JURNAL TEKNIK INFORMATIKA

Vol. 5 No. 2 Tahun. 2025
ISSN 2809-1353
DOI :10.58794/jekin.v5i2.1281

Deteksi Penyakit Sura pada Kuda Menggunakan CNN Berbasis Citra Mikroskopik

Hildegard Rambu Konda Ndima[1], Fajar Hariadi[2], Raynesta Mikaela Indri Malo[3]
[1,2,3]Teknik Informatika, Sains dan Teknologi, Universitas Kristen Wira Wacana Sumba
e-mail: [1]hildegardndima@gmail.com, [2]fajar@unkriswina.ac.id, [3]raynesta@unkriswina.ac.id

Abstract *– Sura disease, caused by the Trypanosoma evansi parasite, is an endemic disease that affects horses in East Sumba Regency. Its impact on the livestock sector is significant, as diagnosis is still conducted manually through microscopic examination by veterinary experts. This study aims to develop an automated detection system based on microscopic images of horse blood using the Convolutional Neural Network (CNN) method. The dataset consists of 200 blood images, equally divided between normal and infected conditions. The data were validated by a veterinarian and underwent preprocessing steps including conversion to grayscale, normalization, and resizing to 128×128 pixels. The CNN architecture comprises two convolutional layers, pooling layers, and fully connected layers with dropout. The training results achieved 100% accuracy, while the validation accuracy reached 97.5%. Evaluation using a confusion matrix and classification report demonstrated perfect classification performance without any misclassification. These findings indicate the potential of CNN as a fast and accurate diagnostic solution. The system can be further developed into a web- or mobile-based application.*

Keywords : *Sura disease, Trypanosoma, CNN, Microscopic image, Horse*

Abstrak **– Penyakit Sura, yang disebabkan oleh parasit Trypanosoma evansi, merupakan penyakit endemik yang menyerang kuda di Kabupaten Sumba Timur. Dampaknya sangat serius terhadap sektor peternakan karena diagnosis masih dilakukan secara manual melalui pemeriksaan mikroskopik oleh tenaga ahli. Penelitian ini bertujuan untuk mengembangkan sistem pendeteksian otomatis berbasis citra mikroskopik darah kuda menggunakan metode Convolutional Neural Network (CNN). Dataset terdiri atas 200 citra darah yang terbagi merata antara kondisi normal dan terinfeksi. Data divalidasi oleh dokter hewan, kemudian melalui proses preprocessing berupa konversi ke grayscale, normalisasi, dan resize ke ukuran 128x128 piksel. Arsitektur CNN terdiri dari dua lapisan konvolusi, pooling, dan fully connected dengan dropout. Hasil pelatihan menunjukkan akurasi pelatihan 100% dan akurasi validasi mencapai 97,5%. Evaluasi menggunakan confusion matrix dan classification report menunjukkan hasil sempurna tanpa kesalahan klasifikasi. Temuan ini menunjukkan potensi CNN sebagai solusi diagnosa cepat dan akurat. Sistem ini dapat dikembangkan lebih lanjut sebagai aplikasi berbasis web atau mobile.**

Kata Kunci**: Penyakit Sura, Trypanosoma, CNN, Citra mikroskopik, Kuda**

1. PENDAHULUAN

Sumba Timur termasuk pengekspor kuda terbanyak di wilayah lain di Indonesia Untuk tahun 2023 mencatat pengeluaran ternak dari Kabupaten Sumba Timur yang keluar daerah sudah mencapai 24.424 ekor dari kuota yang diberikan sebanyak 26.300 ekor kuota per tahun yang dikeluarkan Pemerintah Kabupaten *(*Pemkab) melalui Dinas Peternakan dalam tahun 2023 sebanyak 26.300 ekor . Kuda Sumba Timur merupakan salah satu kekayaan hayati yang memiliki nilai budaya dan ekonomi tinggi dan salah satu hewan ternak yang menopang perekonomian masyarakat Sumba Timur sehingga kuda itu merupakan penyebrangan penting sebagai perekonomian dan adat istiadat orang sumba timur[1]. kegiatan pengeksporan kuda dari wilayah ini ke luar daerah seperti Jawa, Kalimantan, dan Sulawesi semakin meningkat. Pengeksporan kuda dari Sumba Timur, termasuk peran pemerintah daerah, peternak lokal, dan pengusaha ekspor[2].

Meskipun permintaan pasar tinggi terdapat juga hambatan yang signifikan seperti adanya penyakit Sura yang menyerang ternak seperti kuda , dan Sebagian besar petani peternak di Sumba Timur masih mengandalkan hidupnya dari sektor peternakan dan hasil ternak memberikan kontribusi yang signifikan terhadap Pendapatan Asli Daerah (PAD)[3]. Penyakit Sura merupakan salah satu penyakit menular yang menyebabkan sebagian besar kuda yang terserang penyakit ini mengalami kematian. Sura merupakan penyakit yang disebabkan oleh *haemoparasit Trypanosoma evansi* (T. *evansi*) penyakit Sura di Kabupaten Sumba Timur mulai ditemukan sejak bulan Agustus 2010 yang menyerang ternak kuda dan kerbau, Dinas Peternakan Kabupaten Sumba Timur (2012) melaporkan bahwa kasus Sura di Pulau Sumba Provinsi Nusa Tenggara Timur terjadi pada tahun 2010 – 2011[4]. Kasus tersebut mengakibatkan 4268 ekor ternak (kuda 1608, kerbau 2464, sapi 196) terjangkit penyakit Sura. Kematian akibat Sura di pulau Sumba tersebut dilaporkan sebanyak 1760 ekor, terdiri dari kuda 1159 ekor, kerbau 600 ekor. Distribusi kejadiannya menyebar di Kecamatan Lewa (3,65%), Kecamatan Kota Waingapu (1,72%) dan

Kecamatan Pahunga Lodu (7,4%). Kuda memiliki nilai sosial budaya, dan juga mempunyai nilai sumber ekonomis sebagai salah satu pendapatan masyarakat peternak yang memberikan kontribusi signifikan terhadap Pendapatan Asli Daerah (Pemerintah Daerah Kabupaten Sumba Timur)[5]. PAD (Pendapatan Asli Daerah) dapat berkurang apabila Sumba Timur terkena wabah penyakit yang menyerang ternak seperti penyakit Sura yang menyebar luas di beberapa kecamatan di Kabupaten Sumba Timur.

untuk itu ketika ada kuda yang terinfeksi Sura dan kuda berhasil keluar akan menyebar ke wilayah lain di Indonesia yang sulit akan penanganannya maka diperlukan pencegahan sebelum kuda dikirim ke luar Sumba Timur[6]. Salah satu teknik yang dapat digunakan dalam membangun suatu sistem pendeteksian penyakit menggunakan kecanggihan teknologi *deep learning* dengan metode *Convolutional Neural Network* (CNN) yang memiliki keunggulan dalam pengenalan gambar visual yang lebih baik. CNN dapat mengklasifikasikan penyakit kulit dengan akurasi tinggi [7]. CNN juga efektif untuk deteksi penyakit malaria pada citra medis. CNN merupakan salah satu algoritma yang ada[8] pada *deep learning*. CNN memiliki cara kerja dengan menggunakan data set gambar atau video sebagai masukannya. CNN sering digunakan untuk penelitian khususnya untuk mengklasifikasikan suatu studi kasus dengan objek gambar atau citra gambar[9].

