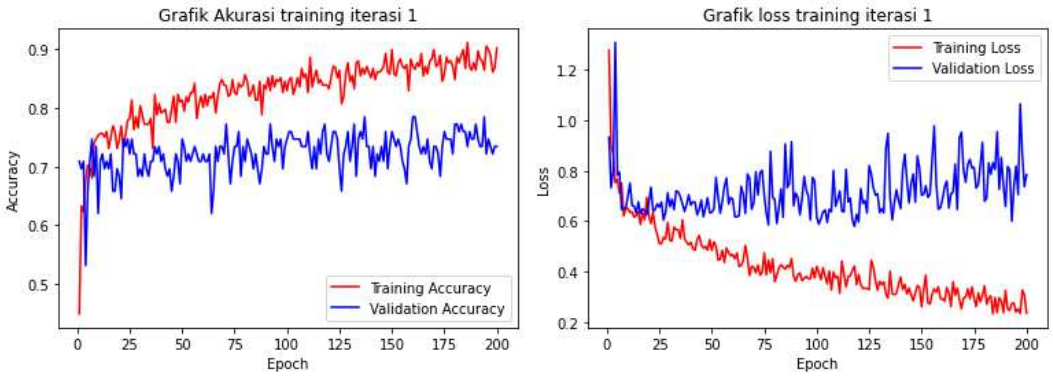
$$\overline{\text{Spesifisitas}} = \frac{\sum f_i x_i}{\sum f_i} \quad (4.6)$$

4.2 Hasil *Training*

Berikut adalah hasil *training* untuk setiap *fold*. Hasil *training* yang akan ditampilkan berupa grafik penurunan loss dan grafik peningkatan akurasi.

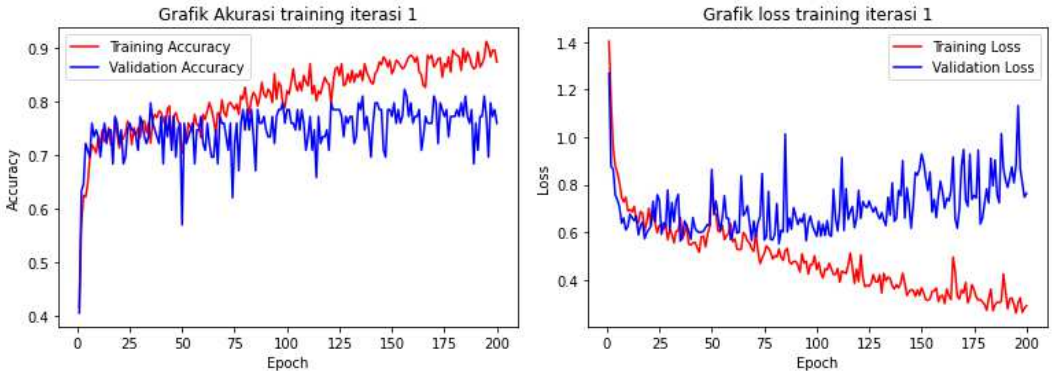
4.2.1 Hasil *Training Fold* Pertama

Pada *training* menggunakan data dengan pencahayaan lampu kamar, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi cenderung datar dan tidak ada penurunan. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu kamar untuk *fold* 1.



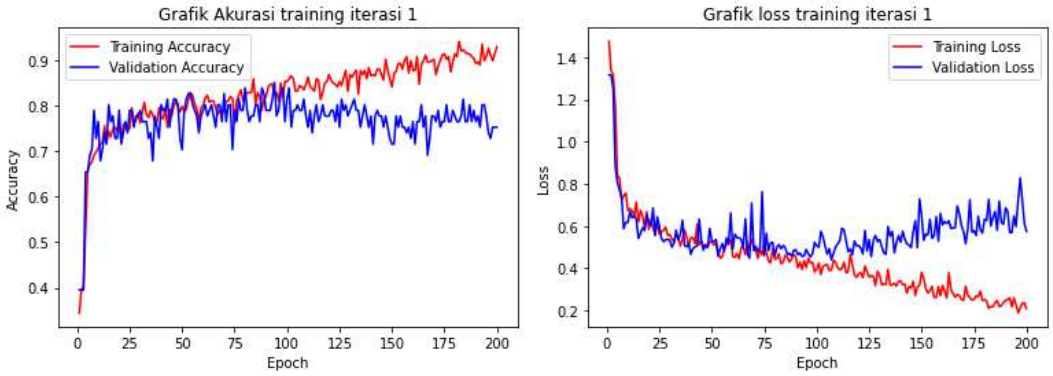
Gambar 4. 1 Grafik *akurasi dan loss* data dengan pencahayaan lampu kamar *training fold* 1

Pada *training* menggunakan data dengan pencahayaan lampu LED, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi cenderung mengalami peningkatan. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu LED untuk *fold* 1.



Gambar 4. 2 Grafik akurasi dan *loss* data dengan pencahayaan lampu LED *training fold* 1

Pada *training* menggunakan data dengan pencahayaan cahaya matahari, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi pada awalnya meningkat kemudian menjadi datar. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi pada awalnya mengalami penurunan kemudian terjadi peningkatan. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan cahaya matahari untuk *fold* 1.



Gambar 4. 3 Grafik akurasi dan *loss* data dengan pencahayaan cahaya matahari *training fold* 1

Setelah dilakukan *training* selama 200 *epoch* diperoleh hasil akurasi dan *loss* untuk setiap pencahayaan pada *training fold* 1 sebagai berikut:

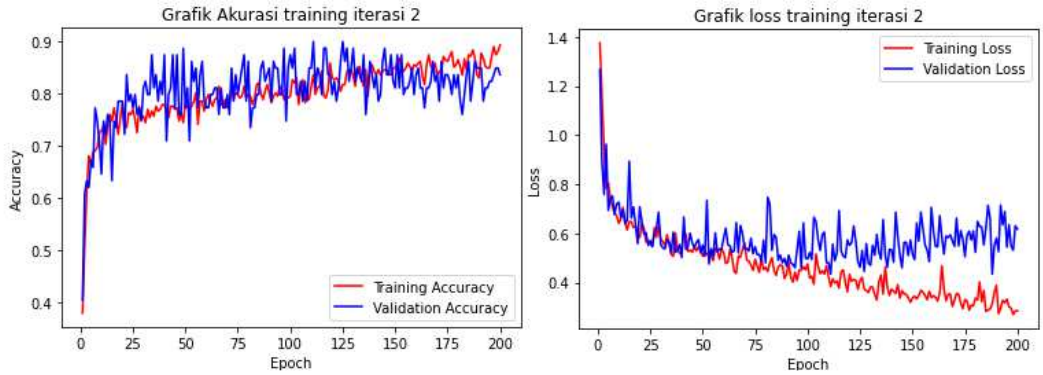
Tabel 4. 1 Hasil training fold 1

Pencahayaan	**Akurasi *training***	***loss***
Lampu Kamar	90.19%	0.2361
Lampu LED	87.38%	0.2934
Cahaya Matahari	92.99%	0.2078

Berdasarkan tabel di atas diketahui bahwa data dengan pencahayaan cahaya matahari memiliki hasil training terbaik untuk *training fold* 1.

4.2.2 Hasil *Training Fold* Kedua

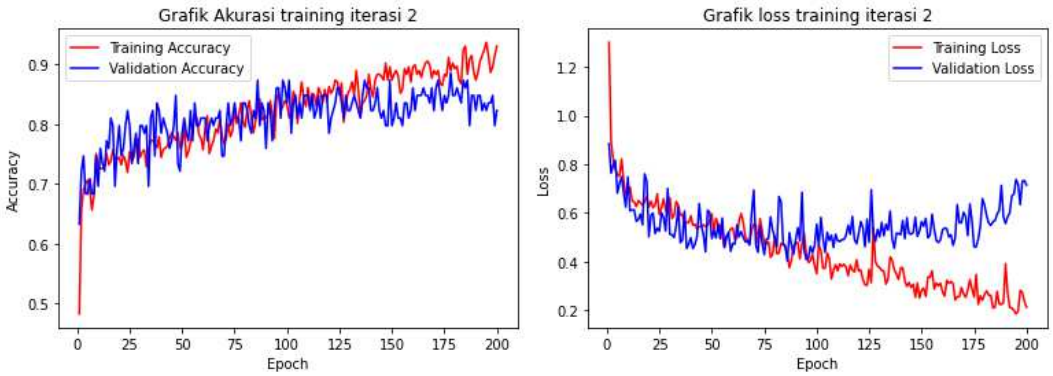
Pada *training* menggunakan data dengan pencahayaan lampu kamar, grafik akurasi *training* dan akurasi validasi sama-sama semakin meningkat seiring bertambahnya jumlah *epoch.* Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi pada awalnya mengalami penurunan kemudian terjadi peningkatan. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu kamar untuk *fold* 2.



