Jurnal Bumigora Information Technology (BITe)
Vol. 6, No. 2, Desember 2024, Hal. 149∼162
E-ISSN: 2685-4066, DOI: 10.30812/bite.v6i2.4401

Deteksi Penyakit Lumpy Hewan Ternak Menggunakan Metode Arsitektur MobileNetV2

Detection of Lumpy Disease in Livestock Using the MobileNetV2 Architecture Method

Dion Pratama Putra*, **Giri Wahyu Wiriasto**, **Paniran**
Universitas Mataram, Mataram, Indonesia

Informasi Artikel:
Diterima: 22 Agustus 2024, Direvisi: 15 Oktober 2024, Disetujui: 1 November 2024

Abstrak-
Latar Belakang: Penyakit *Lumpy Skin Disease* (LSD) menyebabkan lesi kulit, penurunan produksi susu, dan kematian pada hewan ternak seperti sapi.
Tujuan: Tujuan penelitian ini adalah mendeteksi penyakit LSD secara cepat dan akurat menggunakan metode *Convolutional Neural Network* (CNN) MobileNetV2 berbasis aplikasi android.
Metode: Penelitian ini menggunakan metode kuantitatif dengan pendekatan *reuse-oriented development* serta algoritma MobileNetV2 yang dilatih dengan data augmentasi dan klasifikasi citra penyakit LSD.
Hasil: Hasil penelitian ini adalah model klasifikasi MobileNetV2 mampu mendeteksi LSD dengan akurasi 95,91%. Aplikasi yang dikembangkan memudahkan peternak dalam mendeteksi penyakit secara dini sehingga dapat mempercepat tindakan pencegahan.
Kesimpulan: Implikasi dari penelitian ini menunjukkan bahwa model MobileNetV2 dapat meningkatkan efektivitas deteksi penyakit pada hewan ternak dan dapat diterapkan dalam aplikasi kesehatan hewan di lapangan.

Kata Kunci: Deteksi Penyakit Lumpy, Hewan Ternak, MobileNetV2.

Abstract-
Background: *Lumpy Skin Disease (LSD) causes skin lesions, decreased milk production, and death in livestock such as cows.*
Objective: *This study aims to detect LSD disease quickly and accurately using the Convolutional Neural Network (CNN) MobileNetV2 method based on an Android application.*
Methods: *This study uses a quantitative method with a reuse-oriented development approach and the MobileNetV2 algorithm trained with augmentation data and LSD disease image classification.*
Result: *This study's implications indicate that the MobileNetV2 model can improve the effectiveness of disease detection in livestock and can be applied in animal health applications in the field.*

Keywords*: Lumpy Disease Detection, Livestock, MobileNetV2.*

Penulis Korespondensi:

Dion Pratama Putra,
Program Studi Teknik Elektro, Universitas Mataram, Mataram, Indonesia,
Email: dionpratamaputra2012@gmail.com

1. PENDAHULUAN

Penyakit *Lumpy Skin Disease* (LSD) adalah penyakit yang menyerang sapi dan kerbau domestik yang disebabkan oleh *Capripoxvirus* dan dapat mengakibatkan kerugian ekonomi yang signifikan [1]. Gejala klinis yang ditimbulkan akibat infeksi virus *Lumpy* antara lain demam mencapai 41,5°C, tidak nafsu makan, penurunan produksi susu, ingusan, depresi, pembengkakan, terdapat nodul pada kulit yang menonjol di daerah kepala,

How to Cite: D. Putra, G. Wahyu Wiriasto, and P. Paniran, "Detection of Lumpy Disease in Livestock Using the MobileNetV2 Architecture Method", *Jurnal Bumigora Information Technology (BITe)*, vol. 6, no. 2, pp. 149-162, Dec. 2024. doi: 10.30812/bite.v6i2.4401

leher, punggung, ekor dan bagian daerah *genital*. Di Indonesia, penyakit ini masih belum dinyatakan bebas dari infeksi *Lumpy*. Mengingat penyakit ini merupakan penyakit eksotik bagi Indonesia, maka pemahaman tentang penyakit *Lumpy* sangat diperlukan bagi pemilik ternak agar dapat dilakukan antisipasi lebih awal jika ada kasus yang dicurigai sebagai *Lumpy* [2].

Oleh karena itu, beberapa penelitian terkait pendeteksian LSD menggunakan pendekatan *Convolutional Neural Network* (CNN) telah dilakukan seperti penelitian [3] membangun model deteksi citra *Lumpy* menggunakan arsitektur *Visual Geometric Group* (VGG16), VGG19, dan *Residual Network* (ResNet50). Penelitian serupa telah dilakukan dengan membangun model deteksi citra Lumpy menggunakan arsitektur *Densely Connected Convolutional Networks* (DenseNet121) [4]. Masalah yang ingin diselesaikan adalah bagaimana mengembangkan aplikasi yang dapat mendeteksi penyakit LSD secara efektif dan efisien dalam kondisi nyata. Selain itu, penyesuaian tingkat akurasi klasifikasi penyakit LSD pada sapi ternak perlu dilakukan agar aplikasi yang dikembangkan tidak hanya akurat, tetapi dapat diandalkan dalam berbagai kondisi di lapangan.

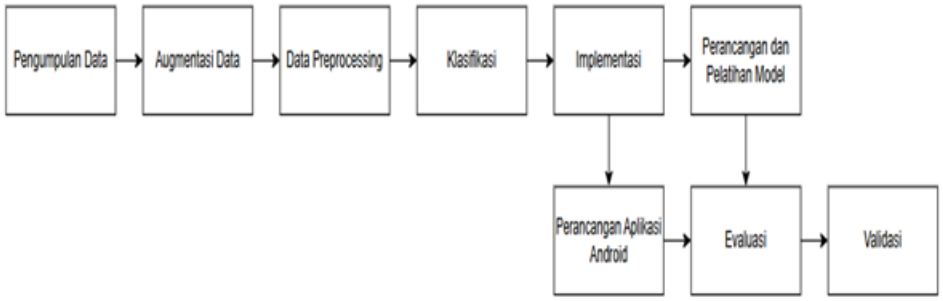
Terdapat *gap* yang belum diselesaikan oleh penelitian sebelumnya yaitu penggunaan model yang lebih ringan dan cocok diterapkan pada perangkat *mobile* untuk deteksi *Lumpy* secara cepat dan akurat di lapangan. Penelitian ini berbeda dari penelitian sebelumnya dalam hal penggunaan arsitektur MobileNetV2 yang dirancang khusus untuk perangkat *mobile* dengan kinerja tinggi dan ringan. Dalam penelitian ini, model deteksi *Lumpy* dikembangkan menggunakan arsitektur MobileNetV2 yang telah dilatih oleh Google dengan dataset *ImageNet* dan dirancang untuk diterapkan pada *platform* Android yang juga dikelola oleh Google. Tujuan penelitian ini adalah mendeteksi penyakit LSD secara cepat dan akurat menggunakan metode arsitektur CNN MobileNetV2 berbasis aplikasi android. Kontribusi penelitian ini terletak pada pengembangan model deteksi penyakit ternak yang lebih praktis dan dapat diterapkan langsung di lapangan, sehingga dapat membantu peternak dalam mengambil tindakan pencegahan lebih dini dalam pencegahan penyebaran penyakit dan menjaga kesehatan hewan ternak.

