Pengembangan Perangkat Lunak Untuk Pengelompokkan Tumbuhan Berdasarkan Citra Digital Daun Menggunakan CNN

Chelsea Oktaviany

Informatika, Fakultas Ilmu Komputer dan Desain, Institut Teknologi dan Bisnis Kalbis
Jalan Pulomas Selatan Kav. 22, Jakarta 13210
Email: chelseaoktaviany97@gmail.com

Abstract: *This study aims to develop software to classify plants based on digital images of leaves, the data is used as training data is 222 data which is divided into 4 data groups. While the testing data is used in this study is 45 data. The method is used in this study is Convolutional Neural Network which is applied using Tensorflow Keras. From results of the study, algorithm produces an accuracy value of 20% on testing data and 96.29% on training data with 50 epochs.*
Keywords: *Accuracy, Classification, Convolutional Neural Network, Leaves*

Abstrak: *Penelitian ini bertujuan untuk mengembangkan perangkat lunak guna mengelompokkan tanaman berdasarkan citra digital daun, data yang digunakan sebagai data latih sebanyak 222 data yang terbagi dalam 4 kelompok data. Sedangkan data uji yang digunakan dalam penelitian ini sebanyak 45 data. Metode yang digunakan dalam penelitian adalah Jaringan Saraf Konvolusi yang diterapkan menggunakan Tensorflow Keras. Dari hasil penelitian, algoritma menghasilkan nilai akurasi sebesar 20% pada data uji dan 96.29% pada data latih dengan epoch 50.*
Kata kunci: *Akurasi, Daun, Jaringan Saraf Konvolusi, Klasifikasi*

I. PENDAHULUAN

Sistem identifikasi citra daun merupakan sistem yang pengaplikasiannya hampir mirip dengan sistem pengenalan wajah, sistem pengidentifikasian citra daun ini sendiri terdiri dari tahap deteksi dan klasifikasi. Kedua tahap tersebut begitu cepat dilakukan oleh manusia tetapi butuh waktu yang lama bagi komputer. Kemampuan manusia itulah yang ingin ditiru oleh para peneliti dalam beberapa tahun belakangan ini sebagai teknologi biometrik dalam bidang *computer vision* dengan tujuan membentuk suatu model untuk pengidentifikasian citra daun pada komputer [1, hlm. 23]. Daun merupakan salah satu organ tumbuhan yang tumbuh dari ranting dan biasanya berwarna hijau (mengandung klorofil) dan terutama berfungsi sebagai penangkap energi dari cahaya matahari untuk fotosintesis. Daun memiliki bentuk daun dan bentuk tulang daun yang beragam.

Menurut Felix, dalam kehidupan sehari-hari, manusia sering melihat tanaman di sekitarnya dengan ciri-ciri yang beraneka ragam. Tetapi saat ini masih banyak orang-orang yang belum sanggup membedakan jenis tanaman [2, hlm. 2]. Selama ini identifikasi pada pada tumbuhan yang dilakukan masih dengan cara *manual* dan tidak efisien atau kurang teliti untuk tumbuhan dalam jumlah yang cukup banyak. Dengan perkembangan teknologi pada saat ini memungkinkan melakukan identifikasi dan klasifikasi terhadap tumbuhan menggunakan komputer.

Indonesia merupakan negara *megabiodiversity* dimana Indonesia

memiliki kekayaan tumbuhan obat yang sangat potensial untuk dikembangkan. Indonesia memiliki lebih dari 38.000 spesies tanaman dengan lebih dari 2039 spesies merupakan jenis dari tumbuhan Banyaknya kegiatan klasifikasi tidak lain adalah pembentukan kelompok-kelompok makhluk hidup dengan cara mencari keseragaman ciri atau sifat di dalam keanekaragaman ciri yang ada pada makhluk hidup tersebut. Dalam pertanian, tanaman adalah beberapa jenis organisme yang dibudidayakan pada suatu ruang atau media untuk dipanen pada masa ketika sudah mencapai tahap pertumbuhan tertentu.

Pada beberapa penelitian sebelumnya telah dilakukan klasifikasi jenis tanaman berdasarkan citra digital daun. Penelitian tersebut diantaranya melakukan klasifikasi dengan menggabungkan fitur daun. Penelitian tersebut diantaranya melakukan klasifikasi dengan menggabungkan fitur bentuk warna, dan tekstur daun. Dalam bidang Teknik Informatika, permasalahan pengenalan objek dikenal dengan istilah Klasifikasi. Dalam metode klasifikasi ciri-ciri dari tiap objek tanaman akan digunakan sebagai dasar dalam mengenali objek tersebut. Seperti misalnya warna daun, bentuk daun, ukuran daun dan tekstur. Ciri-ciri ini nantinya akan disebut sebagai fitur (*features*), yang akan digunakan sebagai input ke dalam suatu metode pengenalan. *Feature* ini didapatkan dari proses yang disebut sebagai *feature extraction*. Dimana dari gambar suatu daun akan dihitung secara otomatis oleh komputer ciri-ciri *features* yang terdapat padanya. Untuk mengenali atau mengidentifikasi daun cara yang paling sederhana adalah dengan melihat daun berdasarkan bentuk daunnya, akan tetapi tidak banyak orang yang dapat membedakan antara daun yang satu dengan yang lain [3, hlm. 2].

Pengklasifikasian gambar dengan menggunakan CNN banyak digunakan. Contoh penggunaan sistem *computer vision* untuk mengidentifikasi spesis tumbuhan menggunakan *leafsnap*, *foliage* dan *flavia dataset* dengan algoritma CNN pada penelitian sebelumnya pada jurnal "*Automated Plant Species Identification – Trends And Future Directions*" [4]. Mayoritas metode ini tidak dapat langsung diterapkan tetapi memerlukan fase pelatihan di mana pengklasifikasi belajar untuk membedakan kelas yang diminati. Untuk identifikasi spesies, fase pelatihan (*training phase*) terdiri dari analisis gambar yang telah diidentifikasi secara independen dan akurat sebagai taksa dan sekarang digunakan untuk menentukan *parameter* pengklasifikasi untuk memberikan diskriminasi maksimum antara taksa terlatih ini. Dalam fase aplikasi (*application phase*), pengklasifikasi terlatih kemudian diperlihatkan ke gambar baru yang menggambarkan spesimen tidak dikenal dan seharusnya menetapkannya ke salah satu taksa terlatih [4, hlm. 3]. Gambar biasanya terdiri dari jutaan piksel dengan informasi warna terkait. Informasi ini terlalu luas dan berantakan untuk digunakan secara langsung oleh algoritma *machine learning*. Oleh karena itu, dimensi tinggi dari gambar-gambar ini dikurangi dengan menghitung vektor fitur, yaitu representasi gambar yang dikuantifikasi yang berisi informasi yang relevan untuk masalah klasifikasi [4, hlm. 3].

Pada pembelajaran dalam, semua prosesnya membutuhkan algoritma yang dimana algoritma tersebut dapat memberikan hasil yang diinginkan. Algoritma ini bertujuan untuk melakukan proses di setiap pembelajaran dalam. Setiap algoritma memiliki kelebihan dan kelemahan masing-masing. Setiap algoritma yang berhasil akan mengeluarkan hasil berupa akurasi yang berguna untuk melihat apakah algoritma tersebut cocok

digunakan dalam memecahkan sebuah masalah yang dihadapi.

Peneliti menggunakan algoritma *Convolutional Neural Network* untuk mengklasifikasi jenis tanaman berdasarkan citra digital daun. Salah satu algoritma yang sering digunakan dalam pembelajaran dalam (*deep learning*) adalah *Convolutional Neural Network*. Algoritma *Convolutional Neural Network* sangat sering digunakan dalam pembelajaran dalam. *Convolutional Neural Network* bekerja sangat baik dibanding dengan algoritma lain. Algoritma *Convolutional Neural Network* (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menanggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik [5, hlm. 65].

Deep Learning adalah cabang ilmu *machine learning* berbasis Jaringan Saraf Tiruan (JST) atau bisa dikatakan sebagai perkembangan dari JST. Dalam *deep learning*, sebuah komputer belajar mengklasifikasi secara langsung dari gambar atau suara [6, hlm. 49].