Gambar 4. 4 Grafik akurasi dan *loss* data dengan pencahayaan lampu kamar *training fold* 2

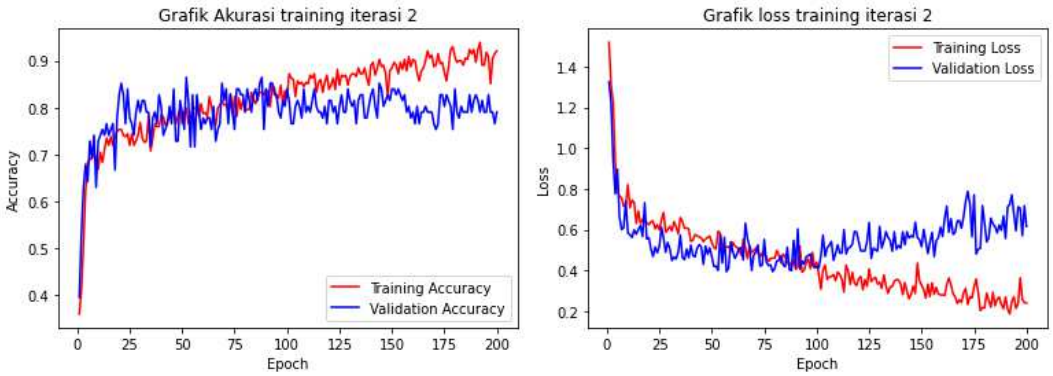
Pada *training* menggunakan data dengan pencahayaan lampu LED, grafik akurasi *training* dan akurasi validasi sama-sama semakin meningkat seiring bertambahnya jumlah *epoch.* Sementara grafik *loss training* semakin menurun

seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi pada awalnya mengalami penurunan kemudian terjadi peningkatan. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu LED untuk *fold* 2.



Gambar 4. 5 Grafik akurasi dan *loss* data dengan pencahayaan lampu LED *training fold* 2

Pada *training* menggunakan data dengan pencahayaan cahaya matahari, grafik akurasi *training* dan akurasi validasi sama-sama semakin meningkat seiring bertambahnya jumlah *epoch.* Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi pada awalnya mengalami penurunan kemudian terjadi peningkatan. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan cahaya matahari untuk *fold* 2.



Gambar 4. 6 Grafik akurasi dan *loss* data dengan pencahayaan cahaya matahari *training fold* 2

Setelah dilakukan *training* selama 200 *epoch* diperoleh hasil akurasi dan *loss* untuk setiap pencahayaan pada *training fold* 2 sebagai berikut:

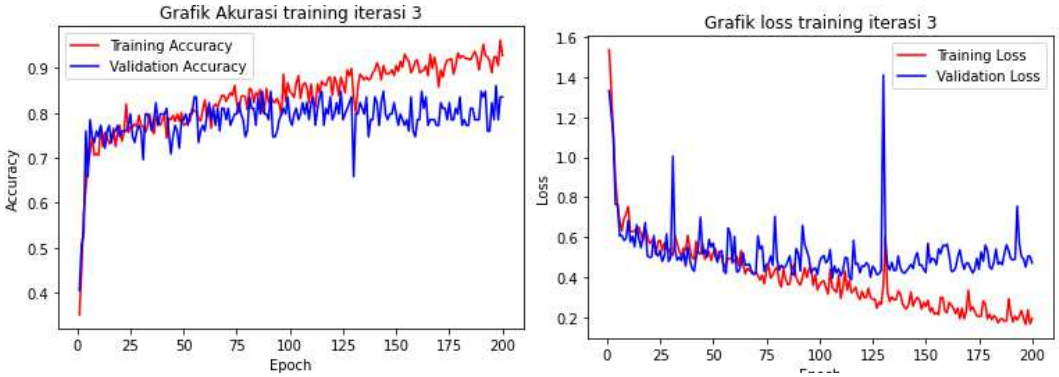
Tabel 4. 2 Hasil training fold 2

Pencahayaan	**Akurasi *training***	***loss***
Lampu Kamar	89.24%	0.2835
Lampu LED	93.06%	0.2112
Cahaya Matahari	92.07%	0.2405

Berdasarkan tabel di atas diketahui bahwa data dengan pencahayaan lampu LED memiliki hasil training terbaik untuk *training fold* 2.

4.2.3 Hasil *Training Fold* Ketiga

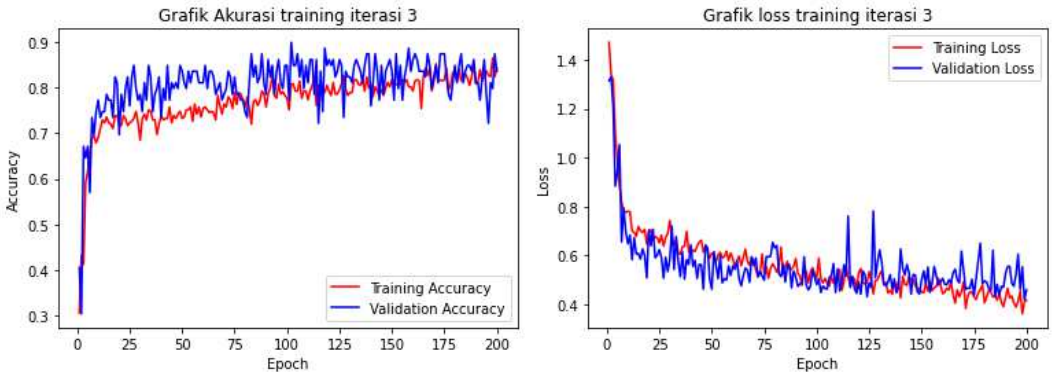
Pada *training* menggunakan data dengan pencahayaan lampu kamar, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi pada awalnya meningkat kemudian menjadi datar. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi pada awalnya mengalami penurunan kemudian mendatar. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu kamar untuk *fold* 3.



Gambar 4. 7 Grafik akurasi dan *loss* data dengan pencahayaan lampu kamar *training fold* 3

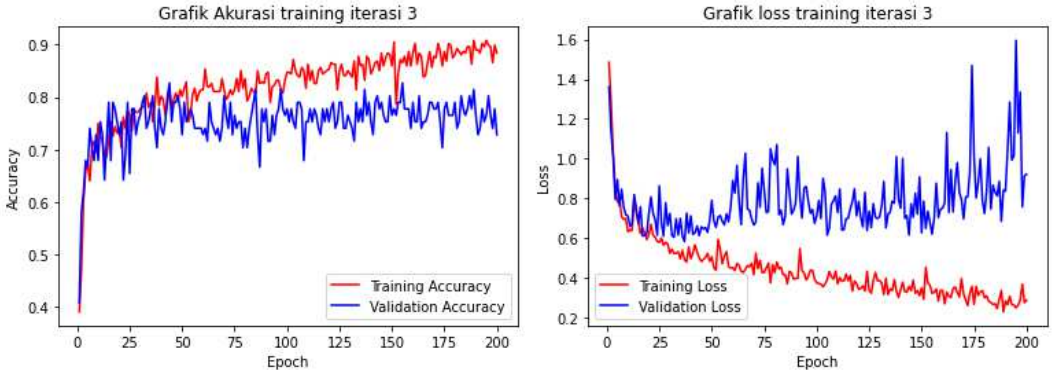
Pada *training* menggunakan data dengan pencahayaan lampu LED, grafik akurasi *training* dan validasi sama-sama semakin meningkat seiring bertambahnya

jumlah *epoch*. Sementara grafik *loss training* dan *loss* validasi sama-sama menurun seiring bertambahnya jumlah *epoch*. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu LED untuk *fold* 3.



Gambar 4. 8 Grafik akurasi dan *loss* data dengan pencahayaan lampu LED *training fold* 3

Pada *training* menggunakan data dengan pencahayaan cahaya matahari, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan cahaya matahari untuk *fold* 3.



Gambar 4. 9 Grafik akurasi dan *loss* data dengan pencahayaan cahaya matahari *training fold* 4

Setelah dilakukan *training* selama 200 *epoch* diperoleh hasil akurasi dan *loss* untuk setiap pencahayaan pada *training fold* 3 sebagai berikut:

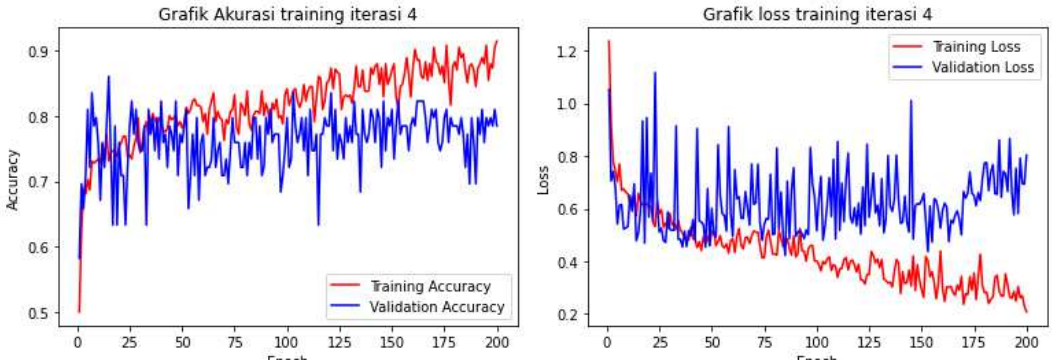
Tabel 4. 3 Hasil training fold 3

Pencahayaan	**Akurasi *training***	***loss***
Lampu Kamar	92.72%	0.1950
Lampu LED	84.23%	0.4168
Cahaya Matahari	88.41%	0.2896

Berdasarkan tabel di atas diketahui bahwa data dengan pencahayaan lampu kamar memiliki hasil training terbaik untuk *training fold* 3.

