# DETEKSI PENYAKIT KULIT KEPALA MENGGUNAKAN *MOBILENETV2* BERBASIS ANDROID

**LAPORAN TUGAS AKHIR**

Laporan ini disusun guna memenuhi salah satu syarat untuk menyelesaikan program studi Teknik Informatika S-1 pada Fakultas Teknologi Industri Universitas Islam Sultan Agung


Disusun Oleh :

**MUHAMMAD I'ZZUDIN**

**NIM 32602000047**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM SULTAN AGUNG**

**SEMARANG**

**2024**

***FINAL PROJECT***

***SCALP DISEASE DETECTION USING MOBILENETV2***

***BASED ON ANDROID***

*Proposed to complete the requitement to obtain a bachelor's degree (S1)*
*At Informatics Engineering Departement of Industrial Technology Faculty*
*Sultan Agung Islamic University*

***Arranged By :***

**MUHAMMAD I'ZZUDIN**
**NIM 32602000047**

**FAKULTAS TEKNOLOGI INDUSTRI**
**UNIVERSITAS ISLAM SULTAN AGUNG**
**SEMARANG**
**2023**

**LEMBAR PENGESAHAN PEMBIMBING**

Laporan Tugas Akhir dengan judul "DETEKSI PENYAKIT KULIT KEPALA MENGGUNAKAN MOBILENETV2 BERBASIS ANDROID" ini disusun oleh :

Nama : Muhammad I'zzudin

NIM : 32602000047

Program Studi : Teknik Informatika

Telah disahkan oleh dosen pembimbing pada :

Hari : Kamis

Tanggal : 29 Februari 2024

Mengesahkan,

| Pembimbing I | Pembimbing II |
|---|---|
| Ir. Sri Mulyono, M.Eng | Bagus Satrio Waluyo Poetro, S.Kom., M.Cs |
| NIDN. 0626066601 | NIDN. 1027118801 |

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Sultan Agung

Ir. Sri Mulyono, M.Eng
NIDN. 0626066601

**LEMBAR PENGESAHAN PENGUJI**

Laporan tugas akhir dengan judul "DETEKSI PENYAKIT KULIT KEPALA MENGGUNAKAN MOBILENETV2 BERBASIS ANDROID" ini telah dipertahankan di depan dosen penguji Tugas Akhir pada :

Hari : Kamis

Tanggal : 29 Februari 2024

**TIM PENGUJI**

| Penguji I | Penguji II |
|---|---|
| Moch Taufik, ST., MIT | Ghufron, ST, M.Kom |
| NIDN. 0622037502 | NIDN. 0602079005 |

## SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama : Muhammad I'zzudin

NIM : 32602000047

Judul Tugas Akhir : DETEKSI PENYAKIT KULIT KEPALA MENGGUNAKAN MOBILENETV2 BERBASIS ANDROID

Dengan bahwa ini saya menyatakan bahwa judul dan isi Tugas Akhir yang saya buat dalam rangka menyelesaikan Pendidikan Strata Satu (S1) Teknik Informatika tersebut adalah asli dan belum pernah diangkat, ditulis ataupun dipublikasikan oleh siapapun baik keseluruhan maupun sebagian, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka, dan apbila di kemudian hari ternyata terbukti bahwa judul Tugas Akhir tersebut pernah diangkat, ditulis ataupun dipublikasikan, maka saya bersedia dikenakan sanksi akademis. Demikian surat pernyataan ini saya buat dengan sadar dan penuh tanggung jawab.

Semarang, 29 Februari 2024

Yang Menyatakan,

Muhammad I'zzudin

# KATA PENGANTAR

Dengan mengucap Syukur alhamdulillah atas kehadirat Allah SWT yang telah memberikan Rahmat dan kanunianya kepada penulis, sehingga dapat menyelesaikan Tugas Akhir dengan judul "Deteksi Penyakit Kulit Kepala Menggunakan *MobileNetV2* Berbasis Android" ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar sarjana (S-1) pada Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang.