Aplikasi yang digunakan untuk mengidentifikasi spesies tumbuhan diperlukan diakses dari berbagai *platform*, sehingga berbagai pihak dapat menggunakan aplikasi tersebut. Semakin banyak pihak yang menggunakan aplikasi identifikasi spesies tumbuhan maka pihak tersebut secara tidak langsung memberikan data gambar daun dan meningkatkan jumlah *dataset* gambar daun yang dapat digunakan untuk pembelajaran. Penggunaan aplikasi identifikasi spesies tumbuhan dari berbagai *platform* membutuhkan sistem yang terintegrasi dengan model pembelajaran sehingga sistem dapat melakukan transaksi data dengan model pembelajaran. Berdasarkan penjelasan di atas, maka peneliti mengambil judul penelitian "Pengembangan Perangkat Lunak Untuk Pengelompokkan Tumbuhan Berdasarkan Citra Digital Daun Menggunakan CNN". Dengan adanya aplikasi tersebut, diharapkan identifikasi jenis tumbuhan menggunakan citra daunnya dapat lebih akurat dan efisien.

II. METODE PENELITIAN

A. Kerangka Berpikir

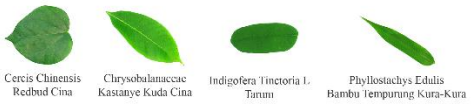
Aplikasi yang dihasilkan berupa aplikasi klasifikasi jenis tanaman berdasarkan citra digital daun tersebut. Untuk mengklasifikasi jenis tanaman, aplikasi tersebut penulis membutuhkan data dari sumber yang sudah dijelaskan pada batasan masalah. Untuk proses pengembangan aplikasinya, metodologi yang digunakan adalah inkremental. Hasil akhir aplikasi ini berupa hasil akurasi dari data gambar yang telah dilatih dan diuji secara *offline*.

Gambar 1 Kerangka Pemikiran

B. *Dataset*

Citra daun yang akan menjadi *dataset* merupakan citra dari beberapa daun yang memiliki bentuk daun yang berbeda. *Dataset* yang dikumpulkan oleh penulis didapatkan secara *opensource* yang berjudul "*Flavia Dataset*" oleh *sourceforge* melalui *link website* http://flavia.sourceforge.net/ [7]. Data yang digunakan berjumlah 1907 data gambar berdasarkan citra digital bentuk daun. Data yang digunakan di jurnal yang berjudul "*A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network*" oleh Stephen Gang Wu, Forrest Sheng Bao, Eric You Xu, Yu-Xuan Wang, Yi-Fan Chang, dan Qiao Liang Xiang [8].

Peneliti akan menggunakan 267 gambar berdasarkan 4 spesies tanaman atau kelas tanaman pada Gambar 3.



Gambar 2 *Dataset* yang digunakan berjumlah 4 kelompok kelas
(Gambar asli dari sumber [7])

C. *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah metode *deep neural network* yang biasa digunakan pada data gambar. Implementasi algoritma CNN pada gambar digunakan untuk mendeteksi dan mengenali objek pada sebuah gambar. CNN terdiri dari *input image*, *convolutional layer*, *pooling layer* dan *fully connected layer* [9, hlm. 6].

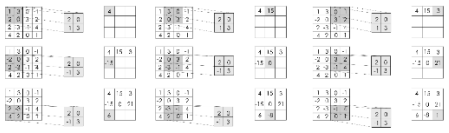
D. *Convolutional Layer*

Convolutional layer digunakan untuk mengekstrak *output feature map* dari *input feature map* dengan sebuah *filter*. Pada *convolutional layer*, kegiatan mengekstrak *output feature map* dari *input feature map* menggunakan perhitungan yang disebut operasi *convolutional* [9, hlm. 45].

2	0
-1	3

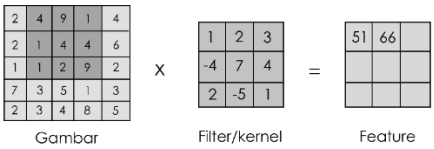
Gambar 3 *Filter* pada *Convolutional Layer*
(Digambar ulang dari sumber [9, hlm. 45])

Pada *convolutional layer* terdapat *filter* yang memiliki ukuran dan jumlah yang telah ditentukan pada awal pendefinisian *convolutional neural network*. *Filter* ini berbentuk *grid* dengan angka tertentu. Angka yang berada di dalam *filter* tersebut disebut bobot *filter*. Bobot *filter* ditentukan dengan acak pada awal pelatihan [9, hlm. 47].



Gambar 4 Proses Mengekstrak *Input Feature Map* Pada *Convolutional Layer*
(Digambar ulang dari sumber [9, hlm. 47, 48])

Proses kerja *convolutional layer* adalah dengan mengekstrak *input feature map* dengan *filter*. *Filter* digeser ke seluruh bagian dari *input feature map* dan dilakukan operasi *convolutional* dari setiap pergeseran *filter* terhadap *input* sehingga menghasilkan *output feature map*. Pergeseran *filter* diatur oleh *stride*. *Stride* adalah jarak pergeseran *filter* ke setiap sisi *input feature map* dengan satuan *grid*. *Filter* yang digunakan untuk mengekstrak *input feature map* menjadi *output feature map* dapat berjumlah lebih dari satu. Berdasarkan jumlah *filter* yang digunakan dapat lebih dari satu, proses tersebut akan terus dilakukan sebanyak jumlah *filter* yang digunakan ke setiap sisi *input feature map* [9, hlm. 47, 48].



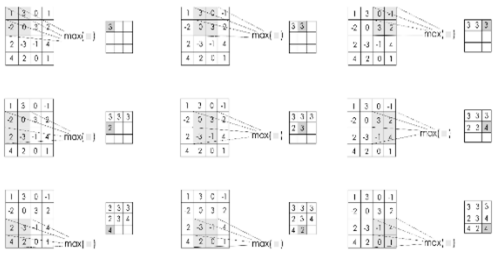
Gambar 5 Skema Operasi *Convolutional*
(Digambar ulang dari sumber [9, hlm. 47, 48])

Gambar skema operasi *convolutional* menjelaskan bahwa proses kerja diawali dengan menghitung *input feature map* dengan ukuran yang sesuai dengan ukuran *filter*. Area *input feature map* yang dihitung sesuai dengan ukuran *filter* yang digunakan. Satu kotak yang menyimpan nilai pada *input feature map* atau *filter* disebut *grid*. Perhitungan operasi *convolutional* dimulai dari *grid* yang memiliki posisi yang menunjukan identitas *grid*. Posisi kolom dan baris yang sedang dihitung disebut *current position*. *Filter* pada posisi *current position* dikalikan dengan *input feature map* pada posisi *current position*. Seluruh hasil perkalian tiap *filter grid* dengan *input map feature grid* dijumlahkan. Hasil penjumlahan tersebut ditaruh pada *output grid*. Posisi hasil penjumlahan pada *output grid* ditentukan dengan menghitung banyaknya dilakukan pergeseran *horizontal* dan vertikal. *Filter* akan dihitung ke setiap area *input feature map* dengan pergeseran sesuai *stride* yang telah ditentukan [9, hlm. 47].

E. *Pooling Layer*

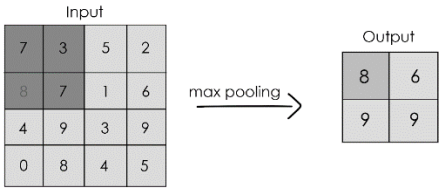
Pooling layer digunakan untuk mengekstrak *output feature map* dari *input feature map* dengan sebuah *filter*. Pada *pooling layer*, kegiatan mengekstrak *output feature map* dari *input feature map* menggunakan perhitungan yang disebut operasi *pooling*. Operasi *pooling* ini didefinisikan oleh sebuah *pooling function* seperti *average function* atau *max function*. Operasi *pooling* dengan *average function* disebut operasi *max pooling*. *Pooling layer* mirip dengan *convolutional layer*, sama-sama memiliki *filter* dan *stride*. *Filter* pada *pooling layer* memiliki ukuran lebar dan panjang yang akan digunakan untuk melakukan operasi *pooling* [9, hlm. 53].

Filter mengambil nilai yang terdapat dari *input feature map* sebanyak ukuran *filter* tersebut. *Filter* melakukan perhitungan dengan *average function* atau *max function*. Ketika menggunakan *average function*, *filter* akan menghasilkan nilai rata-rata dari *input feature map* pada *filter* tersebut. Ketika menggunakan *max function*, *filter* akan menghasilkan nilai terbesar dari *input feature map* pada *filter* tersebut [9, hlm. 53].



Gambar 6 Proses Mengekstrak *Input Feature Map* Pada *Pooling Layer*
(Digambar ulang dari sumber [9, hlm. 53])

Proses kerja *pooling layer* adalah dengan mengekstrak *input feature map* dengan *filter*. *Filter* digeser ke seluruh bagian dari *input feature map* dan dilakukan operasi *pooling* dari setiap pergeseran *filter* terhadap *input* sehingga menghasilkan *output feature map*. Pergeseran *filter* diatur oleh *stride*. *Stride* adalah jarak pergeseran *filter* ke setiap sisi *input feature map* dengan satuan *grid. Filter* yang digunakan untuk mengekstrak *input feature map* menjadi *output feature map* dapat berjumlah lebih dari satu. Berdasarkan jumlah *filter* yang digunakan dapat lebih dari satu, proses tersebut akan terus dilakukan sebanyak jumlah *filter* yang digunakan ke setiap sisi *input feature map* [9, hlm. 53].