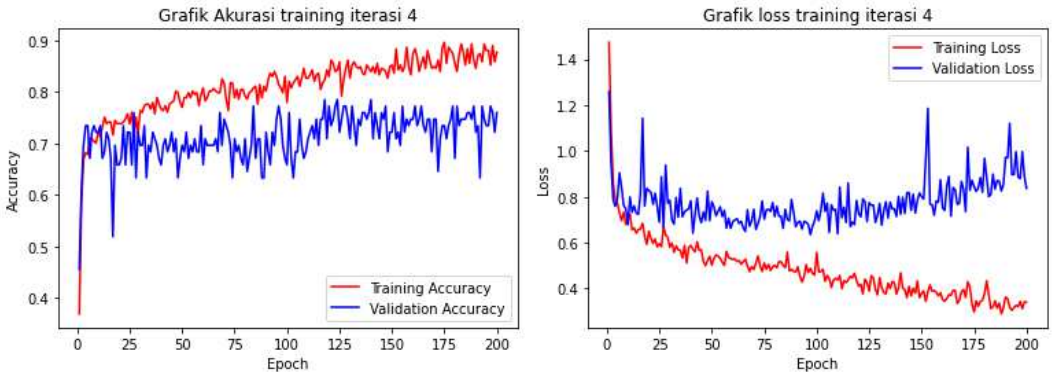
4.2.4 Hasil *Training Fold* Keempat

Pada *training* menggunakan data dengan pencahayaan lampu kamar, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu kamar untuk *fold* 4.



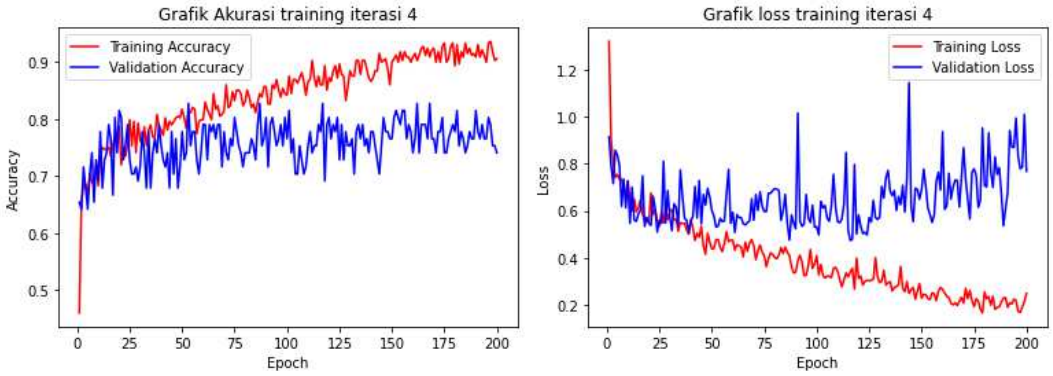
Gambar 4. 10 Grafik akurasi dan *loss* data dengan pencahayaan lampu kamar *training fold* 4

Pada *training* menggunakan data dengan pencahayaan lampu LED, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu LED untuk *fold* 4.



Gambar 4. 11 Grafik akurasi dan *loss* data dengan pencahayaan lampu LED *training fold* 4

Pada *training* menggunakan data dengan pencahayaan cahaya matahari, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan cahaya matahari untuk *fold* 4.



Gambar 4. 12 Grafik akurasi dan *loss* data dengan pencahayaan cahaya matahari *training fold* 4

Setelah dilakukan *training* selama 200 *epoch* diperoleh hasil akurasi dan *loss* untuk setiap pencahayaan pada *training fold* 4 sebagai berikut:

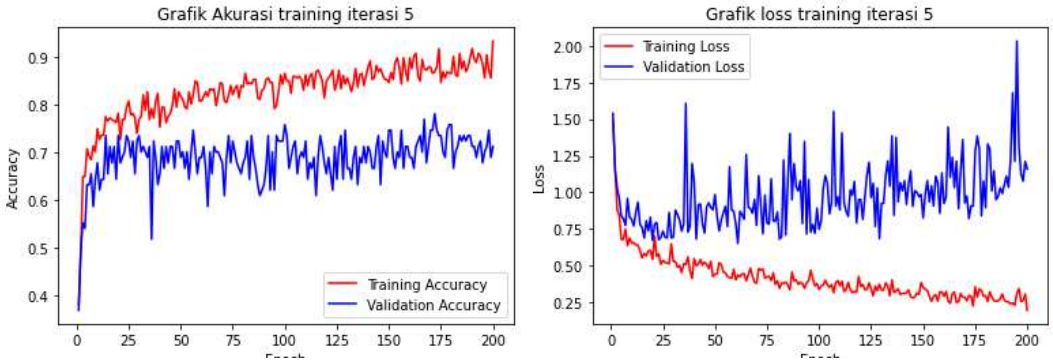
Tabel 4. 4 Hasil training fold 4

Pencahayaan	**Akurasi *training***	***loss***
Lampu Kamar	91.46%	0.2080
Lampu LED	87.70%	0.3400
Cahaya Matahari	90.55%	0.2492

Berdasarkan tabel di atas diketahui bahwa data dengan pencahayaan lampu kamar memiliki hasil training terbaik untuk *training fold* 4.

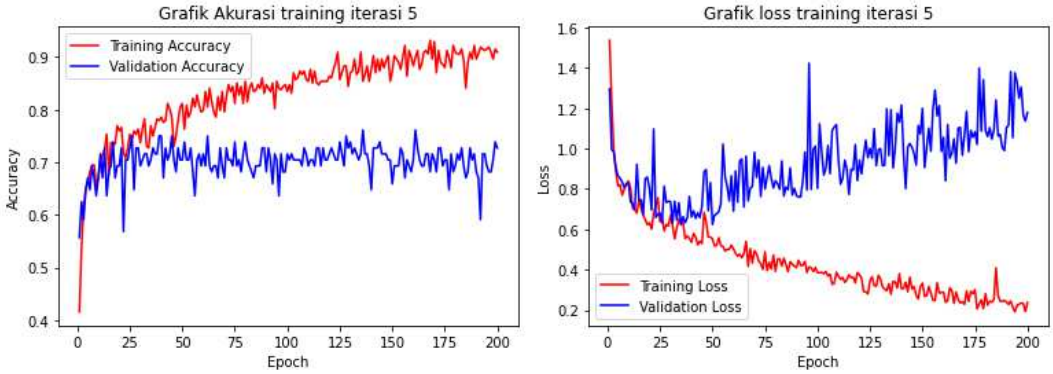
4.2.5 Hasil *Training Fold* Kelima

Pada *training* menggunakan data dengan pencahayaan lampu kamar, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu kamar untuk *fold* 5.