II . PENELITIAN YANG TERKAIT

CNN merupakan arsitektur deep learning yang unggul dalam pengolahan citra, terdiri atas lapisan konvolusi, pooling, dan klasifikasi. CNN telah digunakan dalam berbagai studi pengolahan citra medis dan veteriner. Nurhaeni et al. (2024)[10] menggunakan deep learning untuk mendeteksi parasit malaria, dengan akurasi mencapai 96%[11]. Saksenata et al (2022) menggunakan CNN untuk klasifikasi citra malaria, Identifikasi kandungan citra mikroskop sel darah manusia berbasis *image prosesing* dalam mendeteksi leokimia[12]**.** Profil Biokimia Darah Kuda yang Terlacak Berpenyakit Sura di Sumba Timur, Nusa Tenggara Timur, Indonesia [13]**,**Pada penelitian ini digunakan CNN untuk mendeteksi COVID-19 dari citra ECG spacing**,** meliputi kategori Normal, COVID-19, Infark Miokardium (MI), detak jantung abnormal, dan MI pemulihan. Untuk klasifikasi dua kelas (Normal vs COVID-19) model DenseNet201 mencapai akurasi sekitar 99,1%[14]**,** Evaluasi performa model menunjukkan hasil yang menjanjikan, dengan akurasi rata-rata sebesar 87,14%. Selain itu, model ini juga mencapai nilai precision, recall, dan F1-score masing-masing sebesar 87%, yang menunjukkan kemampuan model dalam mendeteksi dan mengklasifikasikan penyakit kulit secara konsisten[15]**.**Penelitian ini berbeda karena fokus pada deteksi penyakit Sura pada hewan ternak (kuda), menggunakan dataset lokal yang dikumpulkan langsung dari Dinas Peternakan, serta validasi dilakukan oleh tenaga medis veteriner. Belum banyak penelitian sebelumnya yang fokus pada penyakit hewan tropis menggunakan citra mikroskopik. Meskipun kedua penelitian tersebut sama-sama memanfaatkan teknologi *deep learning* dan pengolahan citra mikroskopis, terdapat perbedaan mendasar dalam hal objek penelitian dan jenis penyakit yang dianalisis, yakni penyakit pada hewan versus penyakit pada manusia Selain itu, penelitian ini secara khusus menerapkan metode *Convolutional Neural Network* (CNN) sebagai pendekatan utama, sedangkan penelitian Nurhaeni dan rekan-rekan menggunakan pendekatan *deep learning* secara umum, yang kemungkinan mencakup berbagai metode, tidak terbatas pada CNN saja.

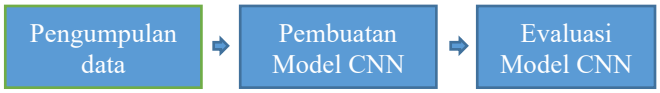
III . METODE PENELITIAN

3.1 ObjekPenelitian

Dinas Peternakan Kabupaten Sumba Timur merupakan Organisasi Perangkat Daerah (OPD) yang bertanggung jawab atas pengelolaan sektor peternakan di wilayah Kabupaten Sumba Timur, Nusa Tenggara Timur. Kantor dinas ini berlokasi di Waingapu, ibu kota Kabupaten Sumba Timur**,** Alamat Jalan Soeharto No. 42, Waingapu, Sumba Timur. Kabupaten Sumba Timur, sebuah wilayah di Nusa Tenggara Timur yang kaya akan keindahan alam dan kearifan lokal, memiliki karakteristik geografis yang unik dengan hamparan savana luas dan lahan kering yang mendominasi. Data diperoleh dari Dinas Peternakan Kabupaten Sumba Timur berupa 200 citra mikroskopik darah kuda, masing-masing 100 citra darah normal dan 100citra darah terinfeksi *Trypanosoma* evansi. Validasi data dilakukan oleh dokter hewan.

3.2 Alur Penelitian.

Pada tahap ini, penelitian dilakukan dengan menggunakan Metode CNN(*Convolutional Neural Network*) dalam mendeteksi penyakit Sura berdasarkan citra mikroskopis darah kuda .



Gambar. 1 Alur Penelitian

3.3 Pengumpulan Data

Pengumpulan data yang dilakukan melalui metode wawancara dan proses dokumentasi dilakukan kepada salah satu dokter hewan di Dinas peternakan Kabupaten Sumba Tmur . Metode dokumentasi ini akan mengambil dokumen yang berupa citra darah kuda yang normal dan yang terinfeksi Sura untuk kemudian dijadikan data set pembelajaran model CNN dan selanjutnya dilakukan analisis akurasi terhadap model yang sudah dibuat.

3.4 Pembutan Model

Pembuatan model merupakan proses membangun suatu sistem berbasis algoritma yang dapat mempelajari pola dari data dan digunakan untuk melakukan prediksi atau klasifikasi terhadap data baru. Dalam konteks pembelajaran mesin, seperti pada penggunaan *Convolutional Neural Network* (CNN), pembuatan model diawali dengan tahap pengumpulan dan persiapan data. Data yang digunakan, seperti citra mikroskopis darah, perlu melalui proses pra-pemrosesan berupa pengubahan ukuran, normalisasi, dan pelabelan sesuai kategori yang diinginkan, misalnya normal atau terinfeksi. Selanjutnya, dilakukan perancangan arsitektur model CNN dengan menyusun sejumlah lapisan (layer), seperti *convolutional layer, pooling layer*, dan *fully connected layer*, yang berfungsi untuk mengekstraksi dan mengolah fitur dari citra.

1. Preprosesing

Preprocessing adalah TahapCitra diubah dari RGB ke grayscale, kemudian dinormalisasi dan diubah ukurannya menjadi 128x128 piksel agar sesuai dengan input layer CNN. Normalisasi dilakukan dengan penskalaan piksel dari rentang [0, 255] ke rentang [0, 1].

a. Normalisasi RGB ke *Grayscale*
   Normalisasi bertujuan untuk mengubah rentang nilai piksel dari [0, 255] menjadi [0, 50]. gambar dikonversi dari RGB ke *grayscale* dengan skala abu–abu. rata –rata tertimbang dari tiga komponen warna yang berbeda, yaitu nilai merah, hijau ,dan biru (RGB),di konversi menjadi nilai abu – abu yang ekuivalen fungsi citra masukan f (a, b), di mana a dan b adalah dimensi dengan koordinat sumbu x dan sumbu y. Nilai rata-rata skala abu-abu tertimbang di representasikan oleh Y *gray* dalam persamaan (1).
b. *Resize* Gambar
   Setelah citra berhasil dikonversi dari format RGB ke *grayscale,* langkah selanjutnya melakukan *resize* atau perubahan ukuran citra. Proses ini penting untuk menyamakan dimensi semua citra masukan, sehingga model CNN dapat memprosesnya secara konsisten dan efisien. Dalam kasus ini, citra *grayscale* yang semula berukuran 5×5 piksel akan diubah menjadi ukuran 4×4 piksel.