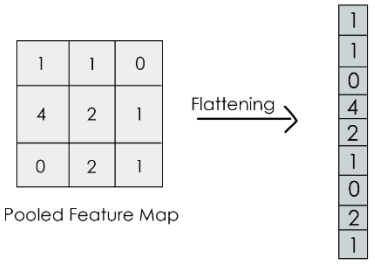
Gambar 7 Skema Operasi *Convolutional*
(Digambar ulang dari sumber [9, hlm. 51])

Skema operasi *pooling* pada Gambar 7 menjelaskan bahwa proses kerja diawali dengan menghitung *input feature map* dengan ukuran yang sesuai dengan ukuran *filter*. Area *input feature map* yang dihitung sesuai dengan ukuran *filter* yang digunakan. Satu kotak yang menyimpan nilai pada *input feature map* atau *filter* disebut *grid*. Perhitungan operasi *pooling* dimulai dari *grid* yang memiliki posisi pada ujung paling kiri *input feature map* dan ujung paling atas *input feature map*. Setiap *grid* memiliki posisi yang menunjukkan identitas *grid*.

Proses kolom dan baris yang sedang dihitung disebut *current position*. Nilai diambil sebanyak ukuran *filter* dari *input feature map*. Nilai-nilai itu dihitung sesuai *pooling function* yang digunakan. Ketika menggunakan *max pooling function* maka dilakukan operasi *max pooling* dengan cara mengambil nilai terbesar dari nilai-nilai tersebut. Ketika menggunakan *average pooling function* maka dilakukan operasi *average pooling* dengan cara mengambil nilai terbesar dari nilai-nilai tersebut. Hasil perhitungan tersebut ditaruh pada *output grid*. Proses hasil penjumlahan pada *output grid* ditentukan dengan menghitung banyaknya dilakukan pergeseran *horizontal* dan vertikal. Filter akan dihitung ke setiap area *input feature map* dengan pergeseran sesuai *stride* yang telah ditentukan [9, hlm. 53].

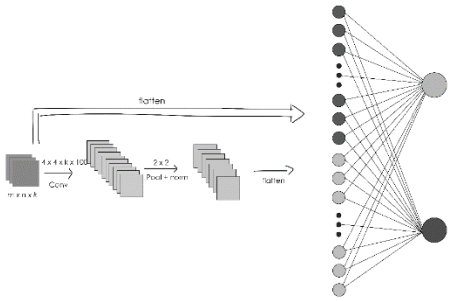
F. *Fully Connected Layer*

Fully connected layer pada *convolutional neural network* pada dasarnya sama dengan *convolutional layer* dengan *filter* berukuran 1 x 1. Tiap unit pada sebuah *fully connected layer* terhubung penuh ke semua unit pada *layer* sebelumnya. *Feature map* yang dihasilkan dari hubungan antar *pooling layer*, *relu layer*, dan *convolutional layer* menghasilkan *neuron-neuron* pada *fully connected layer*. *Neuron* tersebut digunakan untuk mengenali objek pada gambar [9, hlm. 56].



Gambar 8 Proses *Flatten* Sebelum Memasuki *Fully Connected Layer*
(Digambar ulang dari sumber [9, hlm. 53])

Pada Gambar 8 dijelaskan perubahan *feature map* sebelum memasuki *fully connected layer*. *Feature map* yang sebelum memasuki *fully connected layer* telah diproses pada *convolutional layer* atau *pooling layer*. Hasil proses pada *convolutional layer* atau *pooling layer* berupa *feature map* yang berbentuk persegi. *Feature map* yang berbentuk persegi tersebut dilakukan *flatten*, sehingga *feature map* tersebut berubah menjadi sebuah baris [9, hlm. 56].

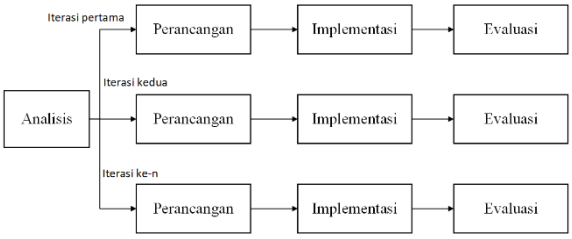


Gambar 9 Proses Mengekstraksi *Input Feature Map* Pada *Fully Connected Layer*
(Digambar ulang dari sumber [10, hlm. 6])

Pada Gambar 9, proses mengekstrak *input feature map* pada *fully connected layer* diawali dengan membuat matriks bobot. Matriks bobot memuat nilai bobot *input feature map* yang telah dilakukan *flatten* dan *output node*. Matriks tersebut memiliki jumlah baris sejumlah banyaknya *input feature map* yang telah dilakukan *flatten*. Matriks tersebut memiliki satu kolom, karena hanya ada satu *output node* pada akhir proses *convolutional neural network*. Nilai *input feature map* berasal dari hasil perhitungan *layer-layer* sebelum *fully connected layer* [9, hlm. 56].

G. Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak yang digunakan pada penelitian ini adalah metode inkremental. Model inkremental merupakan model yang digunakan proses penambahan dikit demi sedikit yang berfokus pada pengiriman modul yang bersifat operasional pada setiap tahapan. Model inkremental merupakan gabungan aliran proses linear dan paralel. Proses linear merupakan tahapan pengembangan produk dari awal hingga penyelesaian. Proses paralel merupakan peningkatan produk dari proses linear sebelumnya. Setiap proses inkremental menghasilkan suatu modul yang sudah bisa digunakan.



Gambar 10 Model Inkremental
(Sumber dari [11])

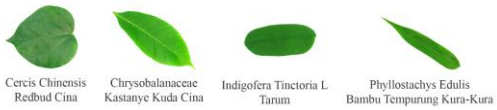
Pada Gambar 10, proses inkremental terus diulang berdasarkan proses dan modul sebelumnya, sehingga menghasilkan modul akhir yang bersifat lengkap pada proses akhir [12, hlm. 43, 44].

H. Iterasi Pertama

Pada tahap iterasi pertama ini dilakukan pembuatan logika program dan melatih algoritma berdasarkan pada data yang telah disiapkan sebelumnya. Proses iterasi pertama terdiri dari proses analisis kebutuhan perangkat penelitian,

desain logika model, implementasi desain ke dalam bentuk *code,* dan pengujian terhadap model aplikasi yang dibuat.

Proses ini dilakukan untuk menganalisis mengenai proses pengumpulan *dataset* gambar daun yang telah dibinerkan, *pre-processing dataset*, *training dataset* dan *testing* model. Berdasarkan studi literatur yang telah dilakukan, peneliti akan menggunakan data berupa 1907 jumlah gambar daun. Data gambar tersebut terdiri dari 33 spesies tanaman atau kelas tanaman (*Phyllostachys Edulis*, *Chrysobalanaceae*, *Cercis Chinensis* dan *Indigofera Tinctoria L*). *Dataset* yang akan dipisah menjadi 2 *folder* untuk pelatihan dan pengujian *dataset* seperti pada Gambar 11.



Gambar 11 Contoh Gambar Daun Yang Dilatih dan Diuji
(gambar asli dari sumber [7])

Dataset yang telah dipisah akan dilakukan *pre-processing*. Tahapan *pre-processing* ini dilakukan untuk mengubah ukuran gambar (*resizing*) menjadi 200 x 200 piksel dengan resolusi 96 dpi. Data yang digunakan akan melewati tahapan *pre-processing* terlebih dahulu. Tahapan *pre-processing* yang dilakukan terhadap data yaitu adalah proses mengubah ukuran gambar (*resizing*), mengubah gambar menjadi *grayscale,* binerisasi, segmentasi, dan normalisasi. Setelah gambar telah di-*preprocessing*, maka gambar tersebut akan diekstrak fitur-fitur daunnya. Fitur-fitur yang kita dapatkan dari proses ekstraksi yang akan digunakan pada proses pengklasifikasian. Metode pengklasifikasian yang digunakan pada penelitian ini adalah metode *Convolutional Neural Network.*

Setelah mendapatkan hasil sebuah model klasifikasi dari *library Tensorflow Keras* yang menerapkan algoritma klasifikasi *Convolutional Neural Network*, maka tahapan yang akan dilakukan adalah tahapan pengujian terhadap model klasifikasi tersebut. Setelah model klasifikasi diuji, sistem akan melakukan perhitungan nilai-nilai akurasi. Setelah perhitungan selesai, maka kita akan mendapatkan nilai akurasi dari hasil model klasifikasi kita.

