Implementasi Metode *Convolutional Neural Network* untuk Mendeteksi Penyakit pada Citra Daun Tomat

Implementation of Convolutional Neural Network Method to Detect Diseases in Tomato Leaf Image

Chrisno R. Kotta[1)], Debby Paseru[2)], Michael Sumampouw[3)]

[1,2,3]Teknik Informatika, Universitas Katolik De La Salle Manado
[1,2,3]Kairagi I Kombos Manado - 95253

16013088@unikadelasalle.ac.id[1)], dpaseru@unikadelasalle.ac.id[2)], msumampouw@unikadelasalle.ac.id[3)]

Diterima : 23 November 2022 || Revisi : 19 Desember 2022 || Disetujui: 21 Desember 2022

Abstrak – Tanaman tomat (*lycopersicum esculentum mill*) merupakan salah satu komoditas sayuran yang berpotensi multiguna dan menjadi penyumbang ekspor. Salah satu penyebab utama penurunan hasil produksi padai tanaman tomat, yaitu munculnya berbagai macam penyakit. Tanaman dikatakan terkena penyakit jika ada perubahan pada seluruh atau sebagian organ tanaman yang menyebabkan terganggunya kegiatan fisiologis sehari-hari. Penelitian ini akan menggunakan metode *deep learning* dan algoritma *Convolutional Neural Network* (CNN) untuk melakukan identifikasi penyakit pada tanaman tomat melalui daun. Pelatihan model CNN akan dilakukan menggunakan menggunakan bahasa pemrograman *Python* dan dilakukan pada *platform Google Colab,* sedangkan pembangunan aplikasi berbasis android menggunkan Android Studio. Pengujian telah dilakukan dengan mengimplementasikan berbagai skenario uji, yaitu pengujian dengan sumber gambar dari galeri dan sumber gambar langsung dari kamera. Hasilnya adalah aplikasi yang dibangun cukup reliabel dengan akurasi uji pada gambar dari galeri sebesar 94% dan 80% akurasi untuk pengujian menggunakan gambar diambil langsung dari kamera.

Kata Kunci: penyakit tanaman tomat, daun tomat, CNN, *deep learning*

Abstract *– Tomato (lycopersicum esculentum mill) is one of the leading commodities that has the potential to be a contributor to exports. One of the main causes of decreased production of tomato plants, namely the emergence of various diseases. Plants are said to be affected by disease if there are changes in all or part of the plant organs that cause disruption of daily physiological activities. This study will use deep learning methods and Convolutional Neural Network (CNN) algorithms to determine disease in tomato plants through leaves. The CNN training model will be carried out using the Python and carried out on the Google Colab platform, while the Android-based application development will use Android Studio. Tests have been carried out by implementing various test scenarios, namely testing with image sources from the gallery and image sources directly from the camera. The result is an application that is built quite reliably with an accuracy of testing on images from the gallery of 94% and 80% accuracy for testing using images taken directly from the camera.*

Keywords: *diseases of tomato plants, tomato leaf, CNN, deep learning*

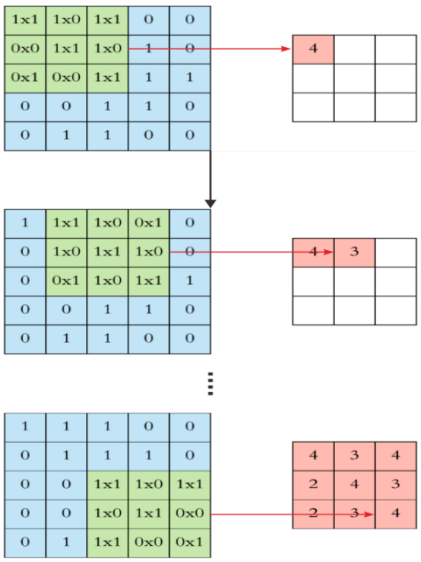
PENDAHULUAN

Pertanian memegang peranan penting dalam proses meningkatkan ekonomi bangsa Indonesia. Secara umum, sektor pertanian terdiri atas beberapa subsektor, yakni subsektor hortikultura, perkebunan, dan pangan. Subsektor hortikultura (buah dan sayuran) merupakan bagian dari sektor pertanian yang cukup penting dan merupakan salah satu penyumbang peningkatan Produk Domestik Bruto (PDB). Tanaman tomat (*Lycopersicum esculentum Mill*) adalah salah satu produk pertanian yang memiliki potensi besar dan dapat dimanfaatkan sebagai sayuran serba guna (Kusumaningrum, 2019). Terdapat berbagai macam penyakit pada tanaman tomat, ada yang disebabkan oleh cendawan, bakteri, virus, dan lain-lain. Penyakit yang ada pada tanaman tomat sebagian besar terjadi pada daun tomat (Wati et al., 2021). Berdasarkan fakta yang ada, maka penelitian ini menjadikan daun tomat sebagai objek untuk mengidentifikasi penyakit pada tanaman tomat.

Deep learning merupakan bidang pembelajaran mesin (*machine learning*) yang mengimplementasikan metode pembelajaran. Data tersebut dimasukkan pada lapisan-lapisan pembelajaran berkelanjutan yang

terus meningkat seiring proses pembelajaran . *Convolutional Neural Network* (CNN) merupakan jenis algoritma jaringan saraf tiruan yang biasa digunakan untuk melakukan klasifikasi maupun pengenalan objek pada data berupa gambar (Abdillah et al., 2022).