Tugas Akhir ini disusun dan dibuat dengan adanya bantuan dari berbagai pihak, materi maupun teknis, oleh karena itu saya selaku penulis mengucapkan terima kasih kepada:

1. Rektor Universitas Islam Sultan Agung Bapak Prof. Dr. H. Gunarto, S.H., M.H yang mengizinkan penulis menimba ilmu di kampus ini.
2. Dekan Fakultas Teknologi Industri Ibu Dr. Novi Marlyana, ST., M.T.
3. Dosen pembimbing I penulis Bapak Ir. Sri Mulyono, M.Eng yang telah meluangkan waktu dan memberi ilmu.
4. Dosen pembimbing II penulis Bapak Bagus Satrio Waluyo Poetro, S.Kom., M.Cs.
5. Orang tua penulis yang telah mengizinkan untuk menyelesaikan laporan ini dan senantiasa memberikan semangat serta doa agar tugas akhir ini berjalan dengan lancar.
6. Dan kepada semua pihak yang tidak dapat saya sebutkan satu persatu.

Dengan segala kerendahan hati, penulis menyadai masih terdapat banyak kekurangan dari segi kualitas maupun dari ilmu pengetahuan dalam penyusunan laporan, sehingga penulis mengharapkan adanya saran dan kritikan yang bersifat membangun demi kesempurnaan laporan ini di masa mendatang.

Semarang, 29 Februari 2024

Muhammad I'zzudin

**DAFTAR ISI**

# DAFTAR TABEL

# DAFTAR GAMBAR

# ABSTRAK

Penyakit kulit kepala adalah masalah dapat memengaruhi kesehatan serta penampilan seseorang. Kerontokan rambut yang berlebihan, ketombe, dan infeksi kulit kepala menjadi masalah umum yang memerlukan deteksi dini dan tindakan cepat. Oleh karena itu, dibutuhkan sistem yang dapat mengklasifikasikan jenis penyakit rambuut yang sedang dialami. Dengan mengetahui lebih cepat jenis penyakit kulit kepala yang mereka derita, masayrakat dapat dengan cepat melakukan tindangan yang diperlukan. Penelitian ini memanfaatkan *convolution neural network* dengan arsitektur *MobileNetV2* untuk klasifikasi jenis penyakit kulit kepala. *MobileNetV2* dipilih karena merupakan model yang efisien dalam komputasi dan memiliki performa tinggi dalam tugas klasifikasi gambar yang dapat diimplementasikan dalam sistem berbasis android. Beberapa uji coba konfigurasi dirancang untuk menemukan model yang optimal. Uji coba tersebut mencangkup menggunaan *learning rate*, jumlah *batch size*, dan *epoch*. Model dengan kinerja terbaik ditemukan dengan konfigurasi *optimizer Adam*, *learning rate* 0.001, *batch size* 30, dan *epoch* 6. Menghasilkan *accuracy* 0.94, *loss* 0.19, *validation accuracy* 0.93, dan *validation loss* 0.1844. pada tahap *testing* model ini mencapai *accuracy* 0.93, *precission* 0.94, *recall* 0.93, dan *fi-score* sebesar 0.93.
**Kata Kunci**: Penyakit Kulit Kepala, Klasifikasi, *MobileNetV2*, Android

# *ABSTRACT*

*Scalp disease is a problem that can affect a person's health as well as appearance. Excessive hair loss, dandruff and scalp infections are common problems that require early detection and quick action. Therefore, there is a need for a system that can classify the type of scalp disease being experienced. By knowing sooner the type of scalp disease they suffer from, people can quickly take the necessary action. This research utilizes convolution neural network with MobileNetV2 architecture for classification of scalp disease types. MobileNetV2 was chosen because it is a computationally efficient model and has high performance in image classification tasks that can be implemented in an android-based system. Several configuration trials were designed to find the optimal model. The tests included learning rate, batch size, and epochs. The best performing model was found with Adam optimizer configuration, learning rate 0.001, batch size 30, and epoch 6. Resulting in accuracy 0.9458, loss 0.1901, validation accuracy 0.9333, and validation loss 0.1844. in the testing phase this model achieved accuracy 0.93, precission 0.94, recall 0.93, and fi-score of 0.93.*
***Keywords**: Scalp Disease, Classification, MobileNetV2, Android*

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Salah satu organ yang memerlukan perhatian lebih adalah kulit. Kulit berperan sebagai perlindungan bagi organ-organ di bawahnya. Terutama, kulit kepala yang memiliki peran penting karena dapat mempengaruhi kesehatan rambut. Di Indonesia, kondisi iklim tropis menyebabkan peningkatan debu dan keringat, sehingga menjaga kebersihan kulit kepala dan rambut melalui kebiasaan mencuci secara teratur menjadi suatu kebutuhan yang sangat penting (Ambarwati dkk., 2020).