Dalam membuat penelitian ini, peneliti menggunakan perangkat-perangkat yang mendukung dalam pembangunan aplikasi. Perangkat yang digunakan dalam proses pembangunan aplikasi ini dapat dilihat pada Tabel 1.

Tabel 1 Perangkat Pembangunan Aplikasi

Perangkat keras	***Perangkat Lunak***
Laptop dengan sistem operasi Windows 10 64-bit 8 GB RAM Memory AMD Ryzen 5 4500U 2.3 GHz	*Miniconda 3 Python 3.6 Numpy 1.19.5 Matplotlib 3.3.4 OpenCV untuk Python 4.5.1.48 Keras 2.4.3 Tensorflow 2.4.1 atau Tensorflow-gpu 2.4.1*

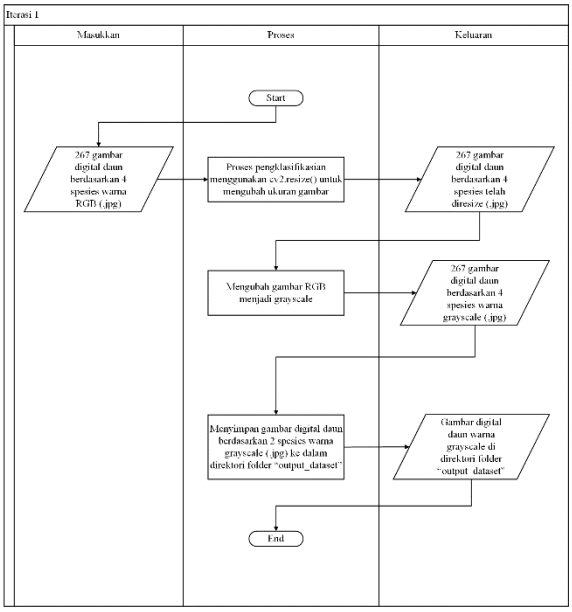
Selain perangkat pembangunan aplikasi di atas, peneliti menetapkan target kebutuhan yang diperlukan agar aplikasi bisa berjalan dengan semestinya. Berikut merupakan perangkat-perangkat yang peneliti tentukan agar aplikasi dapat berjalan dengan semestinya.

Tabel 2 Kebutuhan Aplikasi

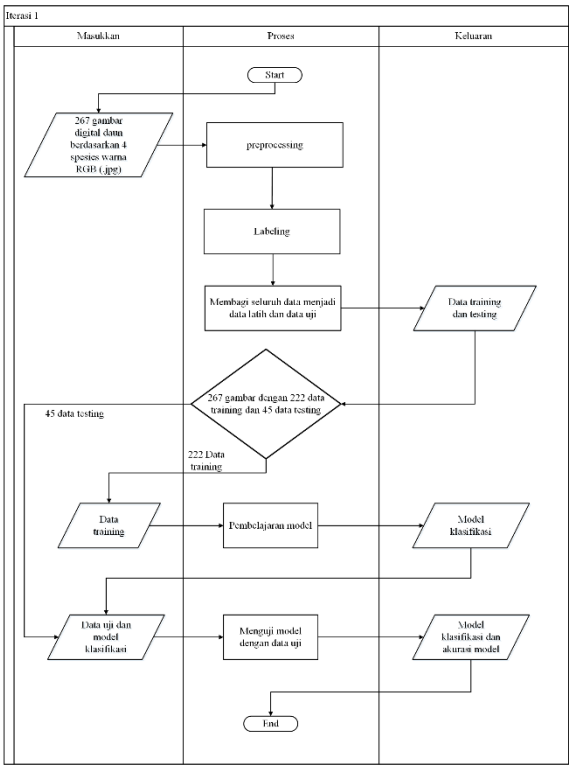
Perangkat keras	***Perangkat Lunak***
Processor: 2.0GHz *RAM: 4 GB* *HDD: 120 GB atau lebih*	*Sistem Operasi: Windows 7 atau lebih baru*

Proses ini dilakukan untuk membuat desain alur logika model. Langkah pertama ini dilakukan adalah memperoleh *dataset* mentah dari *folder directory*. Peneliti akan menggunakan 267 gambar berdasarkan 4 spesies tanaman atau kelas tanaman (*Phyllostachys Edulis*, *Chrysobalanaceae*, *Cercis Chinensis*, dan *Indigofera Tinctoria L*) kemudian, data mentah tersebut akan dipisah menjadi 2 *folder* untuk pelatihan dan pengujian untuk *pre-processing*. Pada tahap *pre-processing*, data *training* akan diubah menjadi gambar *grayscale*, binerisasi, segmentasi, normalisasi dan ekstrak struktur daunnya. Setelah gambar sudah di-*preprocessing* kemudian disimpan ke dalam direktori *folder* "*output_dataset*" untuk kemudian akan dilakukan tahapan berikutnya, data uji tersebut akan memasuki tahap *pre-processing* seperti data *training*. Setelah hasil ekstrak struktur daun kedua data didapatkan. Data latih dan data uji akan dikelompokkan menggunakan metode CNN. Setelah melakukan klasifikasi, data latih dan uji akan mendapatkan hasil akurasi dari model klasifikasi. Hasil tersebut menunjukkan seberapa baik model klasifikasi yang telah dibuat melalui akurasi yang dihasilkan.

Untuk gambar kasus RGB, data kasus yang akan digunakan utnuk menguji melalui tahap *pre-processing*. Pada tahap *pre-processing* data gambar kasus akan diubah menjadi gambar *grayscale*. Gambar daun berwarna akan digunakan sebagai data kasus untuk menguji model klasifikasi.



Gambar 12 Tahap *Pre-processing*



Gambar 13 Iterasi Pertama

Proses ini dilakukan untuk menerjemahkan desain logika yang telah disusun pada tahap sebelumnya ke dalam bahasa koding. Selain itu pada tahapan ini model yang sudah dibuat oleh desain logika di atas akan diuji ke dalam bahasa pemrogramman. Peneliti menggunakan bahasa pemrogramman *Python* dengan *IDE PyCharm* melalui nama *environment* untuk *miniconda 3* adalah "*cnnclassification*" pada penelitian ini.

Peneliti menggunakan *library cv2* dari *OpenCV* untuk melakukan tahapan *pre-processing* dan *library Tensorflow Keras* untuk menerapkan algoritma CNN. Pada proses pengujian logika model peneliti menggunakan *library Python* yaitu *Tensorflow Keras* untuk menerapkan perhitungan CNN. CNN akan digunakan untuk memprediksi hasil dari data *input*.

Terhadap logika model yang kemudian akan dibandingkan dengan label spesies tanaman yang sebenarnya apakah sesuai atau tidak sehingga didapatkan akurasi model. Kode yang berfungsi sebagai *library* yang digunakan peneliti pada penelitian ini adalah sebagai berikut.

Pada Gambar 14, Gambar 15 dan Gambar 16 merupakan kode program untuk menginisialisasi *library* yang dibutuhkan. Peneliti menggunakan *library Tensorflow Keras* dan *Numpy* untuk melakukan pelatihan model. *Library* tersebut akan digunakan peneliti untuk memudahkan dalam tahapan pengembangan aplikasi.

```
import os, glob
import cv2
import shutil

#directory
from os import listdir, makedirs
from os.path import isfile, join

import pathlib
```

Gambar 14 Kode Program Inisialisasi *Library Untuk Pre-processing*

```
#untuk direktori
import os, glob, json, sys
import cv2
import numpy as np

#library untuk membuat visualisasi hasil akurasi
import matplotlib.pyplot as plt

#optimizing
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
os.environ.pop('TF_CONFIG', None)

if '.' not in sys.path:
  sys.path.insert(0, '.')
```

Gambar 15 Kode Program Inisialisasi *Library* Untuk Proses Pemodelan *CNN*

```
#library untuk menerapkan algoritma cnn
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Conv2D, MaxPooling2D, Activation, Flatten, Dense
```

Gambar 16 Kode Program Inisialisasi *Library* Untuk Proses Pemodelan CNN

Tahapan ini yang dilakukan pengolahan citra adalah pengambilan data dari *folder* yang sudah ditetapkan. Lokasi *folder* yang sudah ditetapkan diletakkan di dalam direktori proyeknya untuk penelitian pada Gambar 17. Setelah itu data gambar yang berada pada *folder* diambil untuk didapatkan nilai RGBnya.

```
#memperoleh dataset
path_data = "full_dataset" #direktori input (data kasus)
outputdir ="output_dataset"
```