Dalam *deep neural network* terdapat bagian yang disebut *Convolutional Neural Network* (CNN), merupakan suatu jenis jaringan saraf tiruan yang biasanya dimanfaatkan dalam melakukan pengidentifikasian dan pemrosesan citra. Konvolusi CNN adalah sebuah operasi matematika pada dua fungsi yang kemudian menghasilkan fungsi ketiga. CNN menggunakan konvolusi sebagai pengganti untuk perkalian matriks umum. Operasi ini digunakan setidaknya sekali di setiap lapisan (Batubara & Awangga, 2020).



Gambar 1 Operasi Konvulusi (Batubara & Awangga, 2020)

Penelitian terkait telah dilakukan sebelumnya membahas tentang proses implementasi beberapa metode penyelesaian. Pertama, citra yang awalnya berwarna RGB (*Red, Green, Blue*) ditransformasikan ke HSV (*Hue, Saturation, Value*). Selanjutnya, dilakukan ekstraksi fitur tekstur menggunakan GLCM (*Gray Level Co-Occurrence Matrix*). Citra hasil akan diklasifikasikan dengan CNN dan SVM (*Support Vector Machine*). Setelah dilakukan klasifikasi kedua algoritma akan dibandingkan untuk menentukan *classifier* yang lebih baik dalam mendeteksi penyakit pada daun tomat secara otomatis. Nilai akurasi dari metode CNN menghasilkan tingkat akurasi sebesar 97,5%. Kemudian, citra daun tomat yang telah terdeteksi penyakit akan diberikan informasi berupa solusi penangan dalam mengatasi penyakit yang menyerang daun tomat tersebut (Felix et al., 2019).

Penelitian terkait lain yang membahas tentang proses implementasi algoritma *random forest* menggunakan tiga kombinasi ekstraksi fitur yaitu fitur warna, fitur *Hu-Moment*, dan fitur *haralick* pada citra penyakit daun tomat dengan menggunakan 10 kelas penyakit. Pada penelitian ini diperoleh hasil akurasi yaitu 96% (Khultsum & Subekti, 2021).

Berdasarkan penelitian yang telah dilakukan sebelumnya dapat disimpulkan bahwa algoritma CNN memperoleh nilai akurasi tertinggi yaitu 97.5% dibandingkan algoritma *random forest* dengan akurasi 96%. Kedua penelitian tersebut masih menggunakan *platform* berbasis desktop, sedangkan yang dikembangkan pada penelitian ini adalah berbasis android. A Penelitian yang menggunakan CNN hanya mengklasifikasi 4 kelas penyakit, sedangkan dalam penelitian ini mengidentifikasi 9 kelas penyakit dan 1 kelas daun sehat. Jenis penyakit yang akan diidentifikasi pada tanaman tomat adalah *Bacterial Spot, Early Blight, Late Blight, Leaf Mold, Septoria Leaf, Tomato Two Spotted Spider Mite, Target Spot, Tomato Mosaic Virus, Tomato Yellow Leaf Curl*. Dari berbagai penyakit tersebut, disebabkan oleh jamur, virus, dan serangga, serta didukung oleh kondisi cuaca di lingkungan penanaman (Andrian & Maretta, 2017).

Tujuan dari penelitian ini adalah implementasi metode CNN untuk mendeteksi penyakit pada tanaman tomat berdasarkan citra daun. Penerapan pada aplikasi android mempermudah dalam melakukan identifikasi karena bersifat *mobile*. Sumber citra daun sebagai objek yang akan diidentifikasi dapat diambil dari galeri yang sudah disimpan ataupun dapat diambil secara langsung melalui kamera. Hasil yang diperoleh setelah melakukan identifikasi penyakit tanaman tomat berdasarkan kelas yang sudah ditentukan, yaitu terdapat 10 kelas yang terdiri atas 9 kelas penyakit tanaman tomat dan 1 kelas daun sehat.

METODOLOGI PENELITIAN

Metodologi penelitian yang digunakan sebagai aturan untuk membangun perangkat lunak adalah metode *Waterfall*. Metode *Waterfall* memiliki beberapa tahapan menurut (Tristianto, 2018) yaitu.

1. *Requirement Analisis* adalah tahapan *developer* perangkat lunak membutuhkan pembelajaran

atau studi lebih lanjut mengenai aplikasi atau sistem yang akan dibangun. Pengumpulan informasi pada umumnya dilakukan dengan cara studi pustaka, wawancara atau survei langsung di lapangan. Informasi yang didapatkan dianalisis untuk mendapatkan data. .

2. *System Design* adalah tahapan lanjutan berdasarkan hasil analisis, pengembang dapat menyiapkan desain dari sistem atau aplikasi yang akan dibangun.
3. *Implementation* pada tahap ini berupa hasil dari desain sistem yang sudah dibuat pada fase sebelumnya dan akan diimplementasikan dengan pengembangan aplikasi.
4. *Intergration and Testing* merupakan tahap sistem yang telah dibangun akan diintegrasikan sehingga menjadi suatu kesatuan. Setelah melakukan integrasi, seluruh sistem akan diuji dengan berbagai kasus uji untuk melakukan pengecekan kegagalan maupun kesalahan aplikasi.
5. *Operation and Maintenance* merupakan fase terakhir jika terdapat ketidaksesuaian, maka akan dilakukan pengkajian kembali mengenai sistem yang telah dibangun.