2. *Convolutional Layer*

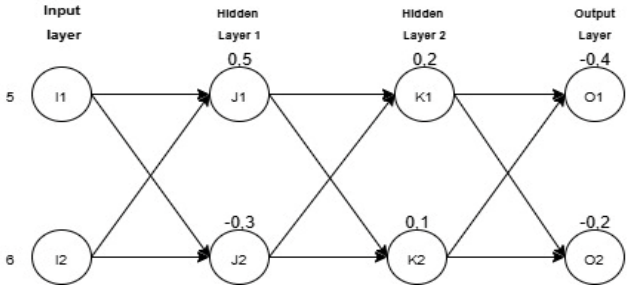
Convolution layer yang dimana dilakukan pemilihan kernel secara random. kernel yang biasa digunakan untuk deteksi tepi terutama dalam deteksi objek. proses perhitungan disetiap *Channel* dikalikan dengan kernel dengan pergeseran kernel sebanyak 1 *stride* di setiap *chennel*nya.

3. Pooling Layer

Hasil dari proses konvolusi sebelumnya akan dilanjutkan ke proses *max pooling*, yaitu dengan memilih nilai tertinggi dari setiap area kecil untuk mempertahankan fitur yang paling signifikan.

4. *Fully Connected Layer*

fully connected layer ,Pada tahap ini, *output* dari *pooling* digunakan sebagai input untuk proses klasifikasi, di mana setiap neuron saling terhubung secara penuh. Proses ini berperan penting dalam pelatihan model agar mampu mengenali serta membedakan antara citra yang mengandung parasit dan yang tidak."



Gambar. 2 Neuron

setiap neuron pada lapisan *input* terhubung dengan seluruh neuron di lapisan berikutnya melalui bobot tertentu. Setiap sinyal yang diteruskan akan dihitung berdasarkan kombinasi antara bobot dan bias yang telah ditentukan. Dalam jaringan ini, nilai input yang digunakan adalah 5 dan 6, yang merupakan hasil keluaran dari proses *pooling*. Perhitungan dilakukan dengan mengalikan setiap nilai input dengan bobot koneksi antar neuron, kemudian menjumlahkannya dengan nilai bias. Hasil dari operasi ini akan menentukan aktivasi masing-masing neuron pada lapisan selanjutnya.

3.5 Analisis akurasi

Analisis Akurasi adalah proses evaluasi untuk mengukur sejauh mana model mampu membuat prediksi yang benar terhadap data yang diberikan. Akurasi menjadi salah satu metrik evaluasi yang paling umum digunakan dalam klasifikasi dan prediksi.

1. *Confusional Matrix*

Mengevaluasi model dengan *confusion matrix*. *Confusion matrix* membandingkan hasil prediksi dengan label sebenarnya dan menunjukkan jumlah prediksi yang benar dan salah dalam empat kategori: *True* Positif (TP) **,***True* Negatif **(**TN**) ,***False* positif **(** FP)**,** dan *False* Negatif **(** FN**).** Dari sini, bisa menghitung akurasi, yaitu jumlah prediksi benar dibagi total prediksi salah untuk mengetahui kasus normal dan terinfeksi . Mengevaluasi kinerja model dengan menggunakan sejumlah metrik pengukuran, yaitu *Accuracy, Precision*, dan *Recall*.

(a) *Accuracy.*
Menentukan *accuracy* dengan membandingkan jumlah prediksi yang benar terhadap keseluruhan data yang digunakan dalam pengujian.

(b) *Precision*
Mengukur persentase prediksi yang benar di antara seluruh prediksi positif.

(c) *Recall*
Menghitung persentase data positif yang berhasil terdeteksi oleh model sebagai prediksi positif.

(d) F1 *score*
F1 Score merupakan perbandingan rata – rata presisi dan recall

IV. HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan secara langsung di Dinas Peternakan Kabupaten Sumba Timur, tepatnya di bagian kesehatan hewan (Keswan). Data yang dikumpulkan berupa citra (gambar) darah kuda, yang telah diambil dan diproses menggunakan mikroskop digital. Citra-citra tersebut dikategorikan ke dalam dua kelompok: Gambar darah kuda terinfeksi *Trypanosoma evansi* (penyakit Surra) sebanyak 100 sampel gambar darah kuda normal (tidak terinfeksi) sebanyak 100 sampel Sehingga total keseluruhan data gambar yang digunakan dalam penelitian ini adalah 200 citra. Setiap gambar telah divalidasi oleh tenaga medis hewan dari dinas tersebut untuk memastikan keakuratan label (normal atau terinfeksi). Tujuan utama dari pengumpulan data ini adalah untuk menyediakan dataset yang dapat digunakan dalam pelatihan dan pengujian model *Convolutional Neural Network* (CNN). CNN digunakan dalam penelitian ini untuk menentukan tingkat akurasi dalam mengklasifikasikan gambar darah kuda, apakah dalam kondisi terinfeksi atau normal.

4.2 Pembuatan Model CNN

Setelah melakukan proses pengumpulan data berupa citra darah kuda dari Dinas Peternakan Kabupaten Sumba Timur, langkah selanjutnya adalah melakukan pembuatan model *Convolutional Neural Network* (CNN) untuk klasifikasi kondisi darah kuda, apakah dalam keadaan normal atau terinfeksi sura. Sebelum data tersebut digunakan dalam proses pelatihan model, diperlukan tahapan awal yang sangat penting . Adapun tahapan yang digunakan adalah yaitu penyimpanan data set, preprosesing data, spliting data, pembuatan arsitektur CNN, evaluasi epoch, dan evaluasi model.