Gambar 17 Kode Program Alamat Pengambilan Data Gambar RGB

Setelah *dataset* didapatkan, *dataset* akan melakukan *pre-processing* untuk didapatkan bagian daunnya. Bagian daun yang sudah ditandai tersebut akan diubah ukuran sehingga didapatkan data latih yang lebih fokus. Kemudian data gambar RGB tersebut akan diubah menjadi *grayscale* yang berdimensi 200 x 200 piksel pada Gambar 18. Data hasil *pre-processing* tersebut disimpan ke dalam folder "*output_dataset*" dengan format gambar JPG.

```
#GRAYSCALE
def preprocessing(path_data='full_dataset', outputdir='output_dataset'):
    # unused
    files = list(filter(lambda f: isfile(join(path_data, f)), listdir(path_data)))

    # Here src_path is the location where images are saved.
    for filenames in files:
        try:
            img = cv2.imread(os.path.join(path_data, filenames))  # membaca gambar yang ada di direktori
            img = cv2.resize(img, (200, 200))  # memperkecilkan gambar
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # konversi menjadi grayscale
            hasil = join(outputdir, filenames)
            cv2.imwrite(hasil, gray)

            print('{} telah berhasil dipre-processing'.format(filenames))
        except:
            print("{} tidak bisa dikonversi!".format(filenames))

preprocessing()


print()
```

Gambar 18 Kode Program *Pre-processing* Gambar Daun Berwarna

Data latih tersebut akan digunakan untuk melatih algoritma pembelajaran *Convolutional Neural Network*. CNN yang sudah dilatih dengan data latih menghasilkan model prediksi. Proses pelatihan berlangsung selama 50 *epoch*. Kode program pelatihan dan pembuatan model dapat dilihat pada Gambar 19, Gambar 20, Gambar 21, dan Gambar 22.

```
#PROSES PELATIHAN MODEL

batch_size = 8

#CNN untuk 4 kelas
model=Sequential()
model.add(Conv2D(32,(3,3),input_shape=(200,200,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4))
model.add(Activation('softmax'))

model.summary()
```

Gambar 19 Kode Program Pembelajaran Model Klasifikasi Untuk Data Latih

```
from tensorflow.keras.optimizers import Adam

model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy',metrics=['accuracy'])
```

Gambar 20 Kode Program Pembelajaran Model Klasifikasi Untuk Data Latih

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

train_set = train_datagen.flow_from_directory('dataset/train',
                                              target_size=(200,200),
                                              batch_size=batch_size,
                                              class_mode='categorical',
                                              shuffle=True
                                              )

val_set = val_datagen.flow_from_directory('dataset/validation',
                                          target_size=(200,200),
                                          batch_size=batch_size,
                                          class_mode='categorical',
                                          shuffle = True
                                          )
```

Gambar 21 Kode Program Pembelajaran Model Klasifikasi Untuk Data Latih

```
#proses training
history = model.fit(train_set,
                    steps_per_epoch=15,
                    epochs = 50,
                    #validation_data = val_set,
                    #validation_steps= 4,
                    verbose = 1
                    )

#menyimpan model
model.save('leafClassification.h5')
```

Gambar 22 Kode Program Pembelajaran Model Klasifikasi Untuk Data Latih

```
#viewing history
# Loss Curves
plt.figure(figsize=[8, 6])
plt.plot(history.history['loss'], 'r', linewidth=3.0)
plt.legend(['Training loss'], fontsize=18)
plt.xlabel('Epochs ', fontsize=16)
plt.ylabel('Loss', fontsize=16)
plt.title('Loss Curves', fontsize=16)
plt.show()

# Accuracy Curves
plt.figure(figsize=[8, 6])
plt.plot(history.history['accuracy'], 'r', linewidth=3.0)
plt.legend(['Training Accuracy'], fontsize=18)
plt.xlabel('Epochs ', fontsize=16)
plt.ylabel('Accuracy', fontsize=16)
plt.title('Accuracy Curves', fontsize=16)
plt.show()
```

Gambar 23 Kode Program *Plot* Hasil *Loss* Dan *Accuracy* Data Latih

Setelah model dilatih kemudian nama file "*leafclassification.h5*" untuk pelatihan model yang akan disimpan di dalam direktori, model tersebut diuji dengan data uji yang sudah dibagi sebelumnya. Hasil dari pengujian model itu berupa akurasi model tersebut. Akurasi tersebut diperlukan untuk menilai performa model prediksi algoritma *Convolutional Neural Network* tersebut. Kode program untuk menguji model tersebut dapat dilihat pada Gambar 24, Gambar 25, Gambar 26, Gambar 27, dan Gambar 28. Hasil nilai akurasi dari pengujian dapat dilihat pada Gambar 29.

```
#PROSES PENGUJIAN MODEL
test_datagen = ImageDataGenerator(rescale=1. / 255,
                                  shear_range = 0.2,
                                  zoom_range = 0.2,
                                  horizontal_flip = True
                                  )
test_generator = test_datagen.flow_from_directory('dataset/test/',
                                                  target_size=(200,200),
                                                  batch_size=batch_size,
                                                  class_mode='categorical',
                                                  shuffle=True)

#mendapatkan nilai akurasi untuk data testing
#test_score = model.evaluate(test_generator)

#print("[INFO] accuracy: {:.2f}%".format(test_score[1] * 100))
#print("[INFO] Loss: ",test_score[0])
```

Gambar 24 Kode Program Pengujian Model

```
#membuat matrix confusion
# Confusion Matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Print the Target names
target_names = []
for key in train_set.class_indices:
    target_names.append(key)
print(target_names)
```

Gambar 25 Kode Program Pengujian Model

```
Y_pred = model.predict(test_generator,
                       steps=np.ceil(test_generator.samples / test_generator.batch_
                       verbose=1, workers=0)
y_pred = np.argmax(Y_pred, axis=1)

print("True label: ", test_generator.classes)
print("Predicted label: ", y_pred)
```

Gambar 26 Kode Program Pengujian Model

```
#confusion matrix
def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix', cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    plt.figure(figsize=(10,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
        print("Normalized confusion matrix")

    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
```

Gambar 27 Kode Program Pengujian Model

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

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

Gambar 28 Kode Program Pengujian Model

```
cm = confusion_matrix(test_generator.classes, y_pred)
plot_confusion_matrix(cm, target_names, title='Confusion Matrix')