Artikel ini dikelompokkan menjadi beberapa bagian sesuai dengan penelitian yang telah dilakukan, di antaranya: metode penelitian yang menjelaskan tahapan dan proses yang dilakukan dalam pembangunan fitur utama serta pengembangan aplikasi pada penelitian ini; hasil dan pembahasan yang menguraikan hasil akhir dari pembangunan yang telah dilakukan; serta kesimpulan yang memaparkan temuan utama dari penelitian ini dan memberikan saran untuk penelitian selanjutnya.

2. METODE PENELITIAN

Pada penelitian ini, metode kuantitatif digunakan dengan pendekatan *reuse-oriented development* atau *reuse-oriented software engineering*, yang merupakan salah satu pendekatan sistematis dalam rekayasa perangkat lunak. Pendekatan ini didasarkan pada gagasan bahwa keberadaan komponen yang dapat digunakan kembali (*reusable*) memungkinkan proses pengembangan perangkat lunak menjadi lebih efisien dan konsisten [5]. Dengan demikian, pengembang dapat mengintegrasikan komponen-komponen tersebut ke dalam rancangan sistem baru, dibandingkan mengembangkannya dari awal. Penelitian ini menggunakan algoritma MobileNetV2 untuk mendeteksi dan mengklasifikasikan citra LSD, serta mengintegrasikan komponen perangkat lunak dalam pengembangan aplikasi, seperti yang ditunjukkan pada Gambar 1.

Pada Gambar 1 ditunjukkan tahapan penelitian yang menggunakan metode *reuse-oriented development* atau *reuse-oriented software engineering*. Studi kasus penelitian ini berfokus pada deteksi penyakit LSD pada hewan ternak, dengan sumber data yang diperoleh dari penelitian [6] yang mencakup 324 gambar kulit *Lumpy* dan 700 gambar kulit sehat, sehingga total data yang terkumpul berjumlah 1.024 citra. Seluruh citra dalam *dataset* berukuran 256×256 piksel dan disimpan dalam format PNG seperti yang ditunjukkan pada Gambar 2. Sebelum menggunakan dataset tersebut, dilakukan proses sortir atau data *cleaning* untuk menghapus data duplikat sehingga pada kelas kulit Lumpy diperoleh 299 data bersih. Untuk membangun model klasifikasi menggunakan *dataset* ini, setiap kelas dataset dibagi menjadi dua bagian yaitu data latih dan data validasi. Model ini dirancang untuk mendeteksi dan mengklasifikasikan antara kondisi sehat dan *Lumpy*. Data latih



Gambar 1. Tahapan Penelitian Mengunakan *Reuse Oriented Architecture*

digunakan untuk membantu model mempelajari pola-pola dalam data agar akurat dalam melakukan klasifikasi, sehingga data latih sebaiknya lebih banyak dibandingkan data validasi. Data validasi berfungsi untuk menilai kemampuan model dalam memprediksi data yang belum pernah diproses sebelumnya. Pembagian dataset dilakukan dengan menetapkan parameter ukuran *split,* yang secara otomatis membagi dataset menjadi proporsi tertentu tanpa mengabaikan keseimbangan antar kelas. Penelitian ini, dataset terdiri atas 630 citra dengan kelas kulit *Lumpy* menggunakan 291 citra untuk data latih, 70 citra kelas kulit sehat, dan 30 citra kelas kulit Lumpy untuk data validasi. Proses ini merupakan bagian dari data *preprocessing* yang dilakukan untuk menyiapkan dataset agar sesuai dengan *input* yang diharapkan oleh model.

Gambar 2. Sampel Data Citra Hewan Terjangkit Lumpy [6]

Dataset citra hewan yang terinfeksi Lumpy yang dikumpulkan melalui sumber terbuka, tahap awal penelitian ini meliputi pengumpulan data, termasuk dataset dan library yang diperlukan. Setelah semua komponen terkumpul, dilakukan augmentasi data pada *dataset* yang diperoleh, diikuti oleh proses data *preprocessing* untuk membagi data latih dan data validasi. Selanjutnya, dilakukan klasifikasi menggunakan model arsitektur MobileNetV2 sebagai komponen utama untuk mendeteksi dan mengklasifikasikan infeksi. Penelitian ini diimplementasikan di *Google Colab*, sementara integrasi ke dalam aplikasi Android dirancang menggunakan Android Studio untuk *user interface* (UI). Semua komponen diintegrasikan menjadi satu sistem untuk mengevaluasi keseluruhan klasifikasi dan deteksi, baik dari segi aplikasi Android maupun model klasifikasi, guna memvalidasi perancangan penelitian ini. Pengujian klasifikasi dilakukan dengan memberikan masukan berupa gambar hewan ternak untuk mendeteksi apakah hewan tersebut terinfeksi LSD atau tidak. Dengan pendekatan *reuse-oriented development*, pengembang dapat menghemat waktu dan usaha serta meningkatkan kualitas dan konsistensi produk akhir pada tahapan terakhir yaitu pengujian dan analisis untuk menjamin sistem sudah memenuhi harapan sesuai spesifikasi persyaratan [5].

3. HASIL DAN PEMBAHASAN

Temuan penelitian ini adalah model arsitektur MobileNetV2 berhasil mencapai akurasi sebesar 95,91% pada data validasi setelah melalui 100 *epoch* dengan *batch size* sebesar 45. Penggunaan augmentasi data seperti *rotasi, shear, zoom,* dan *horizontal flip* terbukti efektif dalam meningkatkan performa model dan mampu mempelajari pola data tanpa *overfitting*. Evaluasi dengan *confusion matrix* menunjukkan bahwa nilai *precision* untuk kelas sehat mencapai 1, sementara kelas *Lumpy* memiliki *precision* sebesar 0.97. Hasil penelitian ini sejalan dengan penelitian sebelumnya yang menggunakan arsitektur CNN untuk klasifikasi penyakit hewan ternak seperti penggunaan arsitektur VGG16, ResNet50 [3], dan DenseNet121 [4], dilaporkan memiliki akurasi yang hampir setara. Namun, model yang digunakan dalam penelitian ini menunjukkan performa yang lebih optimal pada perangkat *mobile*.