Analisis sistem

Metode yang digunakan pada penelitian ini untuk mendeteksi objek memiliki tiga tahapan, yaitu *input*, proses, dan *output*:

1. *Input*

Input yang digunakan adalah citra RGB yang diambil *platform kaggle* yang terdiri atas sepuluh kelas dapat dilihat pada tabel 1.

Tabel 1 *Dataset* Daun Tomat

No	Nama Penyakit	Jumlah
1	Tomat *Bacterial spot*	1.701
2	Tomat *Early blight*	1.813
3	Tomat *healthy*	1.882
4	Tomat *Late blight*	1.851
5	Tomat *Leaf Mold*	1.882
6	Tomat *Septoria leaf spot*	1.745
7	Tomat *Spider mites Two spotted spider mite*	1.741
8	Tomat *Target_Spot*	1.795
9	Tomat *Mosaic virus*	1.790
10	Tomat *Yellow Leaf Curl Virus*	1.960
Jumlah		18.160

Pada Tabel 1 terdapat sepuluh kelas dengan total 18.160 citra. Pada penelitian ini data dibagi menjadi tiga bagian, yakni 85% data sebanyak 15.436 citra dibuat sebagai data *training*, dan 10% data sebanyak 2043 citra dibuat sebagai data validasi, serta 5% data sebanyak 681 citra dibuat sebagai data uji.

2. Proses

Pada tahap ini proses pengolahan citra penyakit tanaman tomat telah diperoleh. Proses CNN yang dilakukan terdiri atas *resize*, pelabelan *dataset*, *generate* data, *training*, *testing* dan evaluasi, konversi model, serta pengodean aplikasi android.

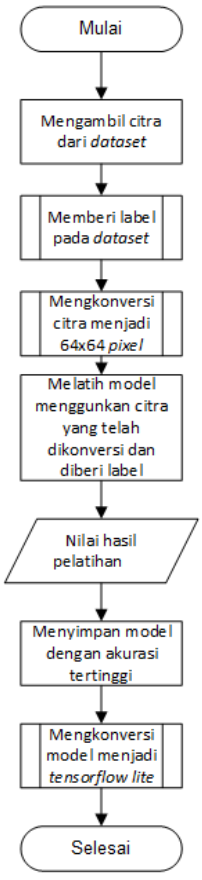
a. *Resize*, tahap awal proses *preprocessing* yaitu data citra diambil dari *dataset* yang tersedia kemudian dilakukan *resizing* untuk mengubah ukuran citra. D Selain itu untuk memperkecil ukuran citra secara *horizontal* dan/atau *vertical*, maka gambar akan di-*resize* agar ukuran citra tidak lebih dari 64 x 64 *pixel*. Hal ini dilakukan untuk membuat proses *training* lebih ringan dan cepat.
b. Pelabelan *Dataset*, data citra yang telah di-*resize* kemudian diberi label berdasarkan nama *folder* data tersebut disimpan, misalnya data dari *folder* Tomat *bacterial spot* akan diberi label Tomat *bacterial* Spot.
c. *Generate* data, data citra yang telah diberi label kemudian akan dibagi secara acak dengan presentasi yang telah ditentukan, yakni 85% data *training, 10%* data validasi, dan 5% data uji.
d. *Training*, data citra yang telah diberi label dan dibagi sesuai presentase digunakan untuk melatih model secara terus-menerus sampai selesai.
e. *Testing* dan evaluasi, model yang telah terbentuk dari proses *training* kemudian dilakukan pengujian apakah model dapat mengidentifikasi penyakit secara akurat. Berdasarkan hasil *testing*, maka evaluasi akan dilakukan.
f. Konversi model, model yang telah diuji dan dievaluasi kemudian dikonversi menjadi *TensorFlow Lite* agar bisa digunakan di Android Studio.
g. Pengodean aplikasi android, model CNN yang telah dikonversi ke dalam bentuk *TensorFlow Lite* dimuat ke dalam Android Studio dan dilakukan pengodean program agar model CNN dapat dijalankan pada *platform* berbasis android.

3. *Output*

Output dari serangkaian proses yang telah dilakukan adalah aplikasi android yang dapat menampilkan hasil probabilitas identifikasi penyakit tomat berdasarkan citra daun.

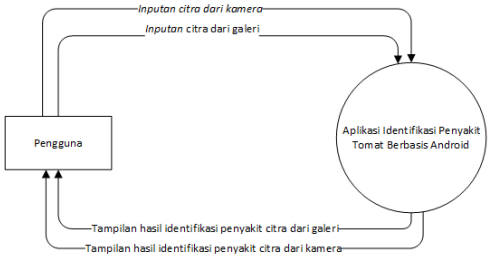
Desain Sistem

Pada tahap ini berisi gambaran pemodelan perangkat lunak yang digunakan untuk menggambarkan alur kerja dari aplikasi android identifikasi penyakit tanaman tomat dan alur kerja proses melatih model CNN. Proses kerja aplikasi dan *training* model CNN akan dijelaskan dengan menggunakan DFD dan *flowchart*.