# Print Classification Report
print('Classification Report')
print(classification_report(test_generator.classes, y_pred, target_names=target_names))
```

Gambar 29 Kode Program Pengujian Model

Kemudian model yang telah diuji dengan kata uji tersebut digunakan untuk memprediksi suatu kelas dari data gambar yang dimasukan. Sebelum data gambar tersebut dilakukan prediksi, data gambar tersebut diubah terlebih dahulu menjadi gambar daun *grayscale* dengan menggunakan kode program dari Gambar 18 berikut.

Setelah itu di tahap pengujian akan diuji fungsi-fungsi yang terdapat dalam logika pemrogramman dengan metode *Black-Box*.

I. Iterasi Kedua

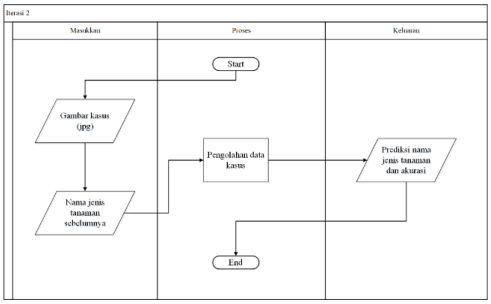
Tahapan akhir dalam pengembangan aplikasi ini adalah pembuatan tampilan untuk mempermudah pengguna dalam menggunakan aplikasi ini. Setelah inti penelitian selesai pada iterasi pertama, maka pada iterasi kedua peneliti memfokuskan aplikasi pada bagian tampilan. Tampilan antar muka akan disesuaikan dengan kebutuhan pada aplikasi ini dan sesuai dengan kebutuhan penelitian.

Pada tahapan pembuatan tampilan ini, peneliti merujuk pada kegunaan aplikasi pada tahapan iterasi sebelumnya. Aplikasi hanya berguna untuk melakukan klasifikasi dan menampilkan hasil akurasi dan nama jenis tanaman berdasarkan citra digital daun. Iterasi sebelumnya merunjukkan fungsi-fungsi untuk menghasilkan klasifikasi yang dilakukan dalam program. Dalam hal ini peneliti ingin memudahkan pengguna untuk menggunakan aplikasi.

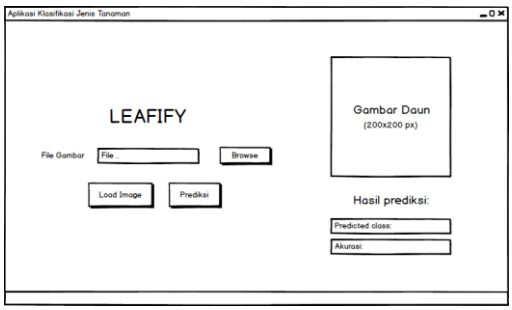
Berdasarkan hasil pada iterasi sebelumnya diketahui fungsi-fungsi apa saja yang dibutuhkan untuk penelitian ini. Pada iterasi kedua ini, peneliti membangun tampilan aplikasi yang bertujuan untuk memudahkan pengguna memakai aplikasi ini. Tampilan aplikasi yang akan dibuat merupakan tampilan antar muka pengguna yang mewakili setiap fungsi pada iterasi pertama. Tampilan antar muka yang dibuat adalah tampilan untuk fungsi memilih *path file* data gambar (data kasus), fungsi untuk memuat gambar, dan fungsi untuk memprediksi data gambar tersebut.

Dalam pengembangan tampilan aplikasi, peneliti menggunakan bahasa pemrograman *Python 3* dan *library Tkinter* untuk *Graphical User Interface* (GUI) yang dibangun. Peneliti juga menganalisis perangkat keras dan perangkat lunak yang dibutuhkan untuk membangun dan menjalankan aplikasi yang akan dibuat dapat dilihat pada Tabel 1 dan Tabel 2. Perangkat keras yang digunakan untuk membangun aplikasi dan perangkat keras minimum yang disarankan untuk menjalankan aplikasi dijelaskan pada Tabel 1. Sementara perangkat lunak yang digunakan untuk membangun aplikasi dan perangkat lunak minimum yang disarankan untuk menjalankan aplikasi dijelaskan pada Tabel 2.

Dari tahapan analisis, diketahui fungsi-fungsi yang akan dibuat pada iterasi dua ini. Fungsi pertama adalah pengambilan *path file* data gambar daun dalam format JPG. Fungsi yang kedua adalah fungsi untuk memprediksi data gambar daun yang sudah diambil dengan model yang sudah dibuat. Pada tahap ini peneliti membuat desain purwarupa berisi objek-objek sesuai fungsi yang dianalisis. Desain purwarupa dapat dilihat pada Gambar 31 berikut.



Gambar 30 Iterasi Kedua



Gambar 31 Purwarupa Aplikasi

Pada tahap implementasi peneliti merealisasikan fungsi-fungsi yang sudah dianalisis sebelumnya dan merealisasikan purwarupa yang telah didesain pada tahap sebelumnya ke dalam bentuk kode program. Terdapat tiga fungsi utama dalam aplikasi, yaitu fungsi pemilihan file data gambar, fungsi *load* data gambar dan fungsi memprediksi data gambar tersebut dengan model yang sudah dibuat.

```
#import library untuk pembuatan aplikasi GUI
import tkinter as tk
from tkinter import *
from tkinter import messagebox
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
from tkinter import ttk
import tkinter.font as tkFont

#untuk array
import numpy as np

#import library untuk foto
from PIL import Image, ImageTk
import cv2

#import library untuk sistem operasi
import os

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import load_model, Model
```

Gambar 32 Kode Program Inisialisasi *Library*

Kode program pada Gambar 32 merupakan kode program untuk menginisialisasi *library-library* yang akan digunakan untuk pengembangan aplikasi.

```
model = load_model('leafClassification.h5', compile=False)  # load model
global height, width, channel
model.summary()

load_input = model.input
input_shape = list(load_input.shape)
height = int(input_shape[1])
width = int(input_shape[2])
channel = int(input_shape[3])

print(height, width, channel)
```

Gambar 33 Kode Program *Load Model* Yang Telah Disimpan Pada Iterasi Pertama

Kode program pada Gambar 33 merupakan kode program untuk melakukan *load model* yang telah disimpan pada iterasi pertama tahap implementasi.

```
root = Tk()
root.title("Aplikasi Klasifikasi Jenis Tanaman")
root.geometry("920x550")
root.resizable(False,False)
```

Gambar 34 Kode Program Desain Aplikasi

```
fontTitle = tkFont.Font(size=40)
lbl_judul = tk.Label(root, font=fontTitle, text="LEAFIFY") #label judul
lbl_judul.place(x=180, y=150, width=230, height=60)

lbl_filejpg = tk.Label(root, text="File Gambar ") #label
lbl_filejpg.place(x=50, y=250, width=100, height=25)

txt_filejpg = tk.Text(root) #text input state="disabled"
txt_filejpg.place(x=150, y=252, width=280, height=20)

btn_filejpg = tk.Button(root, text="Browse", command=browse_jpg) #browse button
btn_filejpg.place(x=450, y = 250, height=25)

fontButton = tkFont.Font(size=13)
btn_load = tk.Button(root, font=fontButton, text="Load Image", command=load_img) #load image button
btn_load.place(x=160, y = 310, width=120, height=50)

btn_predict = tk.Button(root, font=fontButton, text="Prediksi", command=predict) #predict button
btn_predict.place(x=320, y = 310, width=100, height=50)
```

Gambar 35 Kode Program Desain Aplikasi

```
#canvas
daun_img = tk.Label(root, bg='gray', borderwidth=1, relief="solid")
daun_img.place(x=600, y=100, width=200, height=200)

#text label "Hasil prediksi"
fontTitle2 = tkFont.Font(size=14)
lbl_hasil = tk.Label(root, font=fontTitle2, text="Hasil prediksi:") #label judul
lbl_hasil.place(x=590, y=310, width=230, height=60)

text_kelas = tk.Text(root, font=tkFont.Font(size=9))
text_kelas.place(x=595, y=370, width=210, height=20)
text_kelas.insert('end', test_result_var.get())

text_akurasi = tk.Text(root, font=tkFont.Font(size=9))
text_akurasi.place(x=595, y=410, width=210, height=20)
text_akurasi.insert('end', accuracy_result_var.get())
```

Gambar 36 Kode Program Desain Aplikasi

Pada Gambar 34, Gambar 35, dan Gambar 36, kode program tersebut merupakan inisialisasi objek-objek yang digunakan untuk membentuk desain antarmuka. Objek-objek yang diinisialisasi adalah tombol "*Browse*", tombol "*Load Image*", dan tombol "Prediksi", teks masukan yang digunakan untuk menampung *path* data gambar yang sudah dipilih, label "File Gambar", label kosong yang akan diisi teks dari hasil prediksi, dan label yang digunakan untuk menampung gambar daun dari data gambar yang dipilih.

```
file_daun = tk.StringVar()
test_result_var = tk.StringVar()
accuracy_result_var = tk.StringVar()
```

Gambar 37 Kode Program Variabel *Tkinter* Untuk 3 Fungsi

```
def browse_jpg():
    global filename, img, file_daun
    filename = filedialog.askopenfilename(
        initialdir=os.getcwd(), title="Pilih file daun jpg",
        filetypes=(("JPG files", "*.jpg"), ("all files", "*.*")))
    file_daun.set(filename)
    txt_filejpg.delete("1.0", tk.END)
    txt_filejpg.insert('end', file_daun.get())
    inputValue = txt_filejpg.get("1.0", "end-1c")
    img = Image.open(filename)
    img = img.resize((200,200))
    tkimage = ImageTk.PhotoImage(img)
    daun_img.configure(image=tkimage)
    daun_img.image = tkimage
```

Gambar 38 Kode Program Memilih File Data Gambar

Gambar 38 merupakan kode program untuk memilih *path* tempat *dataset* gambar dalam format JPG berada. Berkas gambar tersebut nantinya akan digunakan sebagai masukan pada fungsi yang kedua, yaitu fungsi *load image* pada Gambar 39.