Dari sisi keunggulan, MobileNetV2 memberikan manfaat yang lebih besar pada perangkat *mobile* dengan sumber daya terbatas. Perbedaan penelitian ini dengan penelitian sebelumnya adalah integrasi teknologi MobileNetV2 ke dalam aplikasi Android yang memungkinkan deteksi LSD secara langsung di lapangan. Pengembangan ini memberikan solusi yang lebih praktis dan efisien dibandingkan penelitian terdahulu yang masih mengandalkan perangkat *desktop* atau *cloud* dengan sumber daya yang lebih besar.

3.1. Augmentasi Data

Augmentasi merupakan suatu teknik yang berfungsi memperbesar gambar, memutar, memberikan Cahaya, dan teknik augmentasi dapat menaikan nilai akurasi model [7]. Augmentasi data pada penelitian ini bertujuan untuk meningkatkan akurasi saat melakukan deteksi dan klasifikasi pada model dengan parameter tertentu saat diberi *input*. Penggunaan teknik augmentasi citra pada penelitian ini meliputi parameter seperti: (1) *Rotation* untuk melakukan putaran dengan sudut 40 derajat pada citra secara acak; (2) *Shear range* untuk menentukan sudut kemiringan derajat dengan nilai 0,2; (3) *Zoom range* untuk menentukan jarak ukuran yang digunakan untuk memperbesar citra dengan nilai 0,2; (4) *Horizintal flip* yang digunakan untuk membalik citra secara horizontal adalah *true* atau benar setiap citra memiliki posisi horizontal dan ukuran citra adalah 150x150. Parameter-parameter tersebut diterapkan selama pelatihan model agar model dapat mengubah citra *input* klasifikasi dengan berbagai variasi menjadi lebih sesuai dengan parameter yang telah ditentukan. Citra sebelum dan sesudah melalui proses augmentasi data ditunjukkan pada Gambar 3.

(a) (b)

Gambar 3. Augmentasi Data; (a) Citra Sebelum Augmentasi; (b) Citra Setelah Augmentasi

3.2. Data *Preprocessing*

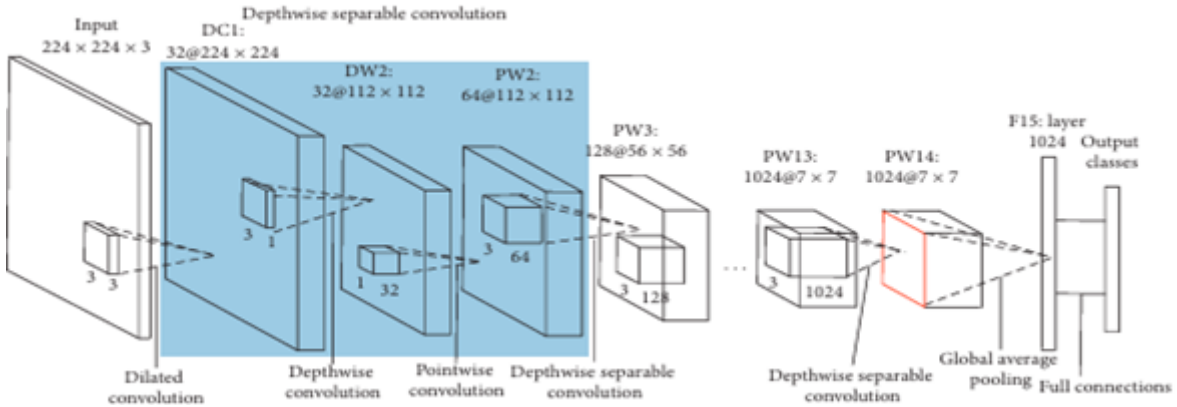
Data *preprocessing* merupakan proses awal yang akan mentransformasikan data masukan menjadi data dengan format yang sesuai dan siap untuk diproses [8]. Tahapan ini dilakukan pada data mentah untuk menghilangkan data yang bermasalah atau inkonsisten. Data bermasalah seperti data yang mengandung *noise*

atau *error* [9] yang menyebabkan data yang diperoleh terdiri dari 700 gambar kelas kulit sehat dan 324 gambar kelas kulit Lumpy. Dataset kemudian dibagi menjadi dua bagian untuk dimasukkan ke dalam model klasifikasi, dengan rincian pembagian sebagai berikut: kelas kulit sehat menggunakan 630 gambar untuk data latih yaitu data yang digunakan untuk melatih model klasifikasi dan 70 gambar untuk data validasi, yang digunakan untuk memvalidasi model klasifikasi agar terhindar dari *overfitting*. Untuk kelas kulit *Lumpy* sebesar 291 gambar digunakan sebagai data latih dan 33 gambar sebagai data validasi.

3.3. Klasifikasi

Proses selanjutnya adalah klasifikasi citra menggunakan arsitektur MobileNetV2 dengan teknik *transfer learning* yang dapat mengatasi keterbatasan data dengan menggunakan *Pre-trainned* untuk melatih model pada *dataset* yang sedikit dan memberikan akurasi dan performa yang tinggi [10]. Pada MobileNetV2 digunakan untuk mengekstrak fitur citra kulit *Lumpy* dalam kumpulan *dataset* penyakit kulit *Lumpy* dan dimodifikasi dengan teknik *transfer learning* pada *top layer* untuk proses klasifikasi menggunakan *Global Average Polling* dan lapisan *dense* dengan aktivasi *sigmoid* untuk membedakan dua kelas untuk logika *Boolean* dan agar tidak terjadi *overfitting* pada model. RMSprop *optimizer* digunakan karena cocok untuk model sederhana yang hanya memerlukan satu jenis klasifikasi citra. Parameter yang terdapat pada *fully connecter layer (dense)* memperlambat kecepatan pelatihan jaringan untuk *training* dan membuat mudah *overfiting* [11]. Oleh sebab itu, *Global Average Polling* digunakan untuk menghasilkan satu peta fitur pada setiap kategori klasifikasi yang langsung memasukkan ke lapisan *sigmoid* [12]. Aktiviasi *Sigmoid* digunakan untuk melakukan klasifikasi terhadap dua kelas penyakit kulit *Lumpy* dan kulit sehat.