Gambar 4. 13 Grafik akurasi dan *loss* data dengan pencahayaan lampu kamar *training fold* 5

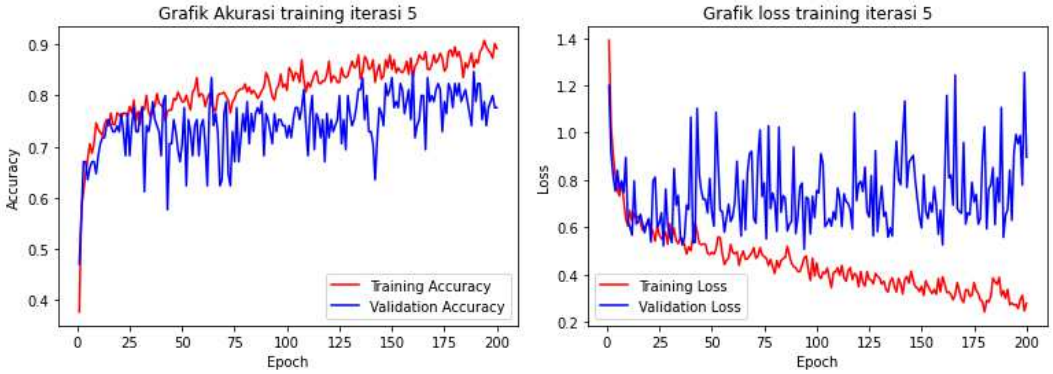
Pada *training* menggunakan data dengan pencahayaan lampu LED, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu LED untuk *fold* 5



Gambar 4. 14 Grafik akurasi dan *loss* data dengan pencahayaan lampu LED *training fold* 5

Pada *training* menggunakan data dengan pencahayaan cahaya matahari, grafik akurasi *training* dan grafik akurasi validasi sama-sama semakin meningkat seiring bertambahnya jumlah *epoch*. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik

akurasi dan *loss* untuk *training* data dengan pencahayaan cahaya matahari untuk *fold* 5



Gambar 4. 15 Grafik akurasi dan *loss* data dengan pencahayaan cahaya matahari *training fold* 5

Setelah dilakukan *training* selama 200 *epoch* diperoleh hasil akurasi dan *loss* untuk setiap pencahayaan pada *training fold* 5 sebagai berikut:

Tabel 4. 5 Hasil training fold 5

Pencahayaan	**Akurasi *training***	***loss***
Lampu Kamar	93.51%	0.1962
Lampu LED	90.91%	0.2374
Cahaya Matahari	90.55%	0.2492

Berdasarkan Tabel di atas diketahui bahwa data dengan pencahayaan lampu kamar memiliki hasil training terbaik untuk *training fold* 5.

4.3 Hasil Validasi

Berikut adalah hasil validasi untuk setiap *fold*. Hasil validasi yang akan ditampilkan berupa tabel jumlah TP, TN, FP, FN beserta nilai akurasi, sensitivitas dan spesifisitas untuk setiap jenis pencahayaan.

4.3.1 Hasil Validasi *Fold* Pertama

Setelah dilakukan validasi diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 1 sebagai berikut:

Tabel 4. 6 Hasil validasi fold 1

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	58	216	21	21	86.70%	73.41%	91.13%
Lampu LED	60	218	19	19	87.97%	75.94%	91.98%
Cahaya Matahari	61	223	20	20	87.65%	75.30%	91.76%

Berdasarkan tabel di atas diketahui bahwa hasil validasi terbaik untuk fold 1 diperoleh oleh data dengan pencahayaan lampu LED yaitu 87.97% akurasi, 75.94% sensitivitas, dan 91.98% spesifisitas.

4.3.2 Hasil Validasi *Fold* Kedua

Setelah dilakukan validasi diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 1 sebagai berikut:

Tabel 4. 7 Hasil validasi fold 2

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	66	224	13	13	91.77%	83.54%	94.51%
Lampu LED	65	223	14	14	91.13%	82.27%	94.09%
Cahaya Matahari	64	226	17	17	89.50%	79.01%	93.00%

Berdasarkan tabel di atas diketahui bahwa hasil validasi terbaik untuk fold 2 diperoleh oleh data dengan pencahayaan lampu kamar yaitu 91.77% akurasi, 83.54% sensitivitas, dan 94.51% spesifisitas.

4.3.3 Hasil Validasi *Fold* Ketiga

Setelah dilakukan validasi diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 1 sebagai berikut:

Tabel 4. 8 Hasil validasi fold 3

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	66	224	13	13	91.77%	83.54%	94.51%
Lampu LED	66	224	13	13	91.77%	83.54%	94.51%
Cahaya Matahari	59	221	22	22	86.41%	72.83%	90.94%

Berdasarkan tabel di atas diketahui bahwa hasil validasi terbaik untuk fold 3 diperoleh oleh data dengan pencahayaan lampu kamar dan lampu LED masing-masing yaitu 91.77% akurasi, 83.54% sensitivitas, dan 94.51% spesifisitas.

4.3.4 Hasil Validasi *Fold* Keempat

Setelah dilakukan validasi diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 1 sebagai berikut:

Tabel 4. 9 Hasil validasi fold 4

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	62	220	17	17	89.24%	78.48%	92.82%
Lampu LED	60	218	19	19	87.97%	75.94%	91.98%
Cahaya Matahari	60	222	21	21	87.07%	74.07%	91.35%

Berdasarkan tabel di atas diketahui bahwa hasil validasi terbaik untuk fold 4 diperoleh oleh data dengan pencahayaan lampu kamar yaitu 89.24% akurasi, 78.48% sensitivitas, dan 92.82% spesifisitas.

4.3.5 Hasil Validasi *Fold* Kelima

Setelah dilakukan validasi diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 1 sebagai berikut:

Tabel 4. 10 Hasil validasi fold 5

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	62	236	25	25	85.63%	71.26%	92.42%
Lampu LED	64	240	24	24	86.36%	72.72%	90.90%
Cahaya Matahari	66	236	19	19	88.82%	77.64%	92.54%

Berdasarkan tabel di atas diketahui bahwa hasil validasi terbaik untuk fold 5 diperoleh oleh data dengan pencahayaan cahaya matahari yaitu dengan 88.82% akurasi, 77.64% sensitivitas, dan 92.54% spesifisitas.

4.4 Hasil Uji Coba

Berikut adalah hasil uji coba untuk setiap *fold*. Hasil uji coba yang akan ditampilkan berupa tabel jumlah TP, TN, FP, FN beserta nilai akurasi, sensitivitas dan spesifisitas untuk setiap jenis pencahayaan

4.4.1 Hasil *Test Fold* Pertama

Setelah dilakukan uji coba diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 1 sebagai berikut:

Tabel 4. 11 Hasil uji coba fold 1

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	75	275	25	25	87.50%	75.00%	91.66%
Lampu LED	75	277	26	26	87.12%	74.25%	91.41%
Cahaya Matahari	78	278	22	22	89.00%	78.00%	92.66%

Berdasarkan tabel di atas diketahui bahwa hasil uji coba terbaik untuk fold 1 diperoleh oleh data dengan pencahayaan cahaya matahari yaitu dengan 89.00% akurasi, 78.00% sensitivitas, dan 92.66% spesifisitas.

4.4.2 Hasil *Test Fold* Kedua

Setelah dilakukan uji coba diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 2 sebagai berikut:

Tabel 4. 12 Hasil uji coba fold 2

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	73	273	27	27	86.50%	73.00%	91.00%
Lampu LED	77	279	24	24	88.11%	76.23%	92.07%
Cahaya Matahari	79	279	21	21	89.50%	79.00%	93.00%

Berdasarkan tabel di atas diketahui bahwa hasil uji coba terbaik untuk fold 2 diperoleh oleh data dengan pencahayaan cahaya matahari yaitu dengan 89.50% akurasi, 79.00% sensitivitas, dan 93.00% spesifisitas.

4.4.3 Hasil *Test Fold* Ketiga

Setelah dilakukan uji coba diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 3 sebagai berikut:

Tabel 4. 13 Hasil uji coba fold 3

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	77	277	23	23	88.50%	77.00%	92.33%
Lampu LED	71	273	30	30	85.14%	70.29%	90.09%
Cahaya Matahari	73	273	27	27	86.50%	73.00%	91.00%

Berdasarkan tabel di atas diketahui bahwa hasil uji coba terbaik untuk fold 3 diperoleh oleh data dengan pencahayaan lampu kamar yaitu dengan 88.50% akurasi, 77.00% sensitivitas, dan 92.33% spesifisitas.

4.4.4 Hasil *Test Fold* Keempat

Setelah dilakukan uji coba diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 4 sebagai berikut:

Tabel 4. 14 Hasil uji coba fold 4

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	77	277	23	23	88.50%	77.00%	92.33%
Lampu LED	74	276	27	27	86.63%	73.26%	91.08%
Cahaya Matahari	75	275	25	25	87.50%	75.00%	91.66%

Berdasarkan tabel di atas diketahui bahwa hasil uji coba terbaik untuk fold 4 diperoleh oleh data dengan pencahayaan lampu kamar yaitu dengan 88.50% akurasi, 77.00% sensitivitas, dan 92.33% spesifisitas.

4.4.5 Hasil *Test Fold* Kelima

Setelah dilakukan uji coba diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada *fold* 5 sebagai berikut:

Tabel 4. 15 Hasil uji coba fold 5

Pencahayaan	**TP**	**TN**	**FP**	**FN**	**Akurasi**	**Sensitivitas**	**Spesifisitas**
Lampu Kamar	74	274	26	26	87.00%	74.00%	91.33%
Lampu LED	77	279	24	24	88.11%	76.23%	92.07%
Cahaya Matahari	81	281	19	19	90.50%	81.00%	93.66%

Berdasarkan tabel di atas diketahui bahwa hasil uji coba terbaik untuk fold 5 diperoleh oleh data dengan pencahayaan cahaya matahari yaitu dengan 90.50% akurasi, 81.00% sensitivitas, dan 93.66% spesifisitas.