1. Peyimpanan Data.
Langkah awal adalah untuk mengunggah file dataset dalam format ZIP ke Google Colab. Pertama, skrip mengimpor beberapa pustaka penting seperti os untuk manipulasi file dan direktori, cv2 dari OpenCV untuk pengolahan citra, zipfile untuk ekstraksi file ZIP, numpy dan matplotlib untuk analisis dan visualisasi, serta modul files dari google.colab untuk mengunggah file dari komputer lokal. Setelah file ZIP diunggah, skrip secara otomatis mengekstraknya ke dalam folder bernama dataset. Selanjutnya, dengan menggunakan os.walk(), skrip akan menelusuri seluruh struktur folder hasil ekstraksi dan mencari folder yang bernama "Normal" dan "Terinfeksi", dua kategori umum yang biasanya digunakan dalam klasifikasi citra medis atau biologis.

```
import os, cv2, zipfile
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
from glob import glob

# Upload dan ekstrak dataset
uploaded = files.upload()
with zipfile.ZipFile(list(uploaded.keys())[0], 'r') as zip_ref:
    zip_ref.extractall("dataset")

# Cari folder Normal dan Terinfeksi
paths = {name: os.path.join(root, name)
         for root, dirs, _ in os.walk("dataset")
         for name in dirs if name.lower() in ["normal", "terinfeksi"]}
```

Gambar. 3 kode uplod file

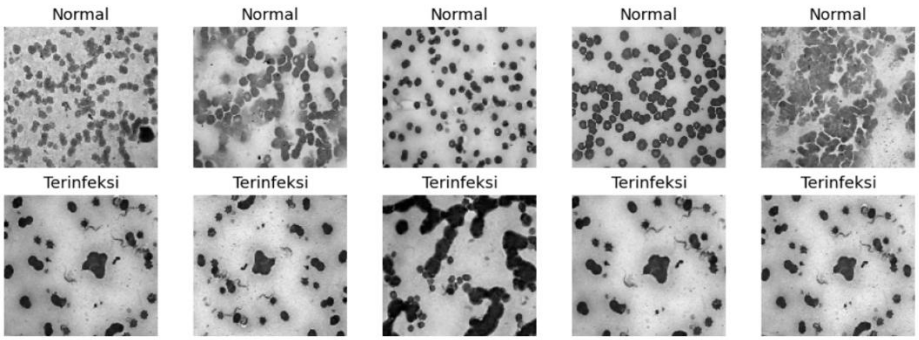
4.3 *Preprosesing* Data

Langkah selanjutnya memproses dataset citra mikroskopis yang terdiri dari dua kategori, yaitu "Normal" dan "Terinfeksi". Proses diawali dengan mengimpor sejumlah pustaka penting seperti os, cv2, zipfile, numpy, matplotlib.pyplot, google.colab.files, dan glob. Library tersebut diperlukan untuk menangani operasi file, membaca dan memproses gambar, serta memvisualisasikannya. Setelah itu, pengguna diminta untuk mengunggah file dataset dalam format ZIP. File yang diunggah kemudian diekstrak ke dalam direktori bernama "dataset". Langkah berikutnya adalah mencari folder di dalam direktori tersebut yang mengandung citra dengan nama "Normal" dan "Terinfeksi". Folder-folder ini diidentifikasi tanpa memperhatikan huruf besar atau kecil, kemudian path-nya disimpan dalam sebuah dictionary. Setelah folder ditemukan, kode akan memuat dan memproses semua gambar di dalamnya. Untuk setiap gambar, dilakukan pembacaan menggunakan OpenCV (cv2.imread).

```
import os, cv2, zipfile
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
from glob import glob

# Upload dan ekstrak dataset
uploaded = files.upload()
with zipfile.ZipFile(list(uploaded.keys())[0], 'r') as zip_ref:
    zip_ref.extractall("dataset")

# Cari folder Normal dan Terinfeksi
paths = {name: os.path.join(root, name)
         for root, dirs, _ in os.walk("dataset")
         for name in dirs if name.lower() in ["normal", "terinfeksi"]}

# Load dan preprocessing gambar
images, labels = [], []
for label_name, label_val in zip(["Normal", "Terinfeksi"], [0, 1]):
    for path in glob(os.path.join(paths[label_name], "*.*")):
        img = cv2.imread(path)
        if img is not None:
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            resized = cv2.resize(gray, (128, 128))
            images.append(resized)
            labels.append(label_val)
```

Gambar. 4 Kode *Preprocesing*

Jika gambar berhasil dibaca, gambar tersebut dikonversi ke format grayscale agar lebih sederhana untuk diproses dan kemudian diubah ukurannya menjadi 128x128 piksel. Gambar hasil preprocessing disimpan ke dalam list images, sedangkan label numerik (0 untuk Normal dan 1 untuk Terinfeksi) disimpan ke dalam list labels. Secara keseluruhan, kode ini bertujuan menyiapkan dataset gambar yang telah terstruktur dan terstandarisasi ukurannya, sehingga siap digunakan untuk pelatihan model klasifikasi citra, seperti convolutional neural network (CNN) .



Gambar 5 . Hasil *Preprocessing*

Setelah proses *resise* di atas maka lanjut pada tahap berikutnya adalah memisahkan data berdasarkan kelasnya. Data dengan label 0 (yang mewakili kelas "Normal") dipisahkan ke dalam list normal, sedangkan data dengan label 1 (kelas "Terinfeksi") dimasukkan ke dalam list terinfeksi. Tujuan dari pemisahan ini adalah agar proses split data latih dan validasi dilakukan secara stratified, yaitu setiap kelas tetap terwakili secara proporsional. Untuk membagi data, didefinisikan fungsi split(d) yang menerima list data sebagai input, kemudian mengacak urutannya menggunakan random.shuffle. Setelah diacak, fungsi ini membagi 80% data untuk pelatihan dan 20% sisanya untuk validasi, lalu mengembalikannya sebagai dua subset data. Fungsi split ini digunakan untuk membagi baik data normal maupun terinfeksi, sehingga diperoleh train_n, val_n (untuk data normal), dan train_t, val_t (untuk data terinfeksi). Akhirnya, seluruh data latih dari kedua kelas digabungkan menjadi satu list train, dan seluruh data validasi digabungkan menjadi list val. Kedua list tersebut kembali diacak agar susunan data tidak bias terhadap urutan kelas. Dengan demikian, dataset sudah siap digunakan dalam pelatihan model, dengan proporsi data yang seimbang dan acak antara kedua kelas pada data latih dan validasi.

```
import numpy as np
import random
import pandas as pd
from tensorflow.keras.utils import to_categorical

# Gabungkan gambar dan label
data = list(zip(images, labels))

# Pisahkan berdasarkan kelas
normal = [d for d in data if d[1] == 0]
terinfeksi = [d for d in data if d[1] == 1]

# Fungsi split
def split(d):
    random.shuffle(d)
    s = int(0.8 * len(d))
    return d[:s], d[s:]

# Split latih dan validasi
train_n, val_n = split(normal)
train_t, val_t = split(terinfeksi)

# Gabungkan & acak
train = train_n + train_t
val = val_n + val_t
random.shuffle(train)
random.shuffle(val)
```

Gambar 6. Kode Memisahkan Data

4.3 *Spliting* Data

Setelah melalui tahap *proses preposesing*, maka lanjut pada tahap spiliting digunakan untuk membagi dataset citra mikroskopik menjadi data latih dan data validasi menggunakan fungsi train test split dari pustaka sklearn.

```
# Siapkan data
X_train = np.array([x[0] for x in train]).reshape(-1, 128, 128, 1)
y_train = to_categorical([x[1] for x in train], num_classes=2)
X_val = np.array([x[0] for x in val]).reshape(-1, 128, 128, 1)
y_val = to_categorical([x[1] for x in val], num_classes=2)

# Ringkasan
df = pd.DataFrame({
    "Kategori": ["Normal", "Terinfeksi", "Total"],
    "Latih": [len(train_n), len(train_t), len(train)],
    "Validasi": [len(val_n), len(val_t), len(val)]
})

print("📊 Ringkasan Pembagian Data (80% latih - 20% validasi):")
print(df.to_string(index=False))
```