Pada Gambar 4 merupakan arsitektur MobileNetV2 pada saat melakukan klasifikasi dan pada Tabel 1 merupakan perincian lapisan-lapisan yang ada pada arsitektur MobileNetV2. MobileNetV2 menambahkan dua fitur baru yaitu linear *bottleneck* dan *shortcut connections* antar *bottlenecks* [13]. Pada *linear residual block* (*Bootleneck*) terdiri dari tiga lapisan konvolusi yaitu *expansion layer, depthwise convolution,* dan *projection layer*. *Ekspansion layer* merupakan konvolusi 1x1 yang bertujuan untuk memperluas jumlah saluran data sebelum masuk ke *depthwise convolution* sehingga *ekspansion layer* memiliki lebih banyak saluran keluaran daripada saluran masukan. Seberapa banyak data yang diperluas diberikan oleh faktor ekspansi. *Depthwise convolution* melakukan penyaringan apapun yang penting pada tahap jaringan ini. Lapisan *projection* mengembalikan data menjadi semula atau lebih kecil.



Gambar 4. Arsitektur MobileNetV2 [14]

Tabel 1. Perincian Arsitektur MobileNetV2 [15]

Input	Operator	t	c	n	s
224x224x3	conv2d	-	32	1	2
112x112x32	bottleneck	1	16	1	1
112x112x16	bottleneck	6	24	2	2

Input	Operator	t	c	n	s
56x56x24	bottleneck	6	32	3	2
28x28x32	bottleneck	6	64	4	2
14x14x64	bottleneck	6	96	3	1
14x14x96	bottleneck	6	160	3	2
7x7x160	bottleneck	6	320	1	1
7x7x320	conv2d 1x1	-	1280	1	1
7x7x1280	globalavgpoll	-	1280	1	-
1x1x1280	dense	-	k	-	-

Pada Tabel 2 merupakan spesifikasi arsitektur MobileNetV2 yang telah dimodifikasi dengan teknik *transfer learning* dimana dimulai dengan *input layer*, model menerima gambar dengan ukuran 150x150 dan tiga saluran warna (RGB). Selanjutnya, lapisan MobileNetV2 yang sudah dilatih sebelumnya digunakan sebagai fitur ekstraktor utama pada lapisan dalam *bottleneck* dengan total 2.257.984 parameter, dimana semua parameter tersebut tidak dapat dilatih ulang (*non-trainable*). Selanjutnya, keluaran dari jaringan diproses oleh lapisan *global average pooling* yang mereduksi dimensi keluaran menjadi 1280 unit. Model ini memiliki dua lapisan *dense* atau *fully connected*, dengan lapisan pertama terdiri dari 512 unit dan 655.872 parameter, diikuti oleh lapisan kedua dengan satu unit untuk menghasilkan keluaran akhir menggunakan aktivasi sigmoid yang membedakan dua kelas. Lapisan kedua memiliki total 513 parameter. Secara keseluruhan, model ini memiliki 2.914.369 parameter, di mana 656.385 di antaranya dapat dilatih (*trainable*) untuk menyesuaikan model dengan data spesifik yang digunakan.

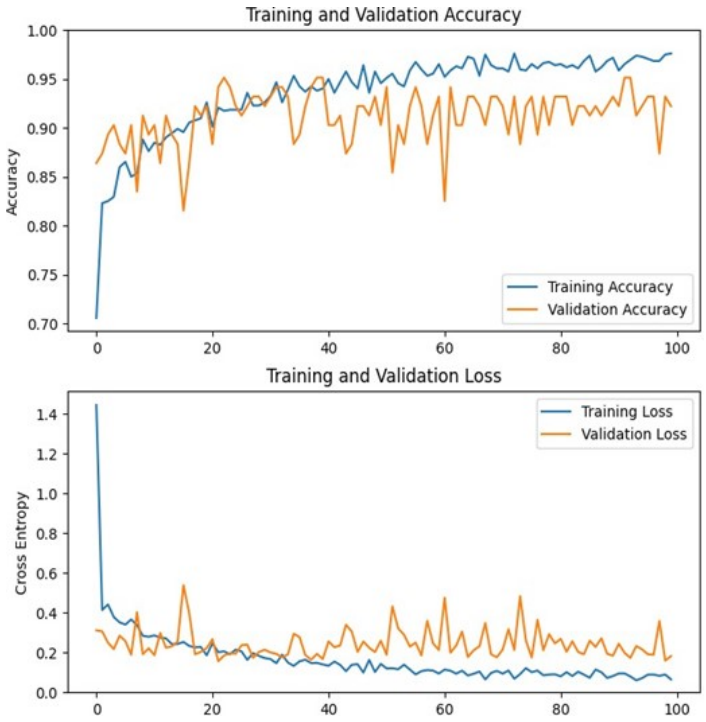
Tabel 2. Hasil Klasifikasi Arsitektur MobileNetV2 dengan *Transfer Learning*

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 150, 150, 3)	0
MobileNet v2 _1.00_224	(None, 5, 5, 1280)	2,257,984
global_average_pooling2d_1	(None, 1280)	0
dense_2 (Dense)	(None, 512)	655,872
dense_3 (Dense)	(None, 1)	513
Total params		2,914,369
Trainable params		656,385
Non-trainable params		2,257,984

3.4. Pelatihan

Pada proses pelatihan, proses optimasi model merupakan suatu hal penting. Selain untuk meningkatkan nilai akurasi pelatihan model, *optimizer* umumnya digunakan untuk meminimalisir terjadinya *overfitting* selama proses pelatihan model [16]. Penelitian ini menggunakan *optimizer* RMSprop dengan parameter yang ditentukan *batch size* dengan nilai 45 dan jumlah *epoch* adalah 100. Nilai *batch size* dan jumlah *epoch* yang memberikan akurasi optimum ditentukan melalui eksperimen. Oleh karena itu, pada penelitian ini dilakukan pelatihan menggunakan beberapa nilai *batch size* dan jumlah *epoch* sehingga diperoleh hasil akurasi tertinggi. Pada Gambar 5 merupakan grafik pelatihan model yang menunjukkan perbandingan akurasi dan loss antara proses pelatihan dan validasi.