*Convolutional Neural Network* adalah salah satu teknik pada bidang kecerdasan buatan yang telah terbukti efektif dalam mengklasifikasi gambar (Ibnul Rasidi dkk., 2022). Metode ini telah digunakan dalam berbagai aplikasi, termasuk dalam bidang kesehatan untuk klasifikasi penyakit kulit (Triyono dkk., 2008).

Dalam upaya meningkatkan efektivitas klasifikasi penyakit kulit kepala, penelitian ini memilih untuk mengimplementasikan arsitektur *MobileNetV2* pada CNN. *MobileNetV2* merupakan salah satu arsitektur yang terkenal karena keefisienan komputasinya dan kemampuannya dalam menangani tugas klasifikasi gambar dengan performa tinggi (Hendri Butar-Butar dan Marpaung, 2023). Dengan mengintegrasikan *MobileNetV2*, diharapkan sistem klasifikasi dapat bekerja secara efisien pada *platform mobile*, seperti aplikasi Android.

Selain itu, penggunaan smartphone di Indonesia semakin meningkat, khususnya smartphone android. Pada tahun 2022, 67,88% penduduk Indonesia sudah memiliki ponsel. Hal ini menunjukkan peningkatan dibandingkan tahun sebelumnya, di mana angka penetrasi mencapai 65,87% (Badan Pusat Statistik Indonesia, 2022). Hal ini membuka peluang untuk menciptakan aplikasi kesehatan yang dapat membuat informasi kesehatan lebih mudah diakses dan diperoleh dengan cepat oleh masyarakat..

Berdasarkan hal tersebut, penelitian ini bertujuan untuk menerapkan metode CNN dengan arsitektur *MobileNetV2* dalam klasifikasi penyakit kulit kepala pada

aplikasi android. Diharapkan, hasil penelitian ini dapat membantu masyarakat dalam mengidentifikasi penyakit kulit kepala dan sebagai langkah awal dalam penanganannya. Selain itu, penelitian ini diharapkan dapat berkontribusi dalam pengembangan aplikasi kesehatan, terutama yang fokus pada masalah kesehatan kulit kepala.

## 1.2 Perumusan Masalah

Bagaimana mengimplementasikan metode *Convolutional Neural Network* dengan arsitektur *MobileNetV2* untuk mendeteksi penyakit kulit kepala yaitu *alopecia areata, folliculitis,* dan *head lice* berbasis android.

## 1.3 Pembatasan Masalah

Berdasarkan masalah tersebut maka batasan masalah yang akan dibuat adalah sebagai berikut :

1. Penelitian ini memanfaatkan dataset yang tersedia dari *Kaggle* (https://www.kaggle.com/datasets/sundarannamalai/hair-diseases/data) yang mencakup berbagai kondisi kulit kepala yang terkait dengan penyakit. Dalam penelitian ini hanya akan difokuskan pada klasifikasi 3 jenis penyakit kulit kepala, yaitu *Alopecia Areata, Folliculitis,* dan *Head Lice.*
2. Total data yang digunakan sebanyak 180 citra, yang terdiri dari 60 citra untuk masing-masing class.
3. Penelitian ini menggunakan algoritma *Convolutional Neural Network* dengan arsitektur *MobileNetV2*.
4. Pada penelitian ini menggunakan platform android.
5. Pengambilan gambar dilakukan menggunakan kamera belakang untuk menghindari gambar buram, sehingga dapat mengurangi potensi kesalahan pada proses deteksi.

## 1.4 Tujuan

Mengimplementasikan metode *Convolutional Neural Network* dengan arsitektur *MobileNetV2* untuk mengklasifikasikan penyakit kulit kepala yaitu *alopecia areata, folliculitis,* dan *head lice* dengan sistem berbasis android.