Gambar.7 Kode *Spiliting*

Pada Tahap selanjut nya, data gambar dan label yang telah digabung dan diacak sebelumnya dipisahkan kembali menjadi fitur (X_train, X_val) dan label (y_train, y_val). Proses ini dilakukan dengan menyusun array dari elemen pertama (gambar) dan elemen kedua (label) dari list train dan val. Gambar-gambar diubah menjadi array NumPy dan di-reshape ke bentuk (jumlah_data, 128, 128, 1) untuk menyesuaikan dengan input model CNN, di mana 1 menandakan gambar grayscale (channel tunggal). Sementara itu, label dikonversi menjadi bentuk one-hot encoding menggunakan fungsi to_categorical, sehingga bisa digunakan dalam klasifikasi dua kelas. Selanjutnya, bagian ringkasan menampilkan informasi pembagian data ke dalam bentuk tabel. Dengan menggunakan pandas.DataFrame, dibuat tabel dengan tiga kategori utama: "Normal", "Terinfeksi", dan "Total". Tabel ini berisi jumlah data untuk masing-masing kategori pada data latih dan data validasi. Kolom "Latih" menampilkan jumlah data train_n (normal), train_t (terinfeksi), dan totalnya, sedangkan kolom "Validasi" menampilkan jumlah data val_n, val_t, dan total data validasi. Terakhir, tabel ini dicetak ke layar dengan label penjelas Ringkasan Pembagian Data (80% latih - 20% validasi)".

```
📊 Ringkasan Pembagian Data (80% latih - 20% validasi):
  Kategori  Latih  Validasi
    Normal     80        20
Terinfeksi     80        20
     Total    160        40
```

Gambar 1. Hasil Spiliting

Total dataset yang digunakan terdiri dari 200 gambar, dengan rincian 100 gambar termasuk kategori normal dan 100 gambar lainnya termasuk kategori terinfeksi. Dataset ini kemudian dibagi menjadi dua bagian, yaitu data latih (training) dan data validasi (validation).

Tabel 4.1 Hasil spliting

Jenis Data	Normal	Terinfeksi	Total
Data Latih (Train)	80	80	160
Data Validasi	20	20	40
Total Gambar	100	100	200

Data latih berjumlah 160 gambar, yang terbagi secara seimbang antara 80 gambar normal dan 80 gambar terinfeksi. Sementara itu, data validasi terdiri dari 40 gambar, yang juga dibagi secara merata menjadi 20 gambar normal dan 20 gambar terinfeksi. Pembagian data ini menunjukkan bahwa proses pembagian dilakukan secara , yaitu dengan menjaga proporsi kelas yang seimbang di setiap subset data. Pendekatan ini sangat penting untuk memastikan bahwa model mendapatkan representasi yang adil dari setiap kelas selama proses pelatihan dan evaluasi.

4.4 Arsitektur Model CNN

Membangun arsitektur lengkap dari model CNN yang digunakan untuk mendeteksi penyakit sura pada darah kuda. Model ini dirancang dengan dua lapisan konvolusi untuk mengekstraksi fitur penting dari citra darah kuda, seperti bentuk, tekstur, atau pola yang mencirikan infeksi Surra. Lapisan-lapisan pooling berfungsi mengurangi dimensi data agar model tetap efisien. Setelah melalui proses flatten dan dense, model menghasilkan output akhir berupa satu nilai (0 atau 1) yang menunjukkan apakah gambar tersebut termasuk darah normal atau terinfeksi. Dengan lebih dari 7 juta parameter yang dilatih, model ini cukup kompleks untuk mendeteksi pola mikroskopik khas penyakit Surra.

```
# Bangun model CNN
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  # binary classification
])
```

Gambar 2 . Kode Model CNN

Model *Convolutional Neural Network* (CNN) di yang digunakan dalam penelitian ini dirancang untuk melakukan klasifikasi biner terhadap citra darah kuda, yaitu antara kondisi terinfeksi dan normal. Arsitektur model terdiri atas beberapa lapisan utama, diawali dengan lapisan *konvolusi* (Conv2D) sebanyak dua kali. Lapisan pertama menggunakan 32 filter dengan ukuran kernel 3×3 dan aktivasi ReLU, diikuti oleh lapisan MaxPooling2D untuk mereduksi dimensi fitur. Kemudian, dilanjutkan dengan lapisan konvolusi kedua sebanyak 64 filter dan kembali diikuti oleh max pooling. Citra hasil ekstraksi fitur kemudian diratakan menggunakan lapisan Flatten**,** lalu diproses melalui lapisan Dense sebanyak 128 unit dengan fungsi aktivasi ReLU. Untuk mencegah overfitting, ditambahkan lapisan Dropout sebesar 0.5**,** yang berfungsi mematikan neuron secara acak saat pelatihan. Akhirnya, lapisan output terdiri dari satu neuron dengan fungsi aktivasi sigmoid yang digunakan untuk klasifikasi biner. Model ini dikompilasi menggunakan optimizer Adam**,** loss function binary_crossentropy, dan metrik evaluasi berupa akurasi**.**

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	320
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
flatten (Flatten)	(None, 57600)	0
dense (Dense)	(None, 128)	7,372,928
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 7,391,873 (28.20 MB)
Trainable params: 7,391,873 (28.20 MB)
Non-trainable params: 0 (0.00 B)

Gambar 3. Output Arsitektur CNN

4.5 Evaluasi Epoch.

Selanjutnya model dikompilasi menggunakan model.compile() dengan beberapa parameter penting. Optimizer yang digunakan adalah Adam dengan learning rate sebesar 0.001, yang merupakan salah satu optimizer populer karena mampu menyesuaikan laju pembelajaran secara otomatis selama pelatihan. Fungsi loss yang digunakan adalah 'categorical_crossentropy', yang cocok digunakan untuk klasifikasi multi-kelas dengan target label dalam bentuk one-hot encoding. Metrik evaluasi yang digunakan selama pelatihan adalah 'accuracy', yang akan memantau seberapa akurat model dalam memprediksi label yang benar.