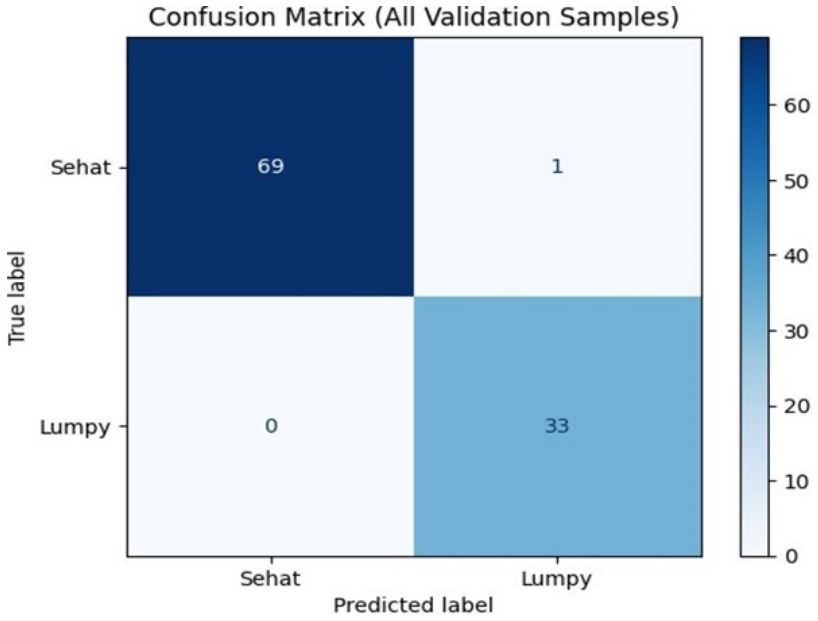
Dari grafik akurasi, terlihat bahwa model menunjukkan tren peningkatan yang seimbang antara data pelatihan dan validasi, yang mengindikasikan bahwa model mampu mempelajari pola data dengan baik tanpa mengalami *overfitting*. Garis biru pada grafik akurasi merepresentasikan akurasi selama proses pelatihan, sedangkan garis *orange* merepresentasikan akurasi selama proses validasi. Selain itu, grafik *loss* juga menunjukkan penurunan yang konsisten baik pada data pelatihan maupun validasi, di mana garis biru pada grafik *loss* menggambarkan loss selama pelatihan dan garis *orange* menggambarkan *loss* selama validasi. Hal ini menandakan bahwa model berhasil meminimalkan kesalahan prediksi baik selama pelatihan maupun validasi. Berdasarkan grafik tersebut, dapat disimpulkan bahwa model yang dihasilkan memiliki performa yang baik dan berada dalam kategori *good fit* untuk melakukan klasifikasi.



Gambar 5. Grafik Akurasi dan Loss Arsitektur MobileNetV2

3.5. Evaluasi

Penentuan performa suatu model klasifikasi dapat dilihat dari parameter pengukuran performanya yaitu tingkat akurasi, sensitivitas, dan presisi. Perhitungan metrik pengukuran tersebut diperlukan sebuah matriks yang biasa disebut *confusion matrix*. *Confusion matrix* mengandung nilai yatu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Seluruh kemungkinan kejadian sebenarnya positif (P) dan seluruh kemungkinan kejadian sebenarnya negatif (N) [15]. Pada Gambar 6, dapat dilihat hasil *confusion matrix* yang dihasilkan oleh Arsitektur MobileNetV2 pada data validasi.



Gambar 6. *Confussion Matrix* Hasilkan Model Arsitektur MobileNetV2

Untuk menghitung *Precission* (presisi) dan *Recall* (sensitivitas) yang diperoleh dapat menggunakan Persamaan (1) dan Persamaan (2) [17]. Presisi kelas Lumpy diperoleh sebesar 0.97, sedangkan *recall* kelas *Lumpy* sebesar 1. Adapun Presisi kelas sehat diperoleh sebesar 1, sedangkan *recall* kelas sehat sebesar 0.98.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Dari hasil perhitungannya, dapat disimpulkan bahwa model yang telah dihasilkan memiliki performa yang bagus dalam melakukan klasifikasi dan pendeteksian. *Confusion Matrix* menggunakan data validasi sebagai acuan model memiliki ketepatan memprediksi walaupun telah diberi pelatihan yang dimana data validasi untuk keseluruhan kelas adalah 103 data yang digunakan. Selebihnya digunakan untuk pelatihan karena pelatihan membutuhkan lebih banyak data agar model dapat mempelajari pola data yang akan digunakan saat klasifikasi. Hasil dari pengujian juga menunjukan bahwa model memiliki akurasi yang bagus saat diberi citra acak saat melakukan klasifikasi. Pada evaluasi ini, berbagai citra acak diuji dengan model yang telah dilatih untuk memastikan bahwa prediksi yang diberikan akurat dan dapat diandalkan. Proses evaluasi melibatkan pengujian model dengan citra yang belum pernah dilihat sebelumnya dan membandingkan hasil prediksi dengan label yang sebenarnya. Hasil dari evaluasi ini akan menunjukkan seberapa baik model dapat menggeneralisasi pengetahuan yang diperolehnya selama pelatihan ke data baru seperti pada Gambar 7 ditunjukkan model klasifikasi kelas sehat dan kelas *Lumpy*.

(a) (b)

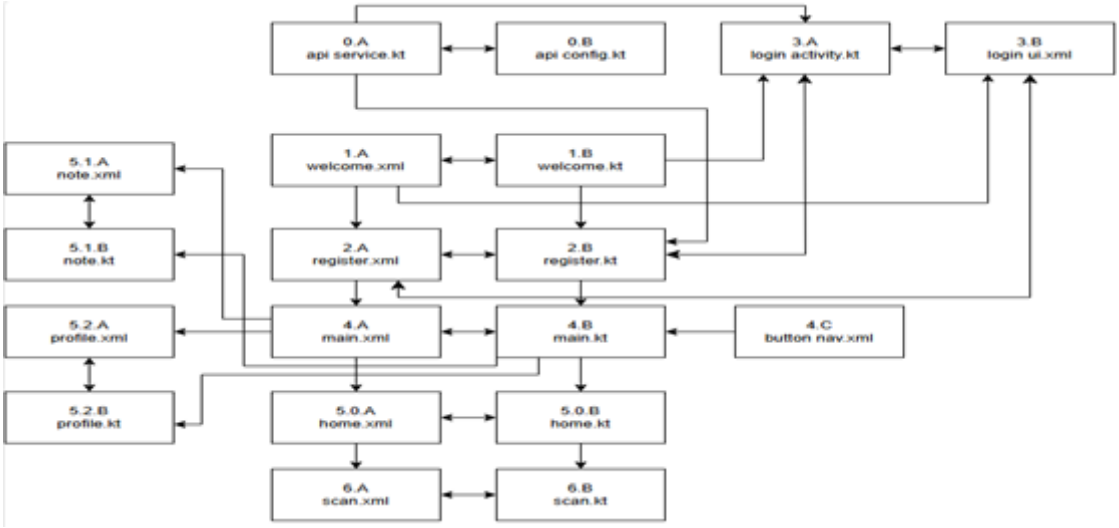
Gambar 7. Model Prediksi; (a) Prediksi kelas sehat; (b) Prediksi kelas Lumpy [18]