4.4.6 Hasil *Test* Rata-Rata Setiap *Fold*

Setelah *testing* sebanyak 5 *fold* dilakukan, hal terakhir yang dilakukan adalah menghitung nilai rata-rata akurasi, sensitivitas dan spesifisitas dari setiap *fold*. Berikut adalah tabel yang menampilkan tabulasi hasil *test* untuk data dengan pencahayaan lampu kamar:

Tabel 4. 16 Tabulasi hasil test data dengan pencahayaan lampu kamar

	Akurasi	Sensitivitas	Spesifisitas
Fold 1	87.50%	75.00%	91.66%
Fold 2	86.50%	73.00%	91.00%
Fold 3	88.50%	77.00%	92.33%
Fold 4	88.50%	77.00%	92.33%
Fold 5	87.00%	74.00%	91.33%
Rata-rata	87.60%	75.20%	91.73%

Berikut adalah tabel yang menampilkan tabulasi hasil *test* untuk data dengan pencahayaan lampu LED:

Tabel 4. 17 Tabulasi hasil test data dengan pencahayaan lampu LED

	Akurasi	Sensitivitas	Spesifisitas
Fold 1	87.12%	74.25%	91.41%
Fold 2	88.11%	76.23%	92.07%
Fold 3	85.14%	70.29%	90.09%
Fold 4	86.63%	73.26%	91.08%
Fold 5	88.11%	76.23%	92.07%
Rata-rata	87.02%	74.05%	91.34%

Berikut adalah tabel yang menampilkan tabulasi hasil *test* untuk data dengan pencahayaan cahaya matahari

Tabel 4. 18 Tabulasi hasil test data dengan pencahayaan cahaya matahari

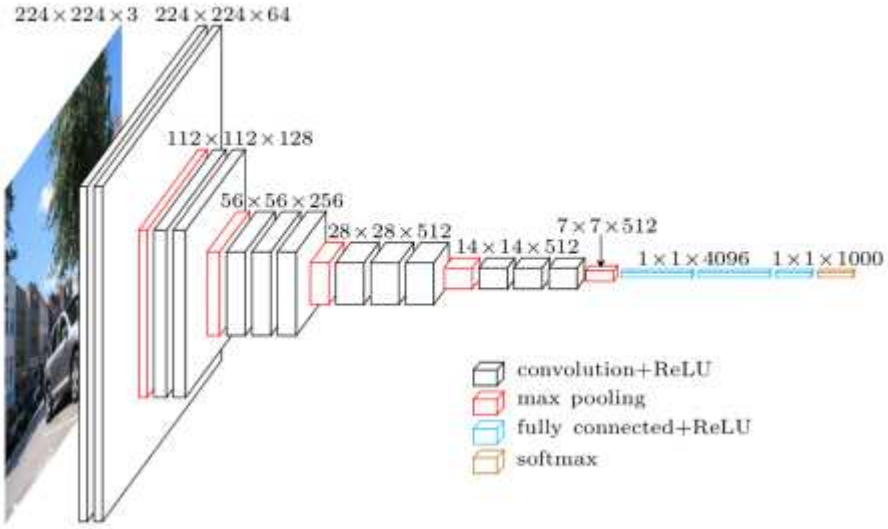
	Akurasi	Sensitivitas	Spesifisitas
Fold 1	89.00%	78.00%	92.66%
Fold 2	89.50%	79.00%	93.00%
Fold 3	86.50%	73.00%	91.00%
Fold 4	87.50%	75.00%	91.66%
Fold 5	90.50%	81.00%	93.66%
Rata-rata	88.60%	77.20%	92.40%

Berdasarkan tiga tabel di atas, diketahui bahwa hasil pengujian data dengan pencahayaan cahaya matahari memperoleh nilai rata-rata terbaik dengan hasil akurasi rata-rata 88.60%, sensitivitas rata-rata 77.20% dan spesifisitas rata-rata 92.40%.

4.5 Hasil Uji Coba Dengan Arsitektur VGG16

Arsitektur VGG16 merupakan salah satu arsitektur CNN yang populer. Arsitektur ini pertama kali diusulkan oleh Simonyan dan Zisserman (2014). Dalam

penelitian ini arsitektur VGG16 akan digunakan sebagai pembanding terhadap arsitektur CNN yang digunakan oleh peneliti. Berikut adalah gambar arsitektur VGG16:

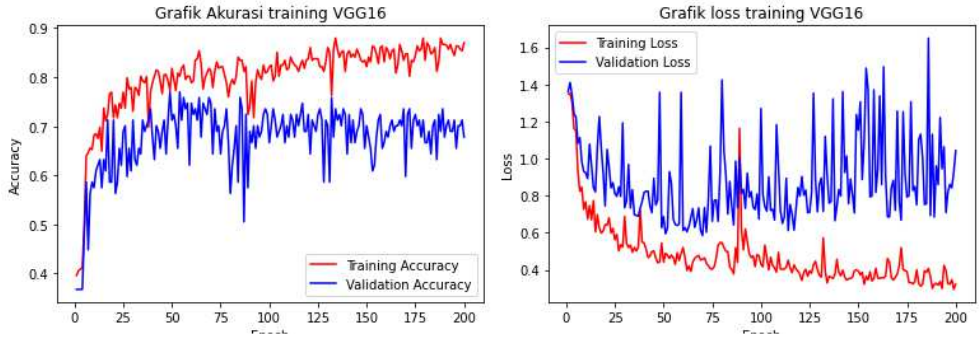


Gambar 4. 16 Arsitektur VGG16

Arsitektur VGG16 yang digunakan telah dimodifikasi agar dapat bekerja dengan data yang digunakan dalam penelitian ini. Data yang digunakan data *fold* 5 karena memiliki hasil terbaik daripada data pada *fold* lainnya. Berikut adalah hasil *training*, validasi dan *testing* menggunakan arsitektur VGG16.

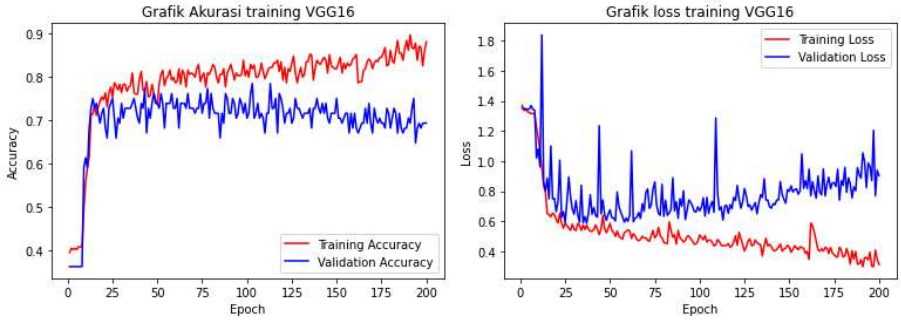
4.5.1 Hasil *Training*

Pada *training* menggunakan data dengan pencahayaan lampu kamar, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu kamar untuk arsitektur VGG16.



Gambar 4. 17 Grafik akurasi dan *loss* data dengan pencahayaan lampu kamar *training* VGG16

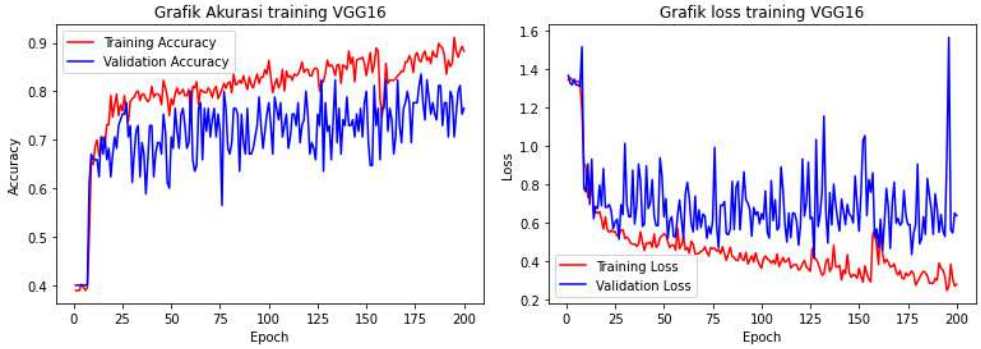
Pada *training* menggunakan data dengan pencahayaan lampu LED, grafik akurasi *training* semakin meningkat seiring bertambahnya jumlah *epoch* dan grafik akurasi validasi cenderung datar dan tidak ada peningkatan. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan lampu LED untuk arsitektur VGG16.



Gambar 4. 18 Grafik akurasi dan *loss* data dengan pencahayaan lampu LED *training* VGG16

Pada *training* menggunakan data dengan pencahayaan cahaya matahari, grafik akurasi *training* dan grafik akurasi validasi sama-sama semakin meningkat seiring bertambahnya jumlah *epoch*. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung meningkat. Berikut adalah grafik akurasi dan

loss untuk *training* data dengan pencahayaan cahaya matahari untuk arsitektur VGG16.