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

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

# Callback untuk menghentikan training saat val_loss tidak membaik
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

Gambar 11. Kode Kompilasi Model

Selanjutnya, diterapkan mekanisme early stopping melalui objek EarlyStopping. digunakan untuk melatih model klasifikasi gambar menggunakan data yang telah dipersiapkan sebelumnya. Fungsi model.fit() memanggil proses pelatihan dengan memasukkan data latih (X_train, y_train) dan data validasi (X_val, y_val) untuk memantau performa model selama pelatihan. Pelatihan dilakukan selama maksimum 50 *epoch* (putaran) dengan ukuran batch sebesar 32, yang artinya model akan memperbarui bobotnya setelah setiap 32 sampel. Parameter callbacks =[early_stop] menunjukkan bahwa pelatihan akan menggunakan callback early stopping, yang secara otomatis menghentikan pelatihan jika performa validasi tidak meningkat setelah beberapa epoch, sehingga membantu mencegah *overfitting*. Argumen verbose=1 digunakan untuk menampilkan progres pelatihan di layar secara detail. Hasil pelatihan akan disimpan dalam variabel history, yang berisi informasi seperti nilai akurasi dan loss dari tiap epoch, dan dapat digunakan untuk membuat grafik performa model setelah pelatihan selesai.

```
# Training model
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=50,
    batch_size=32,
    callbacks=[early_stop],
    verbose=1
)
```

Gambar 12. Kode Evaluasi Epoch

Pada bagian di bawah ini, model dilatih menggunakan data pelatihan (train_data) dan divalidasi dengan data validasi (val_data). Berdasarkan hasil pelatihan model yang ditampilkan, proses training berlangsung selama 18 epoch dari total maksimum 50 epoch yang ditentukan. Pada tiga epoch pertama, akurasi model masih sangat rendah, berkisar antara 49% hingga 51%, dengan nilai loss yang sangat tinggi. Hal ini menunjukkan bahwa pada awal pelatihan, model belum mampu mengenali pola dari data secara efektif. Namun, mulai epoch ke-4 hingga ke-6 terjadi peningkatan performa yang signifikan, di mana akurasi model naik tajam hingga mencapai 88,5% pada epoch ke-5 dan 100% pada epoch ke-7. Sejalan dengan itu, akurasi validasi juga meningkat drastis, dari hanya 50% di awal menjadi 97,5% hingga 100% setelah beberapa epoch. Setelah mencapai akurasi maksimum, performa model tetap stabil. Akurasi data pelatihan terus berada di angka 1.0000 (100%), sedangkan akurasi validasi bertahan di kisaran 97,5% hingga 100%. Nilai loss baik pada data pelatihan maupun validasi juga menunjukkan penurunan drastis hingga mendekati nol, menandakan bahwa model mampu mempelajari data dengan sangat baik dan generalisasi terhadap data validasi juga cukup tinggi. Kondisi ini menunjukkan bahwa model tidak mengalami overfitting, meskipun akurasi pelatihan sempurna, karena performa pada data validasi tetap sangat tinggi dan stabil. Dengan demikian, model yang dilatih menunjukkan performa yang sangat baik dalam membedakan citra normal dan terinfeksi, bahkan dapat dikatakan pelatihan dapat dihentikan lebih awal, sekitar epoch ke-10 hingga ke-12, untuk efisiensi waktu tanpa kehilangan akurasi yang berarti.