Dari sisi performa, model menunjukkan kemampuan yang baik dalam mengklasifikasikan citra kulit ternak sehat dan yang terinfeksi LSD, dengan hasil yang konsisten antara data latih dan data validasi, sehingga tidak terjadi *overfitting*. Pelatihan model menghasilkan akurasi 95.91% dan *loss* 0.1166 serta nilai akurasi pengujian atau validasi model sebesar 88.35% dan nilai *loss* sebesar 0.3058. Pada VGG16 didapat nilai akurasi pelatihan sebesar 95.31% dan nilai *loss* sebesar 0,1292, serta nilai akurasi pengujian atau validasi model sebesar 96.88% dan nilai *loss* sebesar 0.102 [3]. pada DenseNet121 didapat nilai akurasi pelatihan sebesar 46.65% dan nilai *loss* sebesar 0.7814, serta nilai akurasi pengujian atau validasi model sebesar 80.21% dan nilai loss sebesar 0.3937 [4]. Adapun berbagai faktor yang menyebabkan terjadinya variasi akurasi diantaranya seperti kurangnya *dataset,* citra yang kurang jelas atau buruk. Alasan digunakannya arsitektur MobileNetV2 pada penelitian ini, karena arsitekturnya yang dilatih oleh Google dan memiliki kemampuan pembuatan model yang cukup ringan dan dapat meningkatkan akurasi yang cukup tinggi terhadap *dataset* yang sedikit dan dapat diterapkan pada aplikasi

Android yang juga dikembangkan oleh Google.

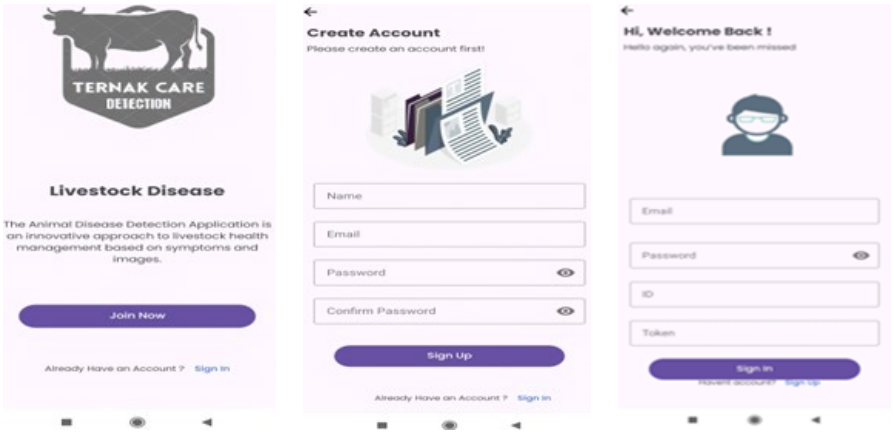
3.6. Implementasi

Hasil penelitian diimplementasikan dalam bentuk aplikasi Android dengan menerapkan API yang dihasilkan *Google Cloud Platform* untuk fitur aplikasi seperti *login register* sebagai fitur penunjang. Pembuatan aplikasi menggunakan *framework flask* yang merupakan web *framework* karena *flask* bertindak sebagai *framework* aplikasi dan tampilan web menggunakan *flask* dan bahasa Python untuk membuat web terstruktur dan mengelola perilaku web dengan lebih mudah [19]. Model yang telah didapatkan dari proses pelatihan akan dimasukkan ke dalam fungsi dari *framework flask* dimana model ini akan digunakan dalam proses klasifikasi pada aplikasi. Proses klasifikasi dilakukan dengan memasukkan satu citra baru penyakit kulit *Lumpy* dan sistem akan melakukan proses klasifikasi dan menampilkan hasil klasifikasi. Dalam penerapan menggunakan aplikasi berbasis Android, terdapat alur dan komponen yang dapat menyusun aplikasi beserta fitur yang akan digunakan seperti yang ditunjukan Gambar 8.



Gambar 8. *File mapping* antarmuka dan *backend* aplikasi

Gambar 8 merupakan diagram alur kerja aplikasi Android yang menggambarkan interaksi antara berbagai komponen XML dan Kotlin. Diagram ini memperlihatkan bagaimana halaman-halaman seperti api service (0.A), api config (0.B), *welcome* (1.A, 1.B), *register* (2.A, 2.B), *login (*3.A, 3.B), *main* (4.A, 4.B), *home* (5.0.A, 5.0.B), *note* (5.1.A, 5.1.B), *profile* (5.2.A, 5.2.B) dan *scan* (6.A, 6.B) yang saling terhubung dan berfungsi secara terpadu. Setiap file XML bertanggung jawab atas tampilan antarmuka pengguna, sementara *file* Kotlin terkait menangani logika program. Proses dimulai dengan autentikasi pengguna melalui *welcome* dimana *welcome* dapat mengarahkan pengguna ke halaman *register* atau *login*, kemudian pengguna dapat menavigasi ke halaman lain seperti halaman *register* jika pengguna belum memiliki akun untuk mengakses aplikasi, halaman *login* jika pengguna telah memiliki akun untuk mengakses aplikasi, halaman *main* merupakan halaman utama yang dilihat pertama kali oleh pengguna setelah memasuki aplikasi yang memuat halaman *home* dan mengatur perpindahan halaman dari *home* ke *profile* atau *note* menggunakan *button nav* yang berfungsi sebagai tombol navigasi, halaman *note* merupakan halaman catatan yang diberikan oleh pengembang kepada penggunannya, halaman *profile* merupakan halaman dimana pengguna dapat mengganti data diri yang terdaftar pada aplikasi, fitur utama pada aplikasi ini adalah fitur klasifikasi pada halaman *scan* dimana untuk mengaksesnya terdapat tombol *'take picture', api service* dan *api config* merupakan sistem yang mengatur transmisi dan respon *login register* pada API. Gambar 9 ditunjukkan tampilan dari aplikasi yang dibangun.