## 1.5 Manfaat

Berdasarkan analisis permasalahan dan tujuan penelitian, manfaat dari penelitian ini adalah memudahkan identifikasi berbagai jenis penyakit kulit kepala.

## 1.6 Sistematika Penelitian

Penulisan dalam membuat laporan tugas akhir mengikuti sistematika penulisan sebagai berikut :

**BAB I : PENDAHULUAN**

Menjelaskan mengenai latar belakang pemilihan judul, rumusan masalah penelitian, membuat batasan masalah, tujuan penelitian, manfaat hasil penelitian, dan sistematika penulisan.

**BAB II : TINJAUAN PUSTAKA DAN DASAR TEORI**

Menerangkan penelitian terkait yang telah dilakukan sebelumnya dan topik yang mendukung mengenai permasalahan yang dibahas dalam penelitian

**BAB III : METODE PENELITIAN**

Menjelaskan mengenai metode penelitian, yang mencakup metode dan langkah-langkah penting untuk menyelesaikan Tugas Akhir.

**BAB IV : HASIL PENELITIAN**

Menerapkan hasil dari penelitian yang dilakukan, termasuk pengolahan dataset, langkah-langkah dalam mengoperasikan sistem, implementasi antarmuka pengguna, dan analisis penelitian yang dilakukan.

**BAB V : KESIMPULAN DAN SARAN**

Menyajikan hasil kesimpulan, saran, dan rekomendasi untuk pihak yang tertarik melanjutkan pengembangan penelitian.

# BAB II
# TINJAUAN PUSTAKA DAN DASAR TEORI

## 2.1 TINJUAN PUSTAKA

Pada penelitian yang dilakukan oleh Sulistiyana dan Anardani, mereka melakukan perbandingan algoritma *Convolutional Neural Network* (CNN) dan *Support Vector Machine* (SVM) dalam aplikasi deteksi penyakit tanaman jagung melalui citra daun. Dataset berisi 3,732 citra daun jagung dengan empat kategori (Hawar, Jamur, Karat, dan Sehat). CNN dan SVM dilatih untuk mendeteksi penyakit tanaman jagung, dan hasil uji menunjukkan bahwa model CNN mencapai tingkat akurasi 98%, sedangkan SVM hanya mencapai 87% (Sulistiyana dan Anardani, 2023). Penelitian ini menegaskan bahwa CNN lebih efektif dalam menganalisis tugas-tugas kompleks seperti deteksi citra.

Dalam sebuah penelitian tahun 2019 dengan judul "Deteksi Objek Dan Pengenalan Karakter Plat Nomor Kendaraan Indonesia Menggunakan Metode CNN Berbasis Python" yang dilakukan oleh Harani dkk. Metode CNN berhasil dan efektif dalam mengidentifikasi huruf dan angka pada plat nomor (Harani dkk., 2019).

Pada penelitian yang berjudul "Pendeteksi Citra Masker Wajah Menggunakan CNN dan Transfer Learning" yang membandingkan empat arsitektur CNN, yaitu *MobileNetV2*, *VGG16*, *DenseNet201*, dan *Xception*, untuk mengembangkan sistem pendeteksian citra masker wajah. Hasil eksperimen menunjukkan bahwa *Xception* memiliki akurasi tertinggi rata-rata sebesar 0,988, tapi waktu komputasi yang diperlukan lebih lama, sementara *MobileNetV2* memiliki waktu komputasi paling cepat (4081 detik) dengan akurasi yang masih baik (0,981) (Naufal dan Kusuma, 2021). Penelitian ini memberikan informasi bahwa *MobileNetV2* memiliki keunggulan dalam waktu komputasi yang sangat cepat (4081 detik) dan akurasi yang masih tergolong tinggi (0,981). Keefisienan ini membuat *MobileNetV2* menjadi pilihan ideal untuk aplikasi Android yang memiliki sumber daya terbatas.