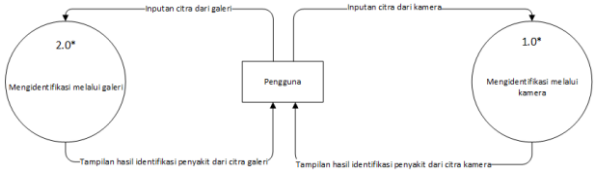
Gambar 2 *Flowchart Training Model* CNN

Gambar 2 memperlihatkan aktivitas yang terjadi dalam melakukan *training* model CNN. Model dimulai dengan memuat *dataset* citra penyakit yang ada pada daun tomat. *Dataset* citra yang telah dimuat kemudian diubah dimensi citranya menjadi 64 x 64 *pixel*. *Dataset* yang telah diubah dimensinya kemudian diberi label sesuai nama penyakitnya kemudian data yang telah diberi label digunakan untuk melatih model CNN. Setelah proses *training* selesai, maka akan ditampilkan hasil *training tersebut* model kemudian akan dikonversi kedalam versi *tensorflow lite* yang akan digunakan untuk pembuatan aplikasi berbasis android.



Gambar 3 DFD Level 0

Pada Gambar 3 dijelaskan mengenai interaksi pengguna dengan perangkat lunak secara umum, pengguna memasukkan citra ke aplikasi dan aplikasi memberikan hasil identifikasi penyakit.



Gambar 4 DFD Level 1

Pada Gambar 4 DFD Level 1 menjelaskan alur pengguna harus memasukkan citra daun tomat melalui kamera atau melalui galeri, kemudian aplikasi akan memberikan data citra tersebut ke model yang telah dilatih dan aplikasi akan menampilkan hasil identifikasi yang berisi nama penyakit serta tingkat akurasi dan waktu yang dibutuhkan untuk identifikasi.

Implementasi Sistem

Imlementasi aplikasi identifikasi penyakit pada tanaman tomat melalui citra daun. Hal ini menggunakan algoritma CNN yang dibangun dengan menggunakan bahasa pemrograman *python,* dengan bantuan infrastruktur *google colab* dan implementasi ke dalam Android Studio menggunakan model *tensor flow lite* dari hasil pelatihan di *google colab*. Kemudian dikemas menggunakan bahasa pemprograman kotlin dan menghasilkan file aplikasi bereksistensi .apk.

Pengujian

Pada tahap ini n serangkaian kasus pengujian pada perangkat lunak telah dilakukan. Berikut daftar pengujian yang telah dilakukan pada aplikasi identifikasi penyakit tomat:

1. Pengguna memasukkan gambar daun tomat yang akan diproses dengan cara diambil langsung dari galeri atau dipotret menggunakan kamera ponsel.

2. Mengidentifikasi penyakit tanaman tomat menggunakan CNN berdasarkan gambar yang dimasukkan pengguna.
3. Membandingkan hasil akurasi berdasarkan gambar daun tomat dengan model fisik yang berbeda.

HASIL DAN PEMBAHASAN

Pada bagian ini membahas tentang analisis penelitian dan temuan-temuan terbaru. Hasil percobaan/eksperimen dan analisis tersebut apakah sesuai dengan hipotesis (jika perlu). Hasil dibahas dengan mengacu pada rujukan yang digunakan.

1. Implementasi CNN Dalam *Google Colab*

Gambar 5 *Dataset* yang Telah Berhasil Dimuat

Dataset telah berhasil dimuat ke dalam *Google Colab* kemudian diubah ukurannya menjadi 64 x 64 *pixel*, selanjutnya diberi label sesuai jenis penyakitnya seperti yang terlihat pada Gambar 5.

```
[ ] #load everything into memory
    x = []
    y = []
    class_names = []
    parent_directory = "image"

    for i,folder in enumerate(os.listdir(parent_directory)):
        print(i,folder)
        class_names.append(folder)
        folder_directory = os.path.join(parent_directory,folder)
        files = os.listdir(folder_directory)
        #will inspect only 1 image per folder
        for file in files:
            file_path = os.path.join(folder_directory,file)
            image = load_img(file_path,target_size=(64,64))
            image = img_to_array(image)/255.
            x.append(image)
            y.append(i)

    x = np.array(x)
    y = to_categorical(y)

    0 Tomato___mosaic_virus
    1 Tomato___Septoria_leaf_spot
    2 Tomato___Yellow_Leaf_Curl_Virus
    3 Tomato___Bacterial_spot
    4 Tomato___Leaf_Mold
    5 Tomato___Target_Spot
    6 Tomato___Spider_mites_20Two-spotted_spider_mite
    7 Tomato___Late_blight
    8 Tomato___healthy
    9 Tomato___Early_blight

[ ] #check the data shape
    print(x.shape)
    print(y.shape)
    print(y[1])

    (18160, 64, 64, 3)
    (18160, 10)
    [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Gambar 6 *Dataset* Berdimensi 64 x 64 *Pixel*

Pada Gambar 6 terdapat *dataset* yang telah diberi label dan telah diubah dimensinya ke ukuran 64 x 64 *pixel* kemudian dibagi ketiga bagian, yakni data *training* sebanyak 15436 citra, data validasi sebanyak 2043 citra, dan data *test* sebanyak 681 citra.

```
x_train, _x, y_train, _y = train_test_split(x,y,test_size=0.15, stratify = y, random_state = 1)
x_valid,x_test, y_valid, y_test = train_test_split(_x,_y,test_size=0.25, stratify = _y, random_state = 1)

print("train data:",x_train.shape,y_train.shape)
print("validation data:",x_valid.shape,y_valid.shape)
print("test data:",x_test.shape,y_test.shape)

train data: (15436, 64, 64, 3) (15436, 10)
validation data: (2043, 64, 64, 3) (2043, 10)
test data: (681, 64, 64, 3) (681, 10)
```

Gambar 7 Pembagian *Dataset*

Setelah persiapan *dataset* selesai, maka dilanjukan dengan proses *training* model CNN.