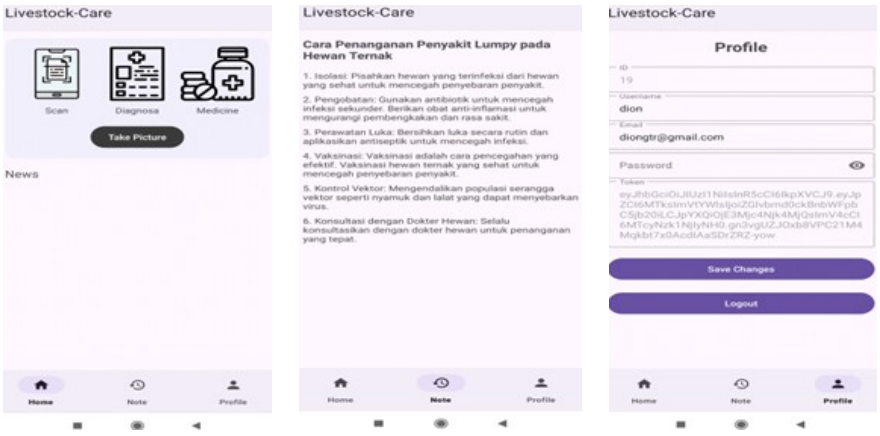


(a) (b) (c)

Gambar 9. Tampilan Aplikasi; (a). Tampilan *welcome;* (b) Tampilan *register;* (c) Tampilan *login.*

Gambar 9a merupakan tampilan pada telepon seluler halaman *welcome* pengguna akan diarahkan dengan sambutan pada aplikasi. Gambar 3b merupakan tampilan halaman *login* terdapat *form* pengisian yang terdiri dari email dan *password*. Bila pengguna sudah memiliki akun, dapat mengisi email dan *password* yang telah terdaftar dan akan memiliki id dan token yang terisi secara otomatis saat *login*. Gambar 9c merupakan tampilan halaman *Register,* apabila pengguna belum memiliki akun, pengguna dapat membuat akun terlebih dahulu dengan memasukan *username*, email, *password* dan *confirm password* sehingga secara tidak langsung pengguna telah mendapatkan id dan token yang diperlukan untuk mengakses aplikasi ini. Jika berhasil *login*, maka akan diarahkan ke halaman utama atau *home*.

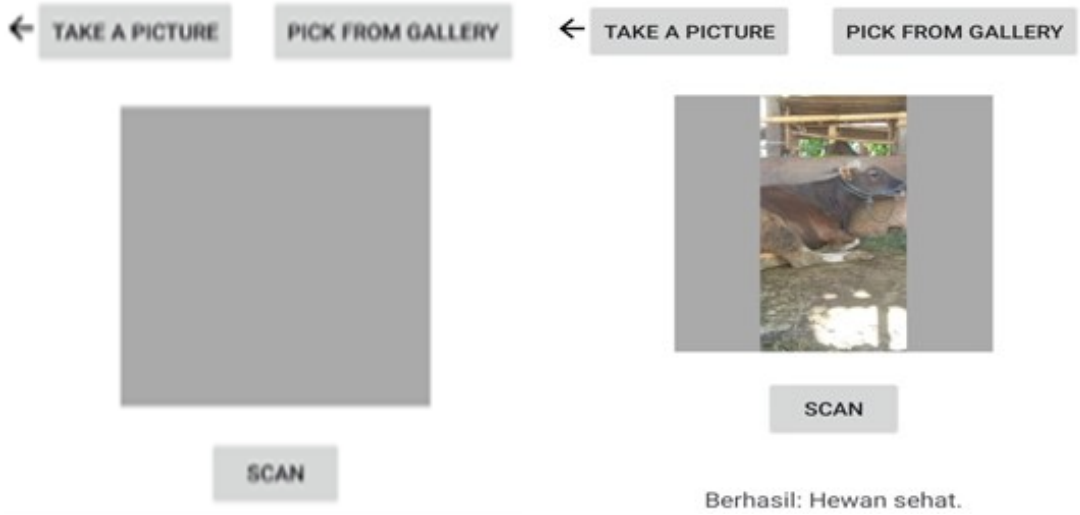
Gambar 10a merupakan tampilan *home*, merupakan halaman utama dimana pengguna dapat mengakses fitur utama yakni *scan* hewan ternak dengan tombol "*take a picture*". Setelah mengakses fitur utama pengguna akan di minta untuk memilih citra hewan ternak seperti sapi yang akan di *scan*. Gambar 10b merupakan tampilan halaman *note* merupakan halaman informasi terbaru yang diberikan oleh *developer* kepada pengguna aplikasi. Gambar 10c merupakan tampilan halaman *profile* merupakan tempat untuk pengguna melakukan perubahan terhadap data yang digunakan untuk *login* pada aplikasi perubahan data hanya dapat dilakukan pada *username* dan *password* saja juga pengguna dapat *logout* akun mereka dari aplikasi yang dimana harus pengguna harus memasukan kembali data mereka saat *login*.



(a) (b) (c)

Gambar 10. Tampilan Halaman Utama; (a). Tampilan Halaman Home; (b) Tampilan Halaman *Note;* (c) Tampilan Halaman *Profile*

Pada Gambar 11a merupakan tampilan halaman fitur *scan* sebelum diberi masukan citra yang akan dilakukan klasifikasi yang memungkinkan pengguna untuk mengambil foto secara langsung melalui akses kamera ataupun mengambil gambar yang telah diambil sebelumnya pada galeri dan setelah mengupload sebuah citra, model tersebut akan membaca dan memprediksi kemungkinan yang dialami pada hewan ternak dan akan memberikan informasi penyakit tersebut. Pada Gambar 11b merupakan tampilan halaman hasil fitur *scan* setelah melakukan klasifikasi citra yang termasuk kelas sehat. Aplikasi akan memberitahukan bahwa citra yang dimasukkan pengguna pada klasifikasi tergolong sehat. Gambar 12 merupakan tampilan halaman hasil fitur *scan* setelah melakukan klasifikasi citra yang termasuk kelas *Lumpy.* Aplikasi akan memberitahukan bahwa citra yang dimasukkan pengguna pada klasifikasi mengalami *Lumpy* dan terdapat penjelasan tentang *Lumpy.*



(a) (b)

Gambar 11. Halaman *Scan;* (a) Tampilan Fitur *Scan*; (b) Tampilan Hasil Fitur Scan Kelas Sehat

Gambar 12. Tampilan Halaman Hasil Fitur *Scan* Kelas *Lumpy* [20]

4. KESIMPULAN

Penelitian ini berhasil mengembangkan aplikasi berbasis Android untuk deteksi penyakit LSD pada hewan ternak menggunakan metode arsitektur MobileNetV2. Kebaruan dari penelitian ini terletak pada penggunaan arsitektur MobileNetV2 yang lebih ringan dan dioptimalkan untuk perangkat *mobile*, sehingga memungkinkan deteksi penyakit secara cepat di lapangan. Hasil penelitian menunjukkan bahwa aplikasi yang dikembangkan mampu mendeteksi LSD dengan akurasi 95,91% dan memberikan kemudahan bagi peternak dalam mencegah penyebaran penyakit melalui deteksi dini. Penelitian ini juga membuka peluang untuk pengembangan lebih lanjut. Saran untuk penelitian selanjutnya mencakup peningkatan ukuran *dataset* serta penambahan fitur deteksi gejala lain yang terkait dengan penyakit LSD untuk memperluas kegunaan aplikasi.