Pada penelitian yang dilakukan oleh Winnarto dkk, mereka melakukan penelitian tentang metode MobileNetV2 yang digunakan untuk mengklasifikasikan

jenis tumor otak. MobileNetV2 berhasil mengklasifikasikan jenis tumor otak dengan akurasi sebesar 88,64% dan nilai kerugian (loss) sebesar 0,3424.(Winnarto dkk., 2022). Dari penelitian ini menunjukkan bahwa *MobileNetV2* dapat digunakan secara efektif untuk klasifikasi citra gambar.

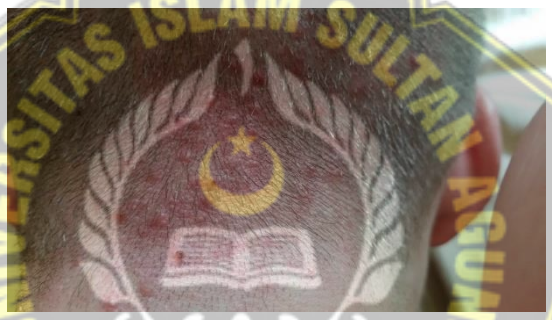
Penelitian dengan judul "Pendeteksian Wajah Bermasker Berbasis Android Menggunakan Algoritma CNN" yang membahas implementasi deteksi wajah bermasker di android menggunakan CNN. Penelitian ini diarahkan untuk memudahkan pendeteksian pengguna yang mengenakan masker di Kantor Desa Sumber Jaya. Algoritma CNN digunakan untuk klasifikasi gambar wajah dengan masker dan tanpa masker. Hasil pengujian menunjukkan keberhasilan aplikasi android mengimplementasikan CNN untuk mendeteksi wajah bermasker dan tanpa masker (Hutauruk dkk., 2022).

Berdasarkan beberapa penelitian diatas, penggunaan CNN dengan arsitektur *MobileNetV2* sangat tepat untuk aplikasi Android. Keberhasilan model CNN, terutama *MobileNetV2*, dalam menganalisis citra dengan tingkat akurasi yang tinggi menunjukkan efektivitasnya dalam berbagai konteks pengolahan citra. Oleh karena itu, penelitian ini akan membuat sebuah sistem klasifikasi penyakit kulit kepala menggunakan arsitektur MobileNetV2 yang akan diimplementasikan pada platform android.

## 2.2 DASAR TEORI

### 2.2.1 Penyakit Kulit Kepala

Kulit merupakan lapisan terluar pada tubuh manusia yang mudah diperhatikan dan berperan dalam pembentukan vitamin D dari pro-vitamin D karena paparan sinar matahari (Sinaga dkk., 2020). Namun kulit juga rentan terhadap berbagai kondisi yang memengaruhi kesehatan dan penampilannya, termasuk di area kepala. Jika kulit kepala terserang penyakit, berbagai dampak bisa muncul. Mulai dari gejala ringan seperti gatal hingga gejala yang lebih serius seperi sensasi perih dan nyeri pada kulit kepala (Fachri dkk., 2022). Contoh penyakit kulit kepala yang umum dialami adalah *folliculitis, alopecia areata, head lice,* dan lain-lain.



Gambar 2. 1 Folliculitis

*Folliculitis* adalah infeksi kulit yang disebabkan oleh bakteri seperti *Staphylococcus aureus* dan *Streptococcus sp.*, terutama pada awal folikel rambut. Jika infeksi melibatkan jaringan sekitarnya, disebut *furunkel*. Kondisi ini sering terjadi di daerah dengan iklim tropis dan kebersihan yang kurang baik. Pemeriksaan penunjang melibatkan mikroskop, kultur, dan resistensi antibiotik. Pengobatan utamanya adalah topikal, diikuti dengan terapi sistemik seperti pemberian antibiotik seperti *penisilin*, *ampicillin*, *erythromycin*, dan *clindamycin*. *Folliculitis* merupakan *pyoderma* yang terjadi pada awal folikel rambut, ditandai dengan *pustula folikular*, erosi, atau krusta. Penyebab utama adalah infeksi bakteri *Staphylococcus aureus*, meskipun dapat disebabkan juga oleh virus atau jamur (Abdi, 2020).