```
Epoch 30/100
157/157 [==============================] - ETA: 0s - loss: 0.0558 - acc: 0.9838
Epoch 30: val_loss did not improve from 0.08567
157/157 [==============================] - 19s 122ms/step - loss: 0.0558 - acc: 0.9838 - val_loss: 0.7102 - val_acc: 0.8776
Epoch 31/100
157/157 [==============================] - ETA: 0s - loss: 0.0727 - acc: 0.9793
Epoch 31: val_loss did not improve from 0.08567
157/157 [==============================] - 19s 121ms/step - loss: 0.0727 - acc: 0.9793 - val_loss: 0.2337 - val_acc: 0.9383
Epoch 32/100
157/157 [==============================] - ETA: 0s - loss: 0.0592 - acc: 0.9822
Epoch 32: val_loss did not improve from 0.08567
157/157 [==============================] - 19s 121ms/step - loss: 0.0592 - acc: 0.9822 - val_loss: 0.9687 - val_acc: 0.8148
Epoch 33/100
157/157 [==============================] - ETA: 0s - loss: 0.0684 - acc: 0.9806
Epoch 33: val_loss did not improve from 0.08567
157/157 [==============================] - 20s 125ms/step - loss: 0.0684 - acc: 0.9806 - val_loss: 0.2761 - val_acc: 0.9291
Epoch 34/100
157/157 [==============================] - ETA: 0s - loss: 0.0595 - acc: 0.9824
Epoch 34: val_loss did not improve from 0.08567
157/157 [==============================] - 19s 121ms/step - loss: 0.0595 - acc: 0.9824 - val_loss: 0.1960 - val_acc: 0.9464
Epoch 35/100
157/157 [==============================] - ETA: 0s - loss: 0.0626 - acc: 0.9823
Epoch 35: val_loss did not improve from 0.08567
157/157 [==============================] - 19s 120ms/step - loss: 0.0626 - acc: 0.9823 - val_loss: 0.5086 - val_acc: 0.9015
Epoch 36/100
157/157 [==============================] - ETA: 0s - loss: 0.0641 - acc: 0.9831
Epoch 36: val_loss did not improve from 0.08567
157/157 [==============================] - 19s 120ms/step - loss: 0.0641 - acc: 0.9831 - val_loss: 2.1296 - val_acc: 0.6633
Epoch 36: early stopping
```

Gambar 8 Proses T*raining* Model CNN

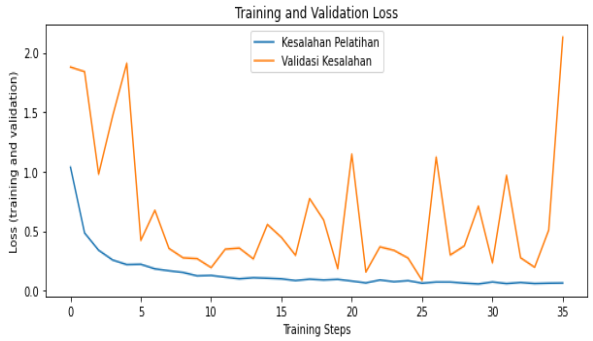
Proses *training* berjalan sebanyak *epoch* dengan hasil akhir kegagalan 0.0641, akurasi 0.9831. Hal ini berarti validasi sukses 0.6633 dan validasi gagal

2.1296 karena ada fungsi *early stoping*, maka model hanya akan menyimpan hasil *training* dengan kegagalan validasi terendah seperti yang terlihat pada gambar 8.

```
Epoch 26/100
157/157 [==============================] - ETA: 0s - loss: 0.0625 - acc: 0.9822
Epoch 26: val_loss improved from 0.15615 to 0.08567, saving model to DeteksiPenyakitTanaman.hdf5
157/157 [==============================] - 20s 127ms/step - loss: 0.0625 - acc: 0.9822 - val_loss: 0.0857 - val_acc: 0.9704
```

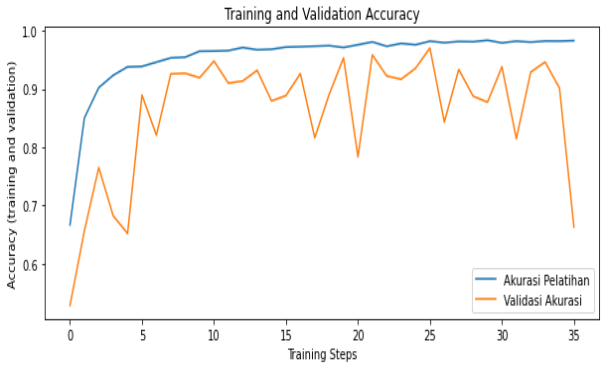
Gambar 9 Hasil T*raining* Dengan Kesalahan Terendah

Seperti yang ditunjukkan pada Gmbar 9, hasil *training* terbaik yang didapatkan berada pada *epoch 26* dengan hasil kegagalan 0.0625, akurasi 0.9822, validasi sukses 0.9704, dan validasi gagal 0.08567. Proses *training* berhenti pada *epoch* ke 36 karena setelah pelatihan model selama sepuluh kali berturut-turut dari *epoch* 26 – *epoch* 36 tidak ada peningkatan yang signifikan.



Gambar 10 Grafik Kesalahan

Gambar 10 menunjukkan sebuah grafik yang merupakan hasil *training and validation loss* dari model CNN yang telah dilatih, pada grafik garis oranye menunjukkan tingkat *error* yang terjadi selama proses validasi, sedangkan untuk garis yang berwarna biru menunjukkan tingkat kesalahan yang terjadi selama proses *training.*



Gambar 11 Grafik Akurasi

Berbanding terbalik dengan grafik sebelumnya gambar 11 menunjukkan tingkat akurasi selama proses *training* dan validasi. Garis oranye menunjukkan tingkat akurasi yang terjadi selama proses validasi, sedangkan untuk garis yang berwarna biru menunjukkan tingkat kesalahan yang terjadi selama proses *training*.

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

WARNING:absl:Function `_wrapped_model` contains input name(s) mobilenet_1.00_224_input with unsupported characters which
INFO:tensorflow:Assets written to: /tmp/tmp4ou5_rd6/assets
INFO:tensorflow:Assets written to: /tmp/tmp4ou5_rd6/assets
WARNING:absl:Buffer deduplication procedure will be skipped when flatbuffer library is not properly loaded

open("DeteksiPenyakitTanamanMoNet.tflite", "wb").write(tflite_model)

13331156
```