UCAPAN TERIMA KASIH

Penulis mengucapkan terimakasih kepada BANGKIT ACADEMY dan Merdeka Belajar Kampus Merdeka atas pelatihan yang dituangkan pada penelitian ini.

DAFTAR PUSTAKA

[1] N. Vasković *et al.*, "Morphological Characteristics of Skin Lesions in Cattle Naturally Infected with Lumpy Skin Disease Virus in Serbia," *Acta Veterinaria*, vol. 69, no. 4, pp. 369–378, Dec. 2019. DOI: 10.2478/acve-2019-0031.

[2] I. Sendow *et al.*, "Lumpy Skin Disease: Ancaman Penyakit Emerging Bagi Kesehatan Ternak Sapi Di Indonesia," *Indonesian Bulletin of Animal and Veterinary Sciences*, vol. 31, no. 2, p. 85, Jun. 2021. DOI: 10.14334/wartazoa.v31i2.2739.

[3] T. Sentoso *et al.*, "Identifikasi Lumpy Skin Disease pada Ternak Sapi dengan Klasifikasi Citra menggunakan Metode Convolutional Neural Network," *Sistemasi: Jurnal Sistem Informasi*, vol. 13, no. 3, pp. 864–873, May 2024. DOI: 10.32520/stmsi.v13i3.2569.

[4] N. S. Alfiansyah dan Y. Litanianda, "Identifikasi Lumpy Skin Disease Menggunakan Tensorflow dengan Metode Convolutional Neuron Network," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 4, pp. 7330–7336, Jul. 2024. DOI: 10.36040/jati.v8i4.10238.

[5] A. Fikri, I. Aknuranda, dan F. Pradana, "Pengembangan Sistem Informasi Aspirasi Online Berbasis Web Menggunakan Pemodelan Reuse-Oriented Development (Studi Kasus : DPM Universitas Brawijaya)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 2, pp. 1174–1183, 2019.

[6] S. Kumar, *Lumpy Skin Images Dataset*, Aug. 2022. DOI: 10.17632/W36HPF86J2.1.

[7] L. Hakim, Z. Sari, dan H. Handhajani, "Klasifikasi Citra Pigmen Kanker Kulit Menggunakan Convolutional Neural Network," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 2, pp. 379–385, Apr. 2021. DOI: 10.29207/resti.v5i2.3001.

[8] D. B. Setyohadi, F. A. Kristiawan, dan E. Ernawati, "Perbaikan Performansi Klasifikasi dengan Preprocessing Iterative Partitioning Filter Algorithm," *Telematika*, vol. 14, no. 1, pp. 12–20, Apr. 2017. DOI: 10.31315/telematika.v14i01.1960.

[9] F. Alghifari dan D. Juardi, "Penerapan Data Mining Pada Penjualan Makanan dan Minuman Menggunakan Metode Algoritma Naïve Bayes: Studi Kasus: Makan Barbeque Sepuasnya," *Jurnal Ilmiah Informatika*, vol. 9, no. 02, pp. 75–81, 2021.

[10] A. Beikmohammadi, K. Faez, dan A. Motallebi, "SWP-LeafNET: A novel multistage approach for plant leaf identification based on deep CNN," *Expert Systems with Applications*, vol. 202, p. 117 470, Sep. 2022. DOI: 10.1016/j.eswa.2022.117470.

[11] J. Hang *et al.*, "Classification of Plant Leaf Diseases Based on Improved Convolutional Neural Network," *Sensors*, vol. 19, no. 19, p. 4161, Sep. 2019. DOI: 10.3390/s19194161.

[12] G. Kang *et al.*, "3D multi-view convolutional neural networks for lung nodule classification," *PLOS ONE*, vol. 12, no. 11, Y. Deng, Ed., e0188290, Nov. 2017. DOI: 10.1371/journal.pone.0188290.

[13] F. Zaelani dan Y. Miftahuddin, "Perbandingan Metode EfficientNetB3 dan MobileNetV2 Untuk Identifikasi Jenis Buah-buahan Menggunakan Fitur Daun: Metode EfficientNetB3 dan MobileNetv2," *Jurnal Ilmiah Teknologi Infomasi Terapan*, vol. 9, no. 1, pp. 1–11, Dec. 2022. DOI: 10.33197/jitter.vol9.iss1.2022.911.

[14] W. Wang *et al.*, "A New Image Classification Approach via Improved MobileNet Models with Local Receptive Field Expansion in Shallow Layers," *Computational Intelligence and Neuroscience*, vol. 2020, pp. 1–10, Aug. 2020. DOI: 10.1155/2020/8817849.

[15] M. N. Winnarto, "Penerapan Arsitektur Mobilenetv2 Pada Klasifikasi Penyakit Daun Teh," Thesis, Universitas Nusa Mandiri, Jakarta, 2021.

[16] R. Richo *et al.*, "Analisis Pengaruh Optimizer pada Model CNN untuk Identifikasi Cacat pada Perekat Kemasan," *SISFOTENIKA*, vol. 13, no. 2, pp. 217–229, Jul. 2023. DOI: 10.30700/jst.v13i2.1447.

[17] A. B. Prakosa, H. Hendry, dan R. Tanone, "Implementasi Model Deep Learning Convolutional Neural Network (CNN) pada Citra Penyakit Daun Jagung untuk Klasifikasi Penyakit Tanaman," en, *Jurnal Pendidikan Teknologi Informasi (JUKANTI)*, vol. 6, no. 1, pp. 107–116, Apr. 2023. DOI: 10.37792/jukanti.v6i1.919.

[18] Dinas Pertanian dan Ketahanan Pangan Provinsi Bali, *Waspada Penyakit Lumpy Skin Diseases (LSD)*, Mar. 2022.

[19] P. T. L. Dewi, F. Dewanta, dan M. A. Nugroho, "Implementasi Machine Learning Model Deployment Pada Website Pemantauan Kondisi Sungai Citarum Menggunakan Platform As-A-Service," *eProceedings of Engineering*, vol. 9, no. 6, pp. 1–11, 2023.

[20] E. S. Bimo, *Wabah Lumpy Skin Disease Kian Merebak, Laos Perpanjang Larangan Impor Hewan Ternak Sapi*, Jul. 2021.

[Halaman ini sengaja dikosongkan.]