Gambar 13. Hasil Evaluasi Epoch

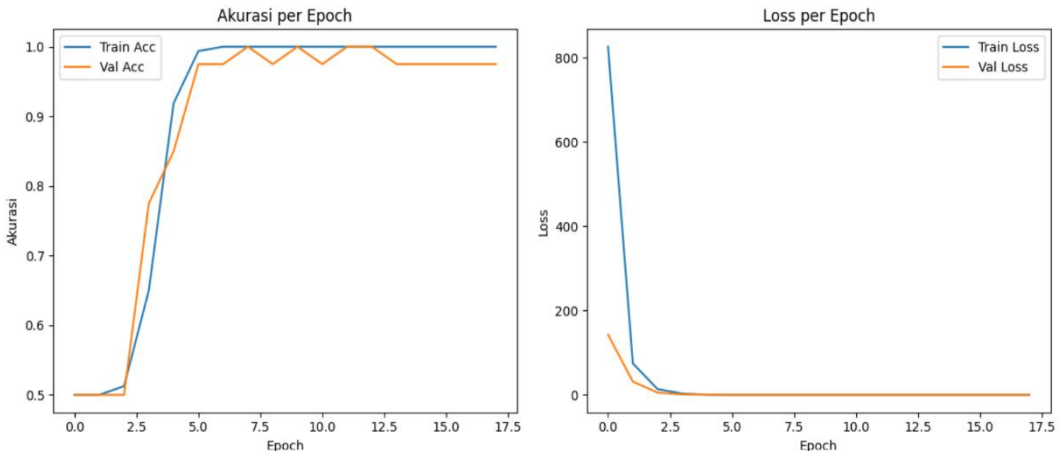
4.6 Evaluasi Model CNN

Setelah mendapat hasil evaluasi epoch di atas lanjut untuk membuat grafik akurasi dan loss. Dua subplot secara berdampingan. Subplot pertama (di kiri) memvisualisasikan akurasi per epoch, yaitu perbandingan antara accuracy (akurasi pada data pelatihan) dan val_accuracy (akurasi pada data validasi). Ini berguna untuk melihat apakah model mengalami overfitting atau underfitting. Jika kurva validasi stagnan atau menurun sementara kurva pelatihan terus meningkat, itu bisa menjadi indikasi overfitting. Subplot kedua (di kanan) menampilkan grafik loss per epoch, yaitu membandingkan loss dan val_loss. Nilai loss menunjukkan seberapa besar kesalahan model saat memprediksi. Loss yang rendah pada data pelatihan tapi tinggi pada data validasi juga mengindikasikan overfitting. Dengan menggunakan plt.tight_layout(), visualisasi akan lebih rapi tanpa saling bertumpukan. Akhirnya, plt.show() akan menampilkan grafik. Grafik ini sangat penting sebagai alat evaluasi visual untuk menentukan kualitas dan stabilitas proses pelatihan model.

```python
import matplotlib.pyplot as plt

# 📈 Plot akurasi dan loss
plt.figure(figsize=(12, 5))

# Akurasi
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Akurasi per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Akurasi')
plt.legend()

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Loss per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

Gambar 14. Kode Visualisasi Grafik

Gambar di bawah ini menunjukkan dua grafik hasil pelatihan model deep learning, yaitu grafik akurasi dan grafik loss (kerugian) terhadap jumlah epoch. Grafik sebelah kiri memperlihatkan akurasi per epoch untuk data pelatihan (Train Acc) dan data validasi (Val Acc). Terlihat bahwa akurasi model meningkat secara tajam mulai dari epoch ke-2 hingga mencapai hampir 100% pada epoch ke-6 untuk data pelatihan. Akurasi validasi juga meningkat dengan pola yang mirip, meskipun sedikit berfluktuasi di kisaran 95%–100% pada epoch selanjutnya. Hal ini menunjukkan bahwa model belajar dengan sangat baik dan mampu menggeneralisasi cukup baik terhadap data validasi. Sementara itu, grafik di sebelah kanan menunjukkan loss per epoch, yaitu nilai kesalahan model selama proses pelatihan. Pada awal pelatihan (epoch 0), nilai loss sangat tinggi, khususnya pada data pelatihan yang berada di atas 800. Namun, loss menurun drastis dan stabil di angka mendekati nol setelah beberapa epoch pertama, baik untuk data pelatihan maupun validasi. Penurunan loss yang drastis dan stabilitas nilai akurasi setelah beberapa epoch menunjukkan bahwa model berhasil melakukan pembelajaran dengan sangat cepat dan efisien ,model secara umum dapat dikatakan bekerja dengan baik dan stabil. Akurasi pada grafik di sebelah kiri menunjukkan seberapa baik model dalam mengklasifikasikan data secara benar pada setiap epoch. Terlihat bahwa akurasi pelatihan (Train Acc) meningkat dengan cepat, dari sekitar 50% di awal pelatihan hingga mencapai 100% pada epoch ke-6 dan tetap stabil sampai akhir. Sementara itu, akurasi validasi (Val Acc) juga meningkat tajam dan mencapai kisaran 97% hingga 100%. Pola ini menunjukkan bahwa model memiliki performa sangat baik dalam mempelajari data pelatihan dan cukup konsisten saat diuji pada data yang belum pernah dilihat sebelumnya (data validasi).



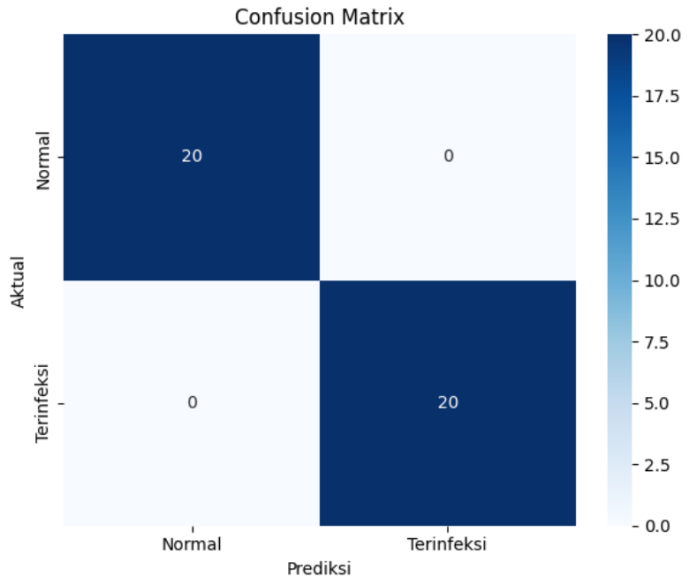
Gambar 15. Hasil Visualisasi Grafik

Kemudian, confusion matrix dihitung dengan fungsi confusion_matrix, yang membandingkan label aktual dan hasil prediksi. Matriks ini menunjukkan jumlah prediksi yang benar dan salah untuk masing-masing kelas. Terakhir, confusion matrix divisualisasikan menggunakan seaborn.heatmap, sebuah peta warna yang memperjelas distribusi nilai dengan anotasi angka dan pewarnaan berdasarkan intensitas. Label sumbu X dan Y diberi keterangan “Prediksi” dan “Aktual” agar memudahkan pembacaan, serta ditambahkan judul "Confusion Matrix".

```
# Confusion matrix
cm = confusion_matrix(y_true, y_pred_classes)

# Visualisasi confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["Normal", "Terinfeksi"],
            yticklabels=["Normal", "Terinfeksi"])
plt.xlabel("Prediksi")
plt.ylabel("Aktual")
plt.title("Confusion Matrix")
plt.tight_layout()
plt.show()
```

Gambar 16. Kode *Confusion Matrix*

Gambar di bawah ini menunjukkan confusion matrix dari hasil evaluasi model klasifikasi terhadap dua kelas, yaitu "Normal" dan "Terinfeksi". Pada confusion matrix ini, semua data uji berhasil diklasifikasikan dengan benar oleh model. Sebanyak 20 gambar yang benar-benar berlabel "Normal" diprediksi dengan benar sebagai "Normal", dan 20 gambar yang berlabel "Terinfeksi" juga diprediksi dengan benar sebagai "Terinfeksi". Tidak ada satu pun kesalahan prediksi (false positive maupun false negative), yang ditunjukkan oleh nol (0) pada sel-sel di luar diagonal utama. Hal ini menandakan bahwa model memiliki akurasi 100% dalam klasifikasi data uji.



Gambar 17. Hasil *Confusion Matrix*

Setelah mendapat hasil hasil evaluasi model proses pelatihan selesai, model digunakan untuk memprediksi kelas dari data validasi (X_val) menggunakan model.predict(), yang akan menghasilkan probabilitas untuk masing-masing kelas. Probabilitas ini kemudian dikonversi ke bentuk kelas prediksi (y_pred_classes) menggunakan np.argmax(), yakni memilih indeks kelas dengan probabilitas tertinggi. Nilai label asli (y_true) juga diubah dari bentuk one-hot encoding ke bentuk label numerik dengan cara yang sama. Langkah selanjutnya adalah mencetak classification report menggunakan fungsi classification_report dari sklearn.metrics. Laporan ini mencakup metrik penting seperti precision, recall, f1-score, dan support untuk masing-masing kelas: "Normal" dan "Terinfeksi". Ini memberikan gambaran lengkap mengenai performa model terhadap masing-masing kelas, tidak hanya dari sisi akurasi, tetapi juga seberapa baik model mengenali masing-masing kategori.

```
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report

# Prediksi data validasi
y_pred = model.predict(X_val)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_val, axis=1)

# Tampilkan classification report
print("\n Classification Report:")
print(classification_report(y_true, y_pred_classes, target_names=["Normal", "Terinfeksi"]))