Gambar 12 Konversi Model Menjadi *Tensorflow Lite*

Setelah model berhasil dilatih dan menunjukkan tingkat akurasi yang bagus serta kesalalahan yang rendah, model kemudian dikonversi menjadi *TensorFlow lite* model (lihat Gambar 12). Model ini akan digunakan untuk proses implementasi CNN kedalam *platform* berbasis android.

```
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Conv2D(32, 2, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

Gambar 13 Fungsi Aktivasi

Gambar di atas menunjukkan fungsi aktivasi yang digunakan selama pelatihan model CNN untuk mengidentifikasi penyakit tanaman tomat melalui citra daun. Fungsi aktivasi yang digunakan adalah *Rectification Linear Unit* (ReLU) dan *softmax*.

Tabel 2 *Parameter* Model CNN

Layer (type)	***Output Shape***	***Param #***
mobilenet_1.00_224 (Functional)	(None, 2, 2, 1024)	3228864
conv2d (Conv2D)	(None, 1, 1, 32)	131104
dropout (Dropout)	(None, 1, 1, 32)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 32)	0
dense (Dense)	(None, 10)	330
Total *params:*		**3,360,298**
Trainable params:		**3,338,410**
Non-trainable params:		**21,888**

Jumlah *paramenter* yang dibutuhkan untuk melakukan proses *training* model CNN aplikasi identifikasi penyakit pada tanaman tomat adalah sebanyak 3.360.298 *parameter*. Hal tersebut menyebabkan butuh waktu dan tenaga yang yang banyak bagi seorang manusia untuk melakukan kalkulasi dengan parameter sebanyak itu dan hanya mesin yang mampu melakukannya dengan cepat.

2. Implementasi antarmuka Aplikasi

Bagian ini akan menjelaskan hasil dari implementasi desain antarmuka dari aplikasi keamanan citra digital yang dapat dilihat pada uraian gambar-gambar berikut ini:

Gambar 14 Implementasi Beranda

Pada tampilan ini terdapat satu *Field Text* untuk menampilkan nama perangkat lunak, juga 4 *Buttons* yang memungkinkan pengguna untuk melakukan navigasi ke halaman Identifikasi Menggunakan Kamera, Identifikasi Dari Galeri, Bantuan, dan Tentang Aplikasi.

Gambar 15 Implenentasi Identifikasi Menggunakan Kamera

Pada tampilan ini terdapat tiga *Field Text* yang masing-masing menampilkan nama perangkat lunak, menampilkan waktu yang dibutuhkan untuk memproses gambar dan menampilkan hasil klasifikasi juga tingkat akurasi. Selanjutnya terdapat satu *view* untuk menampilkan hasil tangkapan dari kamera. Terakhir satu *button* yang berguna untuk memicu fungsi deteksi dan klasifikasi penyakit tanaman tomat.

Gambar 16 Implenentasi Identifikasi dari Galeri

Terdapat tiga *Field Text* dimana masing-masing *Field Text* berfungsi untuk menampilkan nama perangkat lunak, menampilkan waktu yang dibutuhkan untuk memproses gambar, dan untuk menampilkan hasil klasifikasi beserta akurasinya. Selain itu terdapat dua *buttons* , satu *button* berfungsi untuk memilih gambar dari galeri, dan yang satunya berguna untuk memicu fungsi deteksi dan klasifikasi. Terakhir terdapat satu *View* yang berguna untuk menampilkan gambar yang telah dipilih dari galeri.

Gambar 17 Implenentasi Bantuan

Tampilan ini cukup sederhana terdapat tiga *Field Text* yang masing-masing berfungsi untuk menampilkan nama perangkat lunak, cara kerja

aplikasi beserta cara penggunaannya, serta nama dan nim pembuat aplikasi.

Gambar 18 Implenentasi Tentang Aplikasi

Tampilan ini cukup sederhana terdapat *Field Text* yang masing berfungsi untuk menampilkan nama perangkat lunak dan menampilkan deskripsi singkat aplikasi.

3. Pengujian

Dalam proses pengujian normal akan dilakukan uji coba dengan mengambil lima sampel citra dari setiap penyakit sehingga total pengujian pada proses ini adalah 5 x 10 = 50 kali pengujian. Citra yang digunakan dalam keadaan normal atau bagus.