```
def load_img():
    path = file_daun.get()
    global imgs
    imgs = cv2.imread(os.path.join(path))
    imgs = cv2.cvtColor(imgs, cv2.COLOR_BGR2RGB)
    imgs = cv2.resize(imgs, (height, width))
    messagebox.showinfo("Message", "Image loaded!")
    return
```

Gambar 39 Kode Program *Load Image*

```
def predict():
    labels = ['Phyllostachys Edulis', 'Chrysobalanaceae', 'Cercis Chinensis', 'Indigofera Tinctoria L']
    predict_image = tf.keras.preprocessing.image.img_to_array(imgs)
    predict_image = (np.array(predict_image))/255
    predict_image = np.expand_dims(predict_image, axis=0)
    predict_image = np.vstack([predict_image])
    pred = model.predict(predict_image)
    print(pred)
    accuracy_text = "Akurasi: {:.2f}" .format(100 * np.max(pred))
    result_text = "Predicted class: " + str(labels[np.argmax(pred)])

    test_result_var.set(result_text)
    text_kelas.delete("1.0", tk.END)
    text_kelas.insert('end', test_result_var.get())
    inputValue = text_kelas.get("1.0", "end-1c")

    accuracy_result_var.set(accuracy_text)
    text_akurasi.delete("1.0", tk.END)
    text_akurasi.insert('end', accuracy_result_var.get())
    inputValue = text_akurasi.get("1.0", "end-1c")
    return pred
```

Gambar 40 Kode Program Prediksi

Kode program pada Gambar 40 merupakan fungsi yang digunakan untuk memprediksi data gambar dalam format JPG yang telah dipilih sebelumnya. Fungsi tersebut akan mengeluarkan kelas prediksi yang terdiri dari teks "*Phyllostachys Edulis*", "*Chrysobalanaceae*", "*Cercis Chinensis*" dan "*Indigofera Tinctoria L*".

Setelah itu di tahap pengujian akan diuji fungsi-fungsi yang terdapat dalam logika pemrogramman dengan metode *Black-Box*.

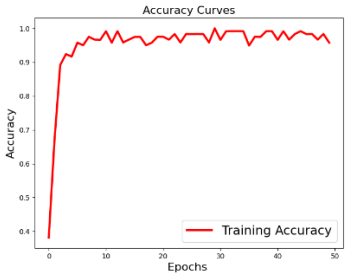
III. HASIL & PEMBAHASAN

Tahap awal pada iterasi pertama adalah melakukan proses *pre-processing dataset* gambar yang berjumlah 267 gambar daun berdasarkan bentuk daun dan nama jenis tanaman tersebut tersimpan dalam format JPG file dan sebanyak 222 gambar daun digunakan sebagai data latih dan sebanyak 45 gambar daun yang digunakan sebagai data uji dan sebanyak 27 gambar daun digunakan sebagai data kasus. Ketiga kategori data tersebut dilakukan *pre-processing* terlebih dahulu sebelum maju ke tahap selanjutnya. Proses *pre-processing* disini adalah mengubah gambar daun menjadi gambar daun dengan rentang warna grayscale yang berdimensi 200 x 200 piksel. Data hasil *pre-processing* tersebut disimpan ke dalam *folder* "*output_dataset*" dengan format gambar

JPG. Hasil *pre-processing* gambar dapat dilihat pada Gambar 41.

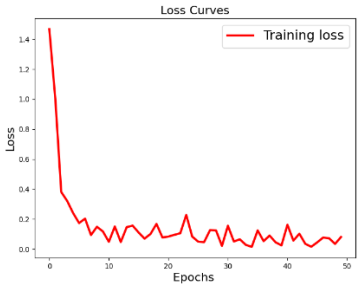
Gambar 41 Hasil *Pre-processing* Gambar Daun *Grayscale*

Setelah proses *pre-processing*, data gambar daun yang merupakan data latih digunakan untuk melatih model. Lalu, pada akhir pelatihan, terdapat grafik *log* yang mengilustrasikan proses pelatihan setiap *epoch*-nya. Grafik tersebut dilihat pada Gambar 42 berikut:



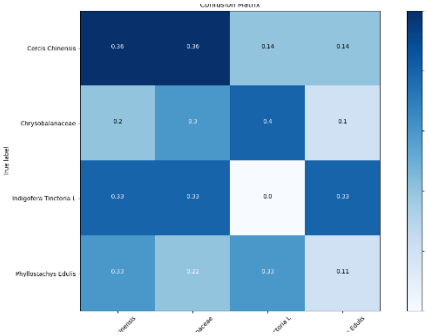
Gambar 42 Grafik *Log Accuracy* Selama Proses Pelatihan

Pada Gambar 42, grafik tersebut memiliki *axis* x yang menunjukkan *epoch* dan *axis* y yang menunjukkan nilai akurasi yang didapatkan selama proses pembelajaran. Grafik *loss* selama proses pelatihan juga dapat dilihat pada Gambar 27, yang dimana *axis* x merupakan *epoch* dan *axis* y merupakan nilai *loss*. Semakin kecil nilai *loss* yang dimiliki oleh sebuah model maka akan semakin baik model tersebut. Jadi nilai akurasi dari 222 data gambar sebagai data latih pada 50 *epoch* adalah 96.29%.



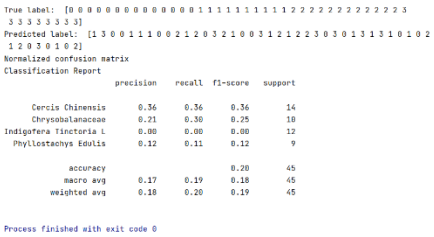
Gambar 43 Grafik *Log Loss* Selama Proses Pelatihan

Setelah model berhasil dilatih, model tersebut diuji menggunakan data uji sebanyak 45 gambar daun *grayscale* sebagai data uji. Hasil prediksi dari model dengan data uji tersebut dicocokan dengan label sebenarnya sehingga menghasilkan nilai akurasi. Perhitungan akurasi tersebut adalah dengan cara menghitung jumlah data yang benar dibagi jumlah data uji yang diprediksi oleh model tersebut.



Gambar 44 Hasil Pengujian Ditempatkan Pada *Confusion Matrix*

Pada Gambar 44 menunjukkan diagram *confusion matrix*. Nilai akurasi merupakan jumlah prediksi benar dibagi total jumlah prediksi. Selain nilai akurasi, didapat juga hasil perhitungan dari *precision*, *recall*, *f-measure* (*f1-score*), dan *support*. Output pada program merupakan hasil akurasi pada model prediksi dapat dilihat pada Gambar 45 berikut.



Gambar 45 Hasil Pengujian Ditempatkan Pada *Confusion Matrix*

Akurasi dihitung menggunakan *confusion matrix* berdasarkan rumus pada di bawah ini.

$$Accuracy = \frac{Klasifikasi\ data\ benar}{Total\ jumlah\ data\ klasifikasi}\ x\ 100\%$$

Atau akurasi dengan rumus *Confusion Matrix:*

TP = *True Positive* (positif tepat)

FP = *False Positive* (positif keliru)

TN = *True Negative* (negatif tepat)

FN = *False Negative* (negative keliru)

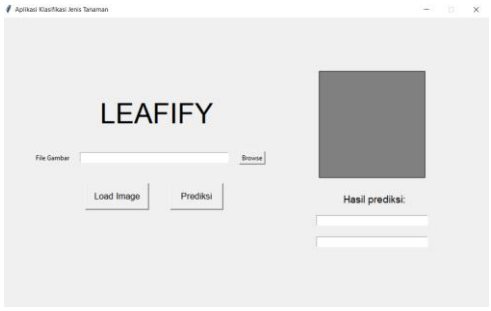
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}\ x\ 100\%$$

$$Accuracy = \frac{0.36 + 0.3 + 0.0 + 0.11}{3.98}\ x\ 100\%$$

$= 19{,}34\% \sim 20\%$

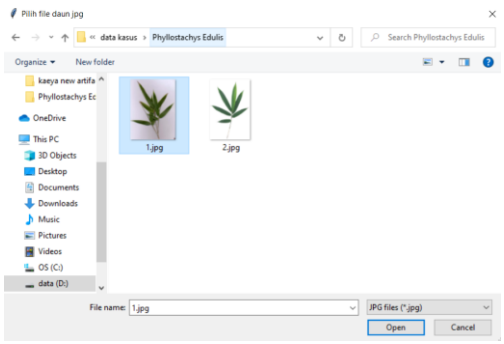
Berdasarkan Gambar 45, hasil akurasi yang didapat adalah sebesar 20%. Hasil tersebut didapat dengan cara memprediksi setiap data uji yang sudah diberi label sebelumnya. Kelas label berdasarkan urutan adalah "*Cercis Chinensis*" merupakan angka 0, "*Chrysobalanaceae*" merupakan angka 1, "*Indigofera Tinctoria L.*" merupakan angka 2 dan "*Phyllostachys Edulis*" merupakan angka 3. "*True label*" dalam Gambar 14 merupakan label kelas yang benar pada setiap data uji, sementara "*Predicted label*" dalam Gambar 27 merupakan label kelas hasil prediksi dari model. Jumlah prediksi yang tepat adalah 9 dan jumlah prediksi yang keliru adalah 36, maka dari itu didapatlah akurasi sebesar 20%.

Hasil ini berupa hasil tampilan desain aplikasi dengan menggunakan *library Tkinter* untuk memproduksi antarmuka pengguna atau yang biasanya disebut GUI (*Graphical User Interface*). Tampilan awal aplikasi dapat dilihat pada Gambar 46.



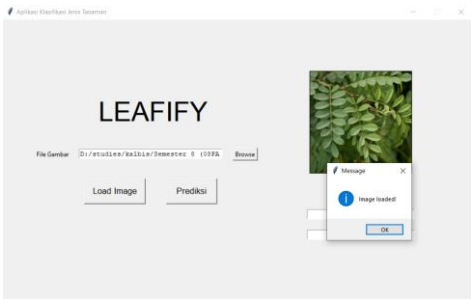
Gambar 46 Tampilan Awal Aplikasi