Gambar 4. 19 Grafik akurasi dan *loss* data dengan pencahayaan cahaya matahari *training* VGG16

menggunakan data dengan pencahayaan cahaya matahari, grafik akurasi *training* dan grafik akurasi validasi sama-sama semakin meningkat seiring bertambahnya jumlah *epoch*. Sementara grafik *loss training* semakin menurun seiring bertambahnya jumlah *epoch* dan grafik *loss* validasi tidak mengalami penurunan dan cenderung semakin meningkat. Berikut adalah grafik akurasi dan *loss* untuk *training* data dengan pencahayaan cahaya matahari

Setelah dilakukan *training* selama 200 *epoch* diperoleh hasil akurasi dan *loss* untuk setiap pencahayaan pada *training* dengan arsitektur VGG16 sebagai berikut:

Tabel 4. 19 Hasil *training* arsitektur VGG16

Pencahayaan	**Akurasi *training***	***loss***
Lampu Kamar	87.01%	0.3231
Lampu LED	87.99%	0.3123
Cahaya Matahari	88.29%	0.2802

Berdasarkan Tabel di atas diketahui bahwa data dengan pencahayaan cahaya matahari memiliki hasil training terbaik untuk arsitektur VGG16.

4.5.2 Hasil Validasi

Setelah dilakukan validasi diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada validasi dengan arsitektur VGG16 sebagai berikut:

Tabel 4. 20 Hasil validasi arsitektur VGG16

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	59	233	28	28	83.90%	67.81%	89.27%
Lampu LED	61	237	27	27	84.65%	69.31%	89.77%
Cahaya Matahari	65	235	20	20	88.23%	76.47%	92.16%

Berdasarkan tabel di atas diketahui bahwa hasil validasi terbaik untuk arsitektur VGG16 diperoleh oleh data dengan pencahayaan cahaya matahari yaitu dengan 88.23% akurasi, 76.47% sensitivitas, dan 92.16% spesifisitas.

4.5.3 Hasil *Testing*

Setelah dilakukan uji coba diperoleh tabel nilai akurasi, sensitivitas dan spesifisitas untuk setiap pencahayaan pada uji coba dengan arsitektur VGG16 sebagai berikut:

Tabel 4. 21 Hasil uji coba arsitektur VGG16

Pencahayaan	TP	TN	FP	FN	Akurasi	Sensitivitas	Spesifisitas
Lampu Kamar	83	283	17	17	91.50%	83.00%	94.33%
Lampu LED	77	279	24	24	88.11%	76.23%	92.07%
Cahaya Matahari	79	279	21	21	89.50%	79.00%	93.00%

Berdasarkan tabel di atas diketahui bahwa hasil uji coba terbaik untuk arsitektur VGG16 diperoleh oleh data dengan pencahayaan lampu kamar yaitu dengan 91.50% akurasi, 83.00% sensitivitas, dan 94.33% spesifisitas.

4.6 Pembahasan

Algoritma *Convolutional Neural Network* merupakan algoritma *deep learning* yang mempunyai keunggulan dalam klasifikasi dan identifikasi citra. Agar keunggulan ini dapat dimaksimalkan diperlukan jumlah data yang besar. Tidak hanya jumlah data yang besar tetapi diperlukan juga arsitektur *network* yang tepat agar dapat diperoleh hasil yang memuaskan.

Setelah dilakukan uji coba dengan metode *K-Fold Cross Validation* penelitian ini berhasil memperoleh hasil terbaik saat dilakukan uji coba menggunakan data dengan pencahayaan cahaya matahari yang memperoleh nilai rata-rata terbaik dengan hasil akurasi rata-rata 88.60%, sensitivitas rata-rata 77.20% dan spesifisitas rata-rata 92.40%.

Selain itu dilakukan uji coba dengan arsitektur CNN yang sudah ada. Dalam penelitian ini dilakukan pengujian menggunakan arsitektur VGG16. Setelah dilakukan uji coba diperoleh hasil terbaik saat dilakukan uji coba menggunakan data dengan pencahayaan lampu kamar yang memperoleh 91.50% akurasi, 83.00% sensitivitas, dan 94.33% spesifisitas.

Berikut adalah tabel perbandingan pengujian menggunakan tiga pencahayaan berbeda:

Tabel 4. 22 Perbandingan hasil *testing* menggunakan tiga pencahayaan berbeda

	Lampu kamar	**Lampu LED**	**Cahaya matahari**
Akurasi rata-rata	87.60%	87.02%	88.60%
Sensitivitas rata-rata	75.20%	74.05%	77.20%
Spesifisitas rata-rata	91.73%	91.34%	92.40%

Berdasarkan tabel di atas Diketahui bahwa pengujian menggunakan data dengan pencahayaan cahaya matahari memperoleh hasil rata-rata terbaik. Hal ini disebabkan karena cahaya matahari merupakan cahaya yang paling terang di antara cahaya yang lain yang digunakan dalam penelitian ini. Semakin terang cahaya maka ciri-ciri atau fitur yang ada daun apel menjadi lebih jelas sehingga identifikasi menjadi lebih baik.

Algoritma CNN pada dasarnya sudah memiliki banyak keunggulan dalam melakukan identifikasi penyakit pada tanaman berbasis citra. Keunggulan utama dalam CNN adalah adanya ekstraksi fitur yang sudah termasuk dalam arsitektur CNN. Ekstraksi fitur ini akan mempermudah proses *feature learning*.

Meskipun begitu, ekstraksi fitur yang ada dalam arsitektur CNN dapat menambah *parameter* yang harus dikomputasi. Hal ini tentu saja dapat menambah waktu untuk memproses data dengan CNN. Dalam hal ini teknologi *cloud computing* sangat membantu penulis dalam menjalankan pengujian. Teknologi *cloud computing* memungkinkan penulis dapat menggunakan perangkat bertenaga tinggi hanya melalui koneksi internet saja.

Algoritma CNN yang digunakan dalam penelitian ini diharapkan memiliki performa yang tinggi sehingga tidak ada citra yang diidentifikasi oleh sistem ke dalam class yang tidak semestinya. Apabila hasil identifikasi tidak akurat maka akan mengakibatkan kerancuan. Sama halnya ketika kebenaran dan kebatilan dicampur adukkan. Hal ini sejalan dengan firman Allah SWT dalam surah Al-Baqarah ayat 42.

وَلَا تَلْبِسُوا الْحَقَّ بِالْبَاطِلِ وَتَكْتُمُوا الْحَقَّ وَاَنْتُمْ تَعْلَمُوْنَ

"Dan janganlah kamu campur adukkan kebenaran dengan kebatilan dan (janganlah) kamu sembunyikan kebenaran, sedangkan kamu mengetahuinya" (QS Al-Baqarah: 42).

Imam Jalaluddin dalam Kitab Tafsirul Jalalain mengatakan, kata "al-haqq" atau kebenaran adalah kitab suci yang diturunkan kepada Ahli Kitab. Sedangkan kebatilan adalah keterangan dusta yang mereka ada-adakan. Sementara kebenaran yang mereka sembunyikan adalah sifat Nabi Muhammad SAW. "Jangan kalian sembunyikan" sifat Muhammad SAW. "Padahal kalian menyadarinya" bahwa itu adalah sesuatu yang hak.

Penelitian ini diharapkan dapat membuat peneliti menjadi orang yang dapat bermanfaat untuk orang lain yakni para petani apel. Seorang muslim yang baik adalah orang dapat bermanfaat kepada sesama. Setiap kebaikan yang kita lakukan akan kembali kepada diri kita sendiri begitu pula sebaliknya. Allah berfirman dalam Al-Quran Surah Al-Isra ayat 7:

اِنْ اَحْسَنْتُمْ اَحْسَنْتُمْ لِاَنْفُسِكُمْ ۗوَاِنْ اَسَأْتُمْ فَلَهَا ۗفَاِذَا جَآءَ وَعْدُ الْاٰخِرَةِ لِيَسُۤوْا وُجُوْهَكُمْ وَلِيَدْخُلُوا الْمَسْجِدَ كَمَا دَخَلُوْهُ اَوَّلَ مَرَّةٍ وَّلِيُتَبِّرُوْا مَا عَلَوْا تَتْبِيْرًا

"*Jika kamu berbuat baik (berarti) kamu berbuat baik untuk dirimu sendiri. Dan jika kamu berbuat jahat, maka (kerugian kejahatan) itu untuk dirimu sendiri. Apabila datang saat hukuman (kejahatan) yang kedua, (Kami bangkitkan musuhmu) untuk menyuramkan wajahmu lalu mereka masuk ke dalam masjid (Masjidil Aqsa), sebagaimana ketika mereka memasukinya pertama kali dan mereka membinasakan apa saja yang mereka kuasai." (QS. Al-Isra ayat 7).*