# Confusion matrix
cm = confusion_matrix(y_true, y_pred_classes)
```

Gambar 18. Kode Prediksi Model

Dari Gambar di bawah ini menampilkan hasil evaluasi model klasifikasi dalam bentuk classification report, yang mencakup metrik precision, recall, f1-score, dan jumlah data (support) untuk masing-masing kelas, yaitu "Normal" dan "Terinfeksi". Berdasarkan hasil tersebut, model menunjukkan performa yang sangat sempurna dengan nilai precision, recall, dan f1-score sebesar 1.00 untuk kedua kelas. Hal ini berarti bahwa model berhasil mengklasifikasikan seluruh data uji dengan benar tanpa kesalahan sama sekali. Total data uji berjumlah 40, masing-masing terdiri dari 20 gambar kelas Normal dan 20 gambar kelas Terinfeksi, dan semuanya diprediksi dengan akurasi 100%. Nilai macro average dan weighted average yang juga mencapai 1.00 menunjukkan bahwa model mampu mempertahankan performa konsisten untuk semua kelas, baik dari segi proporsi maupun keseimbangan.

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step

Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00        20
   Terinfeksi       1.00      1.00      1.00        20

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

Gambar. 19 Hasil Prediksi Model

V . KESIMPULAN

Penelitian ini dilakukan dengan tujuan utama untuk mengembangkan sebuah sistem pendeteksian dini penyakit Sura pada kuda menggunakan pendekatan teknologi berbasis *deep learning*, khususnya algoritma *Convolutional Neural Network* (CNN). Penyakit Sura, yang disebabkan oleh parasit *Trypanosoma evansi*, merupakan salah satu ancaman serius terhadap keberlangsungan peternakan kuda di Kabupaten Sumba Timur, sebuah wilayah yang dikenal sebagai salah satu pusat populasi kuda di Indonesia. Penyakit ini tidak hanya mengancam kesehatan ternak, tetapi juga memiliki dampak signifikan terhadap aspek ekonomi, budaya, dan sosial masyarakat setempat yang sangat bergantung pada sektor peternakan. Dalam penelitian ini, data citra mikroskopis darah kuda yang digunakan diperoleh secara langsung dari Dinas Peternakan Kabupaten Sumba Timur dan telah melalui proses validasi oleh dokter hewan. Dataset terdiri dari 200 gambar, masing-masing mewakili kondisi darah normal dan terinfeksi. Seluruh citra tersebut diproses melalui beberapa tahapan penting, yaitu konversi dari RGB ke grayscale, normalisasi piksel, serta resize menjadi ukuran seragam 128x128 piksel, agar sesuai dengan input layer model CNN. Setelah dilakukan pembagian dataset secara stratified menjadi 80% data latih dan 20% data validasi, model CNN dilatih menggunakan dua convolutional layer, pooling layer, dan fully connected layer dengan dropout untuk mencegah overfitting. Hasil pelatihan menunjukkan peningkatan performa model yang sangat cepat, di mana akurasi pelatihan mencapai 100% dalam 6 epoch, dan akurasi validasi stabil di angka 97,5% hingga 100% tanpa tanda overfitting berat. Visualisasi grafik akurasi dan loss per epoch memperkuat hasil tersebut, dengan grafik loss yang menurun drastis dan konsisten stabil mendekati nol baik pada data pelatihan maupun validasi. Evaluasi model menggunakan confusion matrix menunjukkan bahwa semua prediksi pada data uji dilakukan dengan benar (100% akurasi), tanpa terjadi false positive maupun false negative. Selain itu, classification report menunjukkan nilai precision, recall, dan f1-score yang sempurna (1.00) untuk kedua kelas, yaitu darah normal dan darah terinfeksi. Berdasarkan keseluruhan hasil evaluasi, dapat disimpulkan bahwa metode CNN sangat efektif dalam melakukan klasifikasi citra mikroskopis darah kuda untuk deteksi penyakit Sura. Model yang dikembangkan mampu mengenali fitur visual penting dari darah yang terinfeksi parasit *Trypanosoma evansi* secara akurat dan efisien. Sistem yang dibangun juga dapat dijadikan prototipe awal untuk pengembangan aplikasi diagnosis berbasis komputer dalam sektor kesehatan hewan.

DAFTAR PUSTAKA

[1] U. Y. A. Praing, I. A. P. Apsari, dan N. S. Dharmawan, “Prevalensi dan Faktor Risiko Trypanosomiasis pada Kuda di Kabupaten Sumba Timur,” *Bul. Vet. Udayana*, vol. 2021, no. 158, hal. 737, 202.

[2] R. E. D. Dongga, “Adopsi Teknologi Pengendalian Penyakit Surra oleh Peternak Kuda di Kabupaten Sumba Timur, Nusa Tenggara Timur,” *Bul. Vet. Udayana*, vol. 6, no. 1, 2013.

[3] B. Radita dan D. Nur, “Trypanosoma evansi,” 2017.

[4] S. E. Wibowo *et al.*, “Trypanosoma evansi Infection in Sumba Horses in East Sumba Regency : A Study at BBVet Denpasar Infeksi Trypanosoma evansi pada Kuda Sumba di Kabupaten Sumba Timur : Sebuah Studi di BBVet Denpasar,” vol. 05, no. 1, hal. 60–65, 2024.

[5] W. Yolanda, R. Elisia, dan M. K. Susalam, “Review Literatur : Identifikasi Penyakit Surra (Protozoa Darah) Pada Ternak Ruminansia (Literature Review : Identification Of Surra (Blood Protozoan) Disease in Ruminants),” no. 2, 2024.

[6] N. S. D. Umbu Yabu Anggung Praing1, Ida Ayu Pasti Apsari2, “Prevalensi dan Faktor Risiko Trypanosomiasis pada Kuda di Kabupaten Sumba Timur,” vol. 2021, no. 158, hal. 737–746, 2023.

[7] W. M. Baihaqi *et al.*, “Analisis Gambar Sel Darah Berbasis Convolution Neural Network untuk Mendiagnosis Penyakit Demam Berdarah,” vol. 7, no. 1, hal. 148–159, 2021.

[8] F. Abdurahman, K. A. Fante, dan M. Aliy, “Malaria parasite detection in thick blood smear microscopic images using modified YOLOV3 and YOLOV4 models,” *BMC Bioinformatics*, hal. 1–17, 2021.

[9] K. Amalia, R. Magdalena, dan S. Saidah, “Klasifikasi Penyakit Tumor Otak Pada Citra Mri Menggunakan Metode CNN,” *e-Proceeding Eng.*, vol. 8,

no. 6, hal. 3247–3254, 2022.
[10] Nurhaeni1, S. E. P. A. Hidayat3, dan F. N. Anisa4, "Pemodelan Sistem Deteksi Parasit Malaria pada Citra Mikroskopis Sel Darah Menggunakan Metode Deep Learning," *Univ. Sari Mulia*, vol. 14, no. 2, hal. 409–416, 2024.
[11] S. E. Prastya *et al.*, "Pemodelan Sistem Deteksi Parasit Malaria pada Citra Mikroskopis Sel Darah Menggunakan Metode Deep Learning," vol. 14, no. 2, hal. 409–416, 2024.
[12] A. F. Saksenata, A. E. Minarno, dan Y. Azhar, "Klasifikasi Citra Sel Darah Untuk Penyakit Malaria Dengan Metode CNN," vol. 4, no. 2, hal. 185–194, 2022.
[13] N. K. Hamzidah, "Identifikasi Kandungan Citra Mikroskopik Sel Darah Manusia berbasis Image Processing dalam Mendeteksi Potensi Leukimia," *Media Elektr.*, vol. 20, no. 2, hal. 1–7, 2023.
[14] T. Rahman *et al.*, "COV-ECGNET: COVID-19 detection using ECG trace images with deep convolutional neural network," *Heal. Inf. Sci. Syst.*, vol. 10, no. 1, 2022.
[15] Imam Fathurrahman, Mahpuz, Muhammad Djamaluddin, Lalu Kerta Wijaya, dan Ida Wahidah, "Pengembangan Model Convolutional Neural Network (CNN) untuk Klasifikasi Penyakit Kulit Berbasis Citra Digital," *Infotek J. Inform. dan Teknol.*, vol. 8, no. 1, hal. 298–308, 2025.