Pada Gambar 46, tombol "*Browse*" digunakan untuk menjalankan fungsi pemilihan berkas data gambar. Tampilan pemilihan berkas data gambar dapat dilihat pada Gambar 47 berikut.

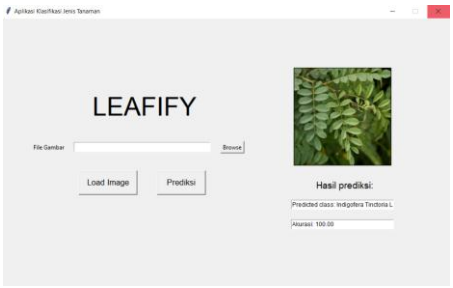


Gambar 47 Pemilihan Berkas Data Gambar

Setelah memilih data gambar, lokasi dari data gambar tersebut otomatis terambil dari tercetak pada teks masukan di sebelah tombol "*Browse*". Setelah data gambar dipilih, tekan tombol "*Load Image*" untuk memuat gambar yang telah dipilih kemudian munculnya dialog pesan "*Image loaded!*" pada Gambar 48, setelah itu tekan tombol "Prediksi" untuk melakukan prediksi data gambar yang telah dipilih. Hasil akhir pada tampilan aplikasi dapat dilihat pada Gambar 49.



Gambar 48 Pemilihan Berkas Data Gambar



Gambar 49 Tampilan Hasil Akhir Pada Aplikasi

Peneliti membuat tampilan aplikasi tersebut menggunakan *library Tkinter*. Proses pembuatan tampilan aplikasi tersebut adalah dengan menginisialisasi tampilan-tampilan antarmuka terlebih dahulu. Lalu setelah tampilan-tampilan antarmuka sudah jadi, peneliti baru membuat aksi atau fungsi pada setiap objek. Pembuatan aplikasi harus memikirkan kesederhanaan pada tampilan dan kemudahan pemakaian aplikasi bagi pengguna. Dalam mempertimbangkan kesederhanaan tampilan, peneliti hanya menyediakan 4 buah objek pada tambilan aplikasi dan dalam mempertimbangkan kemudahan pemakaian aplikasi, peneliti hanya menyediakan 3 buah aksi yang digunakan oleh pengguna, yaitu tombol "*Browse*", tombol "*Load Image*" dan tombol "Prediksi". Pengguna dapat menekan tombol "Prediksi" ketika pengguna sudah melakukan aksi untuk menentukan lokasi berkas gambar yang ingin diprediksi dengan tombol "*Browse*" dan tombol "*Load Image*". Jika pengguna belum melakukan aksi untuk menentukan lokasi berkas gambar daun, maka tombol "*Load Image*" dan tombol "Prediksi" tidak dapat ditekan. Kemudian munculnya hasil kelas dan akurasi yang telah diprediksi pada kotak *input* di atas teks "Hasil Prediksi".

IV. SIMPULAN

Berdasarkan Analisa, percobaan, dan hasil yang telah didapatkan, maka dapat dirumuskan kesimpulan sebagai berikut:

1. Pada penelitian ini, metode *Convolutional Neural Network* dapat digunakan untuk melakukan klasifikasi jenis tanaman berdasarkan citra digital daun.

2. Pokok penelitian ini adalah pengembangan aplikasi

klasifikasi tanaman berdasarkan citra digital daun untuk melakukan klasifikasi jenis tanaman berdasarkan citra digital daun.

3. Proses pengembangan aplikasi klasifikasi jenis tanaman berdasarkan citra daun berwarna untuk diproses, melakukan *pre-processing* dan disimpan ke JPEG. Data yang disimpan ke JPEG tersebut diproses lagi ke model *Convolutional Neural Network* menggunakan *library Tensorflow Keras.*

4. Data yang dihasilkan berjumlah 267 yang kemudian dibagi untuk dilatih dengan menjadi data latih dan data uji. Hasil akurasi dari algoritma *Convolutional Neural Network* yang telah dibuat pada 50 *epoch* untuk data latih bernilai 96.29%, dan untuk data uji adalah 20%. Akurasi pada data latih dan data uji berbeda jauh karena penggunaan *dataset* dengan jumlah data yang sedikit, banyak data di setiap kelompok tidak sama.

5. Dari pengujian terhadap model yang telah dilakukan, terdapat jumlah kelas gambar daun yang tepat sebanyak 9 dan jumlah kelas gambar daun yang keliru sebanyak 36 dari keseluruhan data uji yang berjumlah 45. Nama kelas gambar daun yang tepat adalah ketika model berhasil mengubah nama kelas gambar daun "*Cercis Chinensis*" ke dalam teks "*Cercis Chinensis*", sedangkan nama kelas gambar daun yang salah adalah ketika model mengubah nama kelas gambar daun "*Cercis Chinensis*" ke dalam teks selain "*Cercis Chinensis*", misalnya teks "*Phyllostachys Edulis*".

DAFTAR RUJUKAN

[1] A. F. Saiful Rahman, A. A. B, and S. D. Kurniawan, "Identifikasi Citra Daun Dengan Menggunakan Metode *Deep Learning Convolutional Neural Network* (CNN)," *J. Tek. Elektro Uniba (JTE Uniba)*, vol. 4, no. 1, hlm. 23–28, 2019, doi: 10.36277/jteuniba.v4i1.55.

[2] Felix, J. Wijaya, S. P. Sutra, P. W. Kosasih, and P. Sirait, "Implementasi *Convolutional Neural Network* Untuk Identifikasi Jenis Tanaman Melalui Daun," *J. SIFO Mikroskil*, vol. 21, no. 1, hlm. 1–10, 2020.

[3] I. Jamaliah, "Identifikasi Jenis Daun Tanaman Obat Hipertensi Berdasarkan Citra RGB Menggunakan Jaringan Syaraf Tiruan," *Penelit. Ilmu Komput. Sist. Embed. dan Log.*, vol. 5, no. 1, hlm. 1–11, 2017.

[4] J. Wäldchen, M. Rzanny, M. Seeland, and P. Mäder, "*Automated Plant Species Identification—Trends and Future Directions*," *PLOS Comput. Biol.*, vol. 14, no. 4, hlm. 1–19, 2018, doi: 10.1371/journal.pcbi.1005993.t002.

[5] W. S. Eka Putra, "Klasifikasi Citra Menggunakan *Convolutional Neural Network* (CNN) pada *Caltech 101*," *J. Tek. ITS*, vol. 5, no. 1, 2016, doi: 10.12962/j23373539.v5i1.15696.

[6] S. Ilahiyah and A. Nilogiri, "Implementasi *Deep Learning* Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan *Convolutional Neural Network*," *JUSTINDO (Jurnal Sist. dan Teknol. Inf. Indones.*, vol. 3, no. 2, hlm. 49–56, 2018.

[7] S. G. Wu, F. S. Bao, E. Y. Xu, Y. Wang, Y. Chang, and Q. Xiang, *Flavia Dataset - Sourceforge*. http://flavia.sourceforge.net/ (Diakses pada tanggal 10 Agustus 2021).

[8] S. G. Wu, F. S. Bao, E. Y. Xu, Y. Wang, Y. Chang, and Q. Xiang, "*A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network*," *IEEE 7th Int. Symp. Signal Process. Inf. Technol.*, hlm. 1–6, 2007.

[9] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "*A Guide to Convolutional Neural Networks for Computer Vision*," *Synth. Lect. Comput. Vis.*, vol. 8, no. 1, hlm. 1–207, 2018, doi: 10.2200/s00822ed1v01y201712cov015.

[10] L. Zhang and P. N. Suganthan, "*Visual Tracking with Convolutional Random Vector Functional Link Network*," *IEEE*

Trans. Cybern., vol. 47, no. 10, hlm. 3243–3253, 2017, doi: 10.1109/TCYB.2016.2588526.

[11] Javatpoint, *Javatpoint: Incremental Model*. https://www.javatpoint.com/software-engineering-incremental-model (Diakses pada tanggal 16 Maret 2021).

[12] R. S. Pressman and B. R. Maxim, *Software Engineering: Practitioner's Approach (Eighth Edition)*. New York, United States: Raghu Srinivasa, 2015.