Berdasarkan tafsir Al-Jalalain, berikut adalah penjelasan dari ayat di atas:

Kemudian Kami katakan (Jika kalian berbuat baik) dengan mengerjakan ketaatan (berarti kalian berbuat baik bagi diri kalian sendiri) karena sesungguhnya pahala kebaikan itu untuk diri kalian sendiri (dan jika kalian berbuat jahat) dengan menimbulkan kerusakan (maka kejahatan itu bagi diri kalian sendiri) sebagai pembalasan atas kejahatan kalian. (Dan apabila datang saat hukuman) bagi kejahatan yang (kedua) maka Kami kembali mengutus mereka (untuk menyuramkan muka-muka kalian) untuk membuat kalian sedih karena terbunuh dan tertawan hingga pengaruh kesedihan itu dapat terbaca dari roman muka kalian (dan mereka masuk ke dalam masjid) yakni Baitul Maqdis untuk menghancurkannya (sebagaimana musuh-musuh kalian memasukinya) dan menghancurkannya (pada kali pertama dan untuk menghancurkan) untuk mengadakan pembinasaan (terhadap apa saja yang mereka kuasai) yang dapat mereka kalahkan (dengan penghancuran habis-habisan) dengan pembinasaan yang sehabis-habisnya. Ternyata mereka melakukan kerusakan untuk kedua kalinya, yaitu dengan membunuh Nabi Yahya. Maka Allah mengirimkan untuk membinasakan mereka Raja Bukhtanashar. Raja Bukhtanashar akhirnya membunuh ribuan orang dari kalangan mereka dan menahan anak cucu mereka serta memporak-porandakan Baitul Maqdis.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil uji coba yang telah dilakukan, dapat disimpulkan bahwa sistem yang telah dibuat memiliki akurasi rata-rata 88.60%, sensitivitas rata-rata 77.20% dan spesifisitas rata-rata 92.40% pada pengujian menggunakan data dengan pencahayaan cahaya matahari. Hal ini disebabkan karena cahaya matahari merupakan cahaya yang paling terang di antara cahaya yang lain yang digunakan dalam penelitian ini. Semakin terang cahaya maka ciri-ciri atau fitur yang ada daun apel menjadi lebih jelas sehingga identifikasi menjadi lebih baik.

Kemudian dilakukan uji coba menggunakan arsitektur VGG16 dan diperoleh hasil 91.50% akurasi, 83.00% sensitivitas, dan 94.33% spesifisitas pada pengujian menggunakan data dengan pencahayaan lampu kamar.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, masih banyak kekurangan dan masih banyak ruang untuk penyempurnaan. Beberapa saran untuk penelitian di masa yang akan datang, antara lain:

a. Jumlah data yang digunakan sebaiknya ditambah dan mencakup lebih banyak penyakit yang dapat dideteksi.
b. Sistem yang telah dibuat sebaiknya diintegrasikan dengan aplikasi web atau aplikasi *mobile* agar lebih mudah digunakan.

DAFTAR PUSTAKA

Badan Pusat Statistik Kabupaten Malang, Kabupaten Malang Dalam Angka 2020, Malang : Badan Pusat Statistik

Badan Pusat Statistik Kota Batu, Kota Batu Dalam Angka 2020, Kota Batu : Badan Pusat Statistik

Baranwal, S., Khandelwal, S., & Arora, A. (2019, February). Deep learning convolutional neural network for apple leaves disease detection. In Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India.

Dubey, S. R., & Jalal, A. S. (2016). Apple disease classification using color, texture and shape features from images. Signal, Image and Video Processing, 10(5), 819-826.

Hidayat, A., Darusalam, U., & Irmawati, I. (2019). Detection of Disease on Corn Plants Using Convolutional Neural Network Methods. Jurnal Ilmu Komputer dan Informasi, 12(1), 51-56.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and electronics in agriculture, 147, 70-90.

Liu, T., Fang, S., Zhao, Y., Wang, P., & Zhang, J. (2015). Implementation of training convolutional neural networks. arXiv preprint arXiv:1506.01195.

Lu, Y., Yi, S., Zeng, N., Liu, Y., & Zhang, Y. (2017). Identification of rice diseases using deep convolutional neural networks. Neurocomputing, 267, 378-384.

Ma, J., Du, K., Zheng, F., Zhang, L., Gong, Z., & Sun, Z. (2018). A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network. Computers and electronics in agriculture, 154, 18-24.

Primartha, Rifkie, 2018, Belajar Machine Learning; Teori dan Praktik, Kota Bandung : Penerbit Informatika

Priya, C. A., Balasaravanan, T., & Thanamani, A. S. (2012, March). An efficient leaf recognition algorithm for plant classification using support vector machine. In *International conference on pattern recognition, informatics and medical engineering (PRIME-2012)* (pp. 428-432). IEEE.

Samajpati, B. J., & Degadwala, S. D. (2016, April). Hybrid approach for apple fruit diseases detection and classification using random forest classifier. In 2016 International conference on communication and signal processing (ICCSP) (pp. 1015-1019). IEEE.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

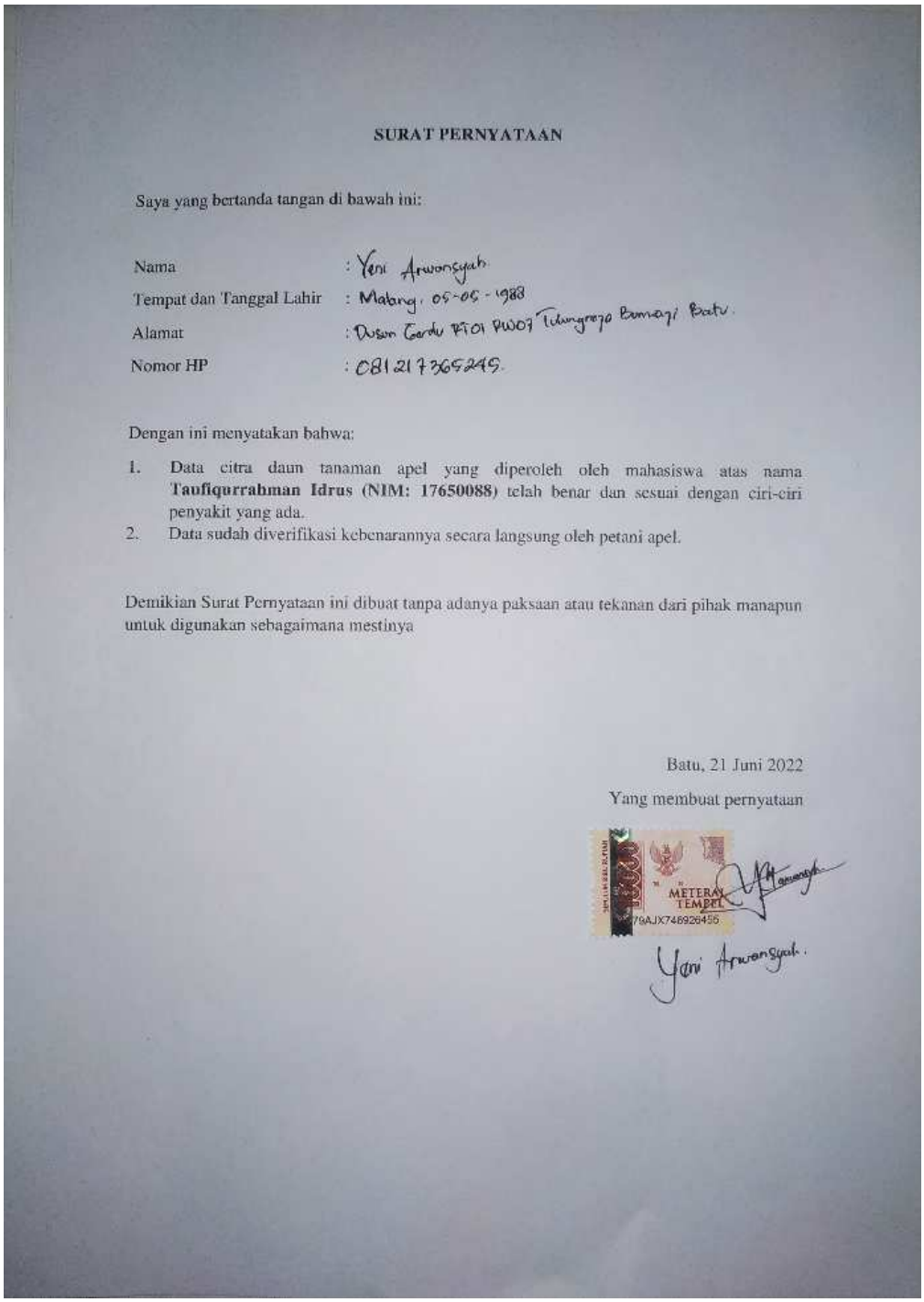
Skalski, P. (2019). Gentle Dive into Math Behind Convolutional Neural Networks. Retrieved December 14, 2021, from https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9

Turkoglu, M., Hanbay, D., & Sengur, A. (2019). Multi-model LSTM-based convolutional neural networks for detection of apple diseases and pests. Journal of Ambient Intelligence and Humanized Computing, 1-11.