Tabel 3 Pengujian Identifikasi Penyakit Tomat

Uji ke-	Respons Deteksi (ms)	Data uji	Hasil Prediksi (Tingkat Probabilitas)	Benar/ Salah
1	121	*T.Bacterial spot*	*T. Bacterial spot* (92.56%)	Benar
2	102	*T.Bacterial spot*	*T. Bacterial spot* (99.97%)	Benar
3	43	*T.Bacterial spot*	*T. Bacterial spot* (100.0%)	Benar
4	18	*T.Bacterial spot*	*T. Bacterial spot* (99.99%)	Benar
5	17	*T.Bacterial spot*	*T. Bacterial spot* (99.99%)	Benar
6	17	*T. Early blight*	*T. Early blight* (98.14%)	Benar
7	18	*T. Early blight*	*T. Early blight* (99.80%)	Benar
8	35	*T. Early blight*	*T. Early blight* (99.99%)	Benar
9	17	*T. Early blight*	*T. Early blight* (99.72%)	Benar
10	15	*T. Early blight*	*T. Early blight* (99.59%)	Benar
11	18	*T. healthy*	*T. healthy* (99.99%)	Benar
12	**20**	***T. healthy***	***T. Target Spot* (99.81%)**	**Salah**
13	**17**	***T. healthy***	***T. Target Spot* (99.97%)**	**Salah**
14	**32**	***T. healthy***	***T. Yellow Leaf Curl Virus* (99.94%)**	**Salah**
15	99	*T. healthy*	*T. healthy* (99.98%)	Benar
16	52	*T. Late blight*	*T. Late blight* (82.39%)	Benar
17	16	*T. Late blight*	*T. Late blight* (99.99%)	Benar
18	49	*T. Late blight*	*T. Late blight* (99.98%)	Benar
19	20	*T. Late blight*	*T. Late blight* (99.99%)	Benar
20	18	*T. Late blight*	*T. Late blight* (99.88%)	Benar
21	36	*T. Leaf Mold*	*T. Leaf Mold* (99.99%)	Benar
22	37	*T. Leaf Mold*	*T. Leaf Mold* (99.98%)	Benar
23	19	*T. Leaf Mold*	*T. Leaf Mold* (62.52%)	Benar
24	41	*T. Leaf Mold*	*T. Leaf Mold* (99.99%)	Benar
25	17	*T. Leaf Mold*	*T. Leaf Mold* (97.07%)	Benar
26	19	*T. Septoria leaf spot*	*T. Septoria leaf spot* (92.22%)	Benar
27	15	*T. Septoria leaf spot*	*T. Septoria leaf spot* (99.99%)	Benar
28	33	*T. Septoria leaf spot*	*T. Septoria leaf spot* (99.77%)	Benar
29	18	*T. Septoria leaf spot*	*T. Septoria leaf spot* (99.99%)	Benar
30	17	*T. Septoria leaf spot*	*T. Septoria leaf spot* (99.99%)	Benar
31	31	*T.Two Spotted Spider*	*T.Two Spotted Spider mites*	Benar

Uji ke-	Respons Deteksi (ms)	Data uji	Hasil Prediksi (Tingkat Probabilitas)	Benar/ Salah
		mites	(99.91%)	
32	18	*T.Two Spotted Spider mites*	*T.Two Spotted Spider mites* (99.98%)	Benar
33	17	*T.Two Spotted Spider mites*	*T.Two Spotted Spider mites* (99.99%)	Benar
34	15	*T.Two Spotted Spider mites*	*T.Two Spotted Spider mites* (99.99%)	Benar
35	18	*T.Two Spotted Spider mites*	*T.Two Spotted Spider mites* (99.25%)	Benar
36	17	*T. Target Spot*	*T. Target Spot* (99.99%)	Benar
37	25	*T. Target Spot*	*T. Target Spot* (95.76%)	Benar
38	15	*T. Target Spot*	*T. Target Spot* (99.99%)	Benar
39	18	*T. Target Spot*	*T. Target Spot* (99.99%)	Benar
40	19	*T. Target Spot*	*T. Target Spot* (99.94%)	Benar
41	30	*T. Mosaic virus*	*T. Mosaic virus* (99.99%)	Benar
42	20	*T. Mosaic virus*	*T. Mosaic virus* (99.99%)	Benar
43	17	*T. Mosaic virus*	*T. Mosaic virus* (99.99%)	Benar
44	16	*T. Mosaic virus*	*T. Mosaic virus* (99.99%)	Benar
45	15	*T. Mosaic virus*	*T. Mosaic virus* (99.99%)	Benar
46	17	*T. Yellow Leaf Curl Virus*	*T. Yellow Leaf Curl Virus* (100%)	Benar
47	16	*T. Yellow Leaf Curl Virus*	*T. Yellow Leaf Curl Virus* (100%)	Benar
48	14	*T. Yellow Leaf Curl Virus*	*T. Yellow Leaf Curl Virus* (100%)	Benar
49	12	*T. Yellow Leaf Curl Virus*	*T. Yellow Leaf Curl Virus* (100%)	Benar
50	13	*T. Yellow Leaf Curl Virus*	*T. Yellow Leaf Curl Virus* (100%)	Benar

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk daun tomat relatif bagus hanya ada tiga kesalahan identifikasi pada kelas *Tomato healthy*. Hal ini dikarenakan jumlah *dataset* yang tidak seimbang atau kurang sehingga memeengaruhi proses pelatihan dan validasi. Berdasarkan data pada Tabel 3 dapat dihitung nilai akurasi dan kesalahan uji dengan persamaan berikut.

$$\%\textbf{akurasi keseluruhan} = \frac{\boldsymbol{jumlahprediksibenar}}{\boldsymbol{jumlahdatapengujian}} \times \mathbf{100}\%$$

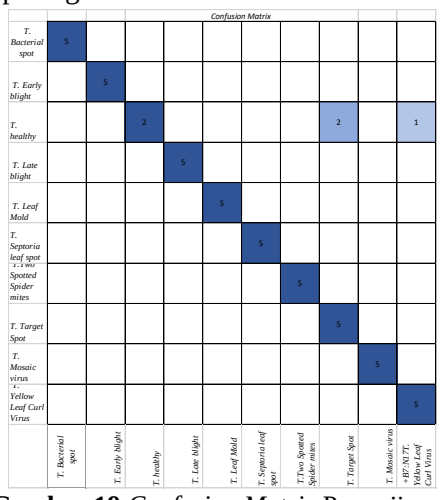
$$\%\textbf{akurasi keseluruhan} = \frac{\mathbf{47}}{\mathbf{50}} \times \mathbf{100}\% = \mathbf{94}\%$$

$$\%\textbf{kesalahan keseluruhan} = \frac{\boldsymbol{jumlahprediksisalah}}{\boldsymbol{jumlahdatapengujian}} \times \mathbf{100}\%$$

$$\%\textbf{kesalahan keseluruhan} = \frac{\mathbf{3}}{\mathbf{50}} \times \mathbf{100}\% = \mathbf{6}\%$$

Dari hasil perhitungan tersebut diperoleh persentase akurasi yang baik. Hal tersebut menandakan bahwa model CNN aplikasi identifikasi penyakit tomat mampu melakukan klasifikasi dengan baik. Rata-rata waktu yang dibutuhkan untuk melakukan klasifikasi adalah 27,18 ms.

Untuk mengetahui tingkat akurasi prediksi setiap penyakit tanaman dapat dilihat *confusion matrix* yang disajikan pada gambar berikut:



Gambar 19 *Confusion Matrix* Pengujian

KESIMPULAN

Berdasarkan hasil analisis, perancangan, implementasi, dan pengujian yang telah dilakukan pada penelitian ini, maka dapat disimpulkan sebagai berikut:

1. Aplikasi identifikasi penyakit tanaman tomat dengan citra daun berhasil dibangun dengan mengimplementasikan algoritma CNN.
2. Aplikasi dapat memproses data gambar yang berasal dari galeri ataupun data gambar yang berasal dari kamera.
3. Berdasarkan pengujian normal yang dilakukan dengan mengambil lima sampel untuk setiap kelas menghasilkan tingkat akurasi yang baik, yaitu 94% tingkat akurasi dan 6% kesalahan.
4. Berdasarkan hasil pengujian yang telah dilakukan dengan mengimplementasikan berbagai skenario uji, aplikasi yang dibangun cukup reliabel dengan akurasi uji pada identifikasi dari galeri sebesar 94%, dan 80% akurasi untuk pengujian menggunakan kamera.

UCAPAN TERIMA KASIH

Kami mengucapkan terima kasih kepada Universitas Katolik De La Salle Manado yang telah memberikan dukungan dalam penelitian ini. Kami juga mengucapkan terima kasih kepada pembimbing serta tim terkait pelaksanaan penelitian inisehingga dapat terlaksana dan selesai dengan baik.

DAFTAR PUSTAKA

Abdillah, M., Rasyad, S., & Alfarizal, N. (2022). Implementasi Sistem Pendeteksi Penggunaan Masker Berbasis Raspberry Pi 4 Menggunakan Metode Convolution Neural Network (CNN) pada Proses Screening Protokol Kesehatan COVID-19. *Jurnal Teknika*, *16*(1).

Andrian, R. F., & Maretta, G. (2017). Keanekaragaman Serangga Pollinator Pada Bunga Tanaman Tomat (Solanum Lycopersicum) Di Kecamatan Gisting Kabupaten Tanggamus. *Jurnal Tadris Pendidikan Biologi*, *8*(1).

Batubara, N. A., & Awangga, R. M. (2020). *Tutorial Object Detection Plate Number With Convolution Neural Network (CNN)*. Kreatif Industri Nusantara.

Felix, Faisal, S., Butarbutar, T. F. M., & Sirait, P. (2019). Implementasi CNN dan SVM untuk Identifikasi Penyakit Tomat via Daun. *Jurnal Sifo Mikroskil*, *20*(2), 117–134.

Khultsum, U., & Subekti, A. (2021). Penerapan Algoritma Random Forest dengan Kombinasi Ekstraksi Fitur Untuk Klasifikasi Penyakit Daun Tomat. *Jurnal Media Informatika Budidarma*, *5*(1), 186–193.

Kusumaningrum, S. I. (2019). Pemanfaatan Sektor Pertanian Sebagai Penunjang Pertumbuhan Perekonomian Indonesia. *Jurnal Transaksi*, *11*(1).

Tristianto, C. (2018). Penggunaan Metode Waterfall Untuk Pengembangan Sistem Monitoring Dan Evaluasi Pembangunan Pedesaan. *Jurnal Teknologi Informasi ESIT*, *12*(1).

Wati, C., Arsi, Karenina, T., Riyanto, Nirwanto, Y., Nurcahya, I., Melani, D., Astuti, D., Septiarini, D., Purba, S. R. F., Ramdan, E. P., & Nurul, D. (2021). *Hama dan Penyakit Tanaman*. Yayasan Kita Menulis.