Gambar 2. 2 Alopesia Areata

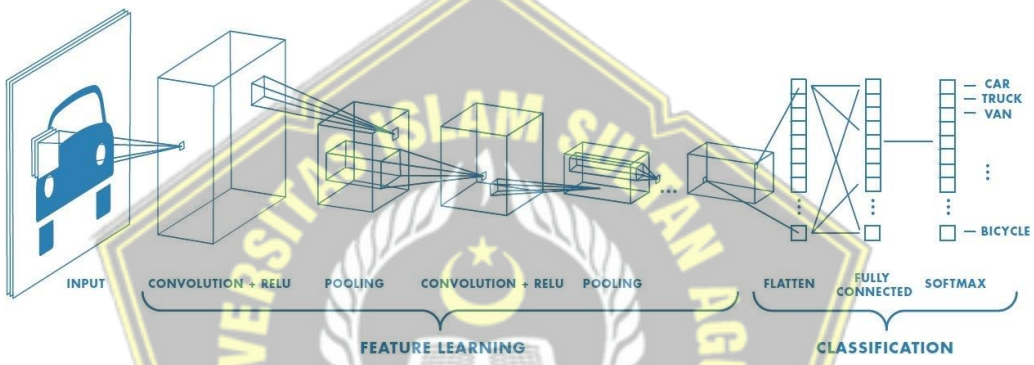
*Alopecia* adalah gangguan pertumbuhan rambut yang mengakibatkan rambut rontok dan berujung kebotakan. Kondisi ini dapat terjadi sejak lahir atau didapat, dengan manifestasi *cicatricial* atau *non-cicatricial alopecia* yang berkisar dari yang terlokalisasi hingga menyebar. Salah satu jenis *alopesia* yang umum ditemukan adalah *alopecia areata*, yang merupakan penyakit autoimun kronis yang ditandai oleh *cicatricial alopecia* (Toena dan Danarti, 2022). *Alopecia areata* umumnya terjadi pada kulit kepala. Luasnya kerontokan rambut dapat bervariasi mulai dari sebesar koin atau seluruh kulit kepala atau seluruh tubuh (Mellaratna dkk., 2023).

Gambar 2. 3 Head Lice

*Head lice* atau kutu kepala adalah parasit berukuran 1-3 mm yang bergantung pada darah dari kulit kepala manusia sebagai makanannya. Mereka umumnya menghisap darah terutama di bagian belakang tengkuk dan wilayah temporal. Kutu ini merupakan salah satu masalah kesehatan utama di seluruh dunia karena dapat menginfeksi individu dari segala kelompok umur, jenis kelamin, dan ras (Muslim dkk., 2022).

### 2.2.2 *Convolution Neural Network* (CNN)

*Convolutional Neural Network* (CNN) merupakan salah satu jenis arsitektur jaringan saraf tiruan yang dirancang khusus untuk mengolah data *grid* seperti gambar dan video. Dengan memanfaatkan operasi *convolution*, CNN memungkinkan model untuk secara otomatis mengekstrak *feature-feature* lokal dari input, seperti tepi, bentuk, dan tekstur. CNN sangat efektif dalam tugas-tugas pengenalan pola spasial, membuatnya sangat berguna dalam pengenalan objek, klasifikasi gambar, dan sejumlah aplikasi pengolahan citra lainnya (Ibnul Rasidi dkk., 2022).

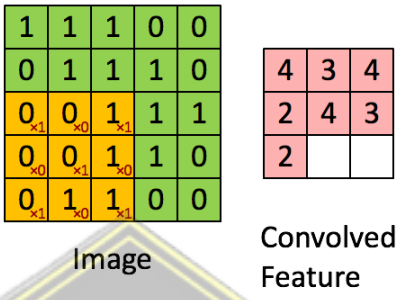



Gambar 2. 4 Struktur CNN

Secara umum, struktur CNN terbagi menjadi dua bagian utama, yakni *feature extraction* dan *classification*. Pada *feature extraction* bisa terdiri oleh *convolution layer*, *activation,* atau fungsi non linear seperti *ReLU*, dan *pooling layer* untuk membuat ukuran spasial atau lebar tinggi *feature*-nya mengecil seperti *max pooling*. Kemudian pada *classification,* hasil ektrakan *features* akan di-*flatten* menjadi vektor memanjang. vektor akan diproses lewat *fully connected* dan *softmax* untuk klasifikasi.

*Convolution layer* merupakan elemen kunci dalam CNN di mana sebagian besar komputasi dilakukan. Struktur dari *convolutional layer* ini terdiri dari sejumlah *neuron* yang tersusun sedemikian rupa sehingga membentuk suatu filter dengan dimensi panjang dan lebar dalam pixel (Peryanto dkk., 2020). Dalam mendesain convolution layer, beberapa hal yang perlu diset adalah kernel *size* dan *stride*. *Kernel size* atau *filter size* digunakan untuk menentukan jumlah dimensi kernel atau kelokalan pada saat konvolusi. Jika ukuran kernel sama dengan input

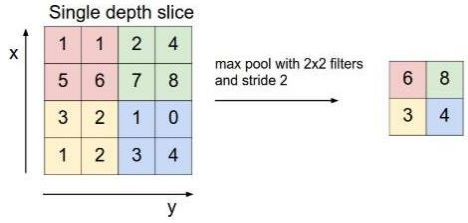
maka bisa disebut *fully connected layer. Stride* adalah parameter yang menentukan jumlah perpindahan kernel. Jika nilai dari *stride* adalah 1, maka *convolution filter* akan berpindah sejauh 1 pixel baik secara horizontal maupun vertical (Peryanto dkk., 2020).




Gambar 2. 5 Operasi Konvulasi

Seperti yang ditunjukkan dalam gambar 2.5, gambar input memiliki dimensi 5 piksel lebar, 5 piksel tinggi, dan 3 channel RGB. *Convulation laye*r akan melihat input secara lokal, seperti 3x3 pixel berwarna kuning. Pergerakan kernel akan bergeser dari posisi kiri atas ke sudut kanan bawah. Dengan demikian, hasil konvolusi dari gambar dapat terlihat dari gambar 2.5 di sisi kanan. (Peryanto dkk., 2020).

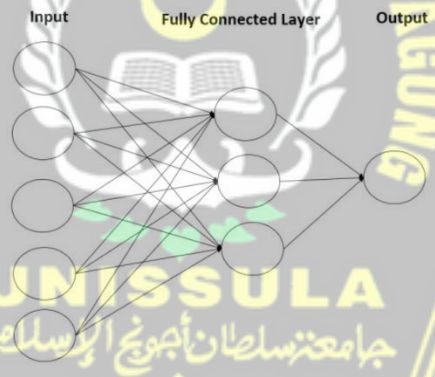
*Pooling layer* atau *sub-sampling layer* merupakan suatu layer yang berperan dalam mengurangi ukuran dimensi dari *feature map* yang dihasilkan *convolutional layer*. Sehingga layer berikutnya tidak memerlukan komputasi yang terlalu banyak. Pada dasarnya pooling layer terdiri dari filter dengan dimensi dan pergeseran tertentu yang bergerak melintasi seluruh area feature map. Jenis pooling yang biasanya sering digunakan yait Max Pooling dan Average Pooling (Peryanto dkk., 2020). *Max pooling* memilih nilai maksimum dari setiap area input lokal yang dilihat. Dengan *max pooling*, kita hanya meneruskan bagian yang paling penting dari input. *Average pooling* akan merata-rata semua nilai dari area lokal tersebut sehingga akan meneruskan semua bagian dari input sama rata.




Gambar 2. 6 Operasi Max Pooling

*Pooling layer* umumnya menggunakan filter yang berukuran 2x2, digunakan dengan pergeseran sebesar 2. Kemudian diterapkan pada setiap bagian irisan input. Pola ini mengakibatkan pengurangan hingga 75% pada ukuran *feature map* asli. (Peryanto dkk., 2020).

Aktivasi ReLU (*Rectified Linear Unit*) adalah layer aktivasi yang digunakan dalam model CNN. Lapisan ini menghasilkan efek *thresholding* dengan nilai nol terhadap piksel-piksel pada citra input. Sehingga dengan menggunakan aktivasi ReLU, setiap nilai piksel yang bernilai negative dalam citra akan diproses menjadi nilai 0.




Gambar 2. 7 fully connected layer

*Fully Connected Layer* adalah layer yang umumnya dipakai dalam model *Multilayer Perceptron* (MLP) dan lapisan ini bertugas untuk mengubah dimensi data menjadi jumlah *neuron* atau unit yang terdapat dalam *layer* tersebut, sehingga data dapat dianalisis secara linear. Sebelum dimasukkan ke dalam fully connected layer, setiap neuron pada convolution layer harus diubah menjadi data satu dimensi terlebih dahulu. Karena langkah ini mengakibatkan hilangnya informasi spasial yang tidak dapat dipulihkan, lapisan fully connected hanya dapat ditempatkan di akhir jaringan untuk mempertahankan integritas informasi spasial pada tahap awal (Ilahiyah dan Nilogiri, 2018).

oftmax berfungsi untuk menghitung probabilitas dari setiap kelas target di antara seluruh kemungkinan kelas target. Tujuannya adalah untuk membantu menentukan kelas target yang paling mungkin untuk suatu input yang diberikan (Geraldy dan Lubis, 2020). Berikut adalah rumus yang digunakan untuk menghitung *softmax*:

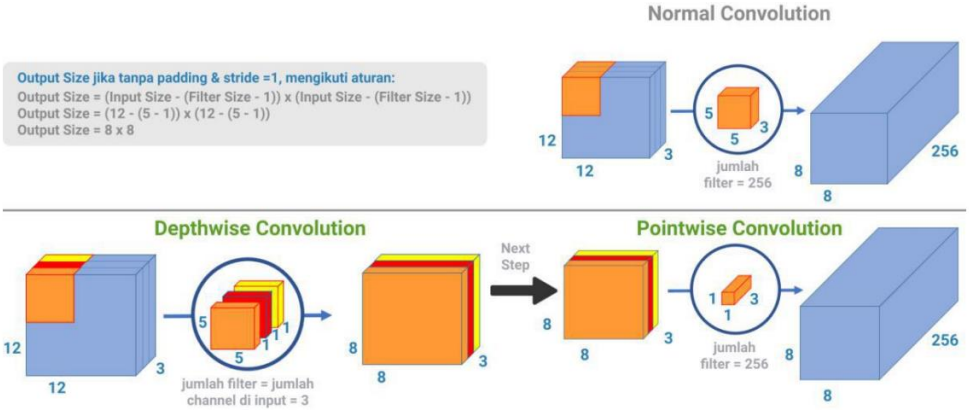
$$f_j(z) = \frac{e^z j}{\sum_k e^z k} \quad (1)$$

Notasi $f_j$ mengindikasikan hasil fungsi untuk setiap elemen ke-j pada output vektor kelas. Argumen z adalah hipotesis yang dihasilkan oleh model pelatihan yang akan diklasifikasikan oleh fungsi Softmax. Softmax memberikan hasil yang lebih intuitif dan memungkinkan interpretasi probabilistik yang lebih baik dibandingkan dengan algoritma klasifikasi lainnya. Fungsi ini memungkinkan perhitungan probabilitas untuk setiap label, mengambil vektor nilai riil dari label yang ada, dan merubahnya menjadi vektor dengan nilai antara nol dan satu. Total nilai dalam vektor tersebut akan selalu setara dengan satu (Ilahiyah dan Nilogiri, 2018).

### 2.2.3 *MobileNetV2*

*MobileNetV2* adalah arsitektur CNN yang dirancang untuk tugas klasifikasi gambar dan deteksi objek. Dikembangkan oleh Google sebagai penyempurnaan dari pendahulunya *MobileNetV1*. *MobileNetV2* memiliki keunggulan dalam efisiensi komputasi, ringan, dan dirancang khusus untuk perangkat dengan sumber daya terbatas, seperti perangkat *mobile* (Annur dkk., 2023). Arsitektur ini mempunyai layer khusus yang dikenal sebagai depthwise separable convolution yang terdiri dari dua tahap, yaitu depthwise convolution dan pointwise convolution. Fungsi layer ini adalah untuk mengurangi komputasi dengan mengurangi jumlah parameter, sehingga menghasilkan model yang lebih kecil dalam ukuran.