LAMPIRAN-LAMPIRAN

Lampiran 1

Surat pernyataan kebenaran data oleh petani apel



SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama	: Yeni Arwansyah
Tempat dan Tanggal Lahir	: Malang, 05-05-1988
Alamat	: Dusun Gardu RT01 RW07 Tulungrejo Bumiaji Batu.
Nomor HP	: 081217365249

Dengan ini menyatakan bahwa:

1. Data citra daun tanaman apel yang diperoleh oleh mahasiswa atas nama **Taufiqurrahman Idrus (NIM: 17650088)** telah benar dan sesuai dengan ciri-ciri penyakit yang ada.
2. Data sudah diverifikasi kebenarannya secara langsung oleh petani apel.

Demikian Surat Pernyataan ini dibuat tanpa adanya paksaan atau tekanan dari pihak manapun untuk digunakan sebagaimana mestinya

Batu, 21 Juni 2022

Yang membuat pernyataan

METERAI TEMPEL 79AJX748926455

Yeni Arwansyah.

Lampiran 2

Kode Program

```
import numpy as np
import pickle
import os
import cv2
import tensorflow as tf
import keras
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, De
nse
from keras import backend as K
from tensorflow.keras.optimizers import Adam
from keras.preprocessing import image
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from numpy import mean
from sklearn.metrics import confusion_matrix, ConfusionMatrixD
isplay, accuracy_score, classification_report, recall_score
!pip install pyunpack
!pip install patool
from pyunpack import Archive
Archive('/content/fold5.rar').extractall('/content/Datasets/')
iterasi = 5
width=256
height=256
default_image_size = tuple((height, width))
directory_root_train = '/content/Datasets/fold5/Train'
directory_root_validation = '/content/Datasets/fold5/Validatio
n'
directory_root_test = '/content/Datasets/fold5/Test'
image_size = 0
depth=3
def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return keras.preprocessing.image.img_to_array(imag
e)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None
image_list_train, label_list_train = [], []
try:
```

```
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root_train)
    for plant_disease_folder in root_dir:
        print(f"[INFO] Processing {plant_disease_folder} ...")
        plant_disease_image_list = listdir(f"{directory_root_t
rain}/{plant_disease_folder}/")
             for image in plant_disease_image_list:
            image_directory = f"{directory_root_train}/{plant_
disease_folder}/{image}"
            if image_directory.endswith(".jpg") == True or ima
ge_directory.endswith(".JPG") == True:
                image_list_train.append(convert_image_to_array
(image_directory))
                label_list_train.append(plant_disease_folder)
    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")
#image labeling
image_size = len(image_list_train)
label_binarizer = LabelBinarizer()
image_labels_train = label_binarizer.fit_transform(label_list_
train)
pickle.dump(label_binarizer,open('label_transform_train.pkl',
'wb'))
n_classes = len(label_binarizer.classes_)
np_image_list_train = np.array(image_list_train, dtype=np.floa
t16) / 255.0

image_list_validation, label_list_validation = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root_validation)

    for plant_disease_folder in root_dir:
        print(f"[INFO] Processing {plant_disease_folder} ...")
        plant_disease_image_list = listdir(f"{directory_root_v
alidation}/{plant_disease_folder}/")

        for image in plant_disease_image_list:
            image_directory = f"{directory_root_validation}/{p
lant_disease_folder}/{image}"

            if image_directory.endswith(".jpg") == True or ima
ge_directory.endswith(".JPG") == True:
                image_list_validation.append(convert_image_to_
array(image_directory))
                label_list_validation.append(plant_disease_fol
der)
    print("[INFO] Image loading completed")

except Exception as e:
    print(f"Error : {e}")
```

```
#image labeling
image_size = len(image_list_validation)
label_binarizer = LabelBinarizer()
image_labels_validation = label_binarizer.fit_transform(label_
list_validation)
pickle.dump(label_binarizer,open('label_transform_validation.p
kl', 'wb'))
n_classes = len(label_binarizer.classes_)
np_image_list_validation = np.array(image_list_validation, dty
pe=np.float16) / 255.0

image_list_test, label_list_test = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root_test)

    for plant_disease_folder in root_dir:
        print(f"[INFO] Processing {plant_disease_folder} ...")
        plant_disease_image_list = listdir(f"{directory_root_t
est}/{plant_disease_folder}/")

        for image in plant_disease_image_list:
            image_directory = f"{directory_root_test}/{plant_d
isease_folder}/{image}"

            if image_directory.endswith(".jpg") == True or ima
ge_directory.endswith(".JPG") == True:
                image_list_test.append(convert_image_to_array(
image_directory))
                label_list_test.append(plant_disease_folder)
    print("[INFO] Image loading completed")

except Exception as e:
    print(f"Error : {e}")

#image labeling
image_size = len(image_list_test)
label_binarizer = LabelBinarizer()
image_labels_test = label_binarizer.fit_transform(label_list_t
est)
pickle.dump(label_binarizer,open('label_transform_test.pkl', '
wb'))
n_classes = len(label_binarizer.classes_)
np_image_list_test = np.array(image_list_test, dtype=np.float1
6) / 255.0

aug = keras.preprocessing.image.ImageDataGenerator(
    rotation_range=25,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
```

```
    fill_mode="nearest")
model = keras.models.Sequential()
inputShape = (height, width, depth)

model.add(Conv2D(20, (5, 5), padding="same",input_shape=inputS
hape))
model.add(Activation("relu"))

model.add(MaxPooling2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(50, (5, 5), padding="same"))
model.add(Activation("relu"))

model.add(MaxPooling2D(pool_size=(2, 2),strides=2))

model.add(Flatten())

model.add(Dense(500))
model.add(Activation("relu"))

model.add(Dense(4))
model.add(Activation("softmax"))

EPOCHS = 200
INIT_LR = 1e-3
BS = 8

with tf.device('/GPU:0'):
  opt = Adam(learning_rate=INIT_LR, decay=INIT_LR / EPOCHS)
  model.compile(loss="binary_crossentropy", optimizer=opt,metr
ics=["Accuracy", "FalseNegatives", "FalsePositives", "TrueNega
tives", "TruePositives"])

  history = model.fit(
      aug.flow(np_image_list_train, image_labels_train, batch_
size=BS),
      validation_data=(np_image_list_validation, image_labels_
validation),
      steps_per_epoch=len(np_image_list_train) // BS,
      epochs=EPOCHS,
      verbose=1,
      )

#plotting
  epochs = range(1, len(history.history['loss'])+1)
  plt.title(f"Grafik loss training iterasi {iterasi}")
  plt.plot(epochs, history.history['loss'], 'r', label='Traini
ng Loss')
  plt.plot(epochs, history.history['val_loss'], 'b', label='Va
lidation Loss')
  plt.xlabel("Epoch")
  plt.ylabel("Loss")
  plt.legend()
```

```
  plt.show()

#plotting
  epochs = range(1, len(history.history['loss'])+1)
  plt.title(f"Grafik Akurasi training iterasi {iterasi}")
  plt.plot(epochs, history.history['Accuracy'], 'r', label='Tr
aining Accuracy')
  plt.plot(epochs, history.history['val_Accuracy'], 'b', label
='Validation Accuracy')
  plt.xlabel("Epoch")
  plt.ylabel("Accuracy")
  plt.legend()
  plt.show()
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from numpy import mean
from sklearn.metrics import confusion_matrix, ConfusionMatrixD
isplay, accuracy_score, classification_report, recall_score

validation = model.predict(np_image_list_validation)

predicted_labels = []
correct_labels = []
for i in range (len(validation)):
  predicted_labels.append(np.argmax(validation[i]))

for i in range (len(image_labels_validation)):
  correct_labels.append(np.argmax(image_labels_validation[i]))
confusion_matrix = confusion_matrix(correct_labels, predicted_
labels)
disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matri
x)
disp.plot()
plt.show()
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from numpy import mean
from sklearn.metrics import confusion_matrix, ConfusionMatrixD
isplay, accuracy_score, classification_report, recall_score
test = model.predict(np_image_list_test)
predicted_labels = []
correct_labels = []
for i in range (len(test)):
  predicted_labels.append(np.argmax(test[i]))
for i in range (len(image_labels_test)):
  correct_labels.append(np.argmax(image_labels_test[i]))
confusion_matrix = confusion_matrix(correct_labels, predicted_
labels)
disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matri
x)
disp.plot()
plt.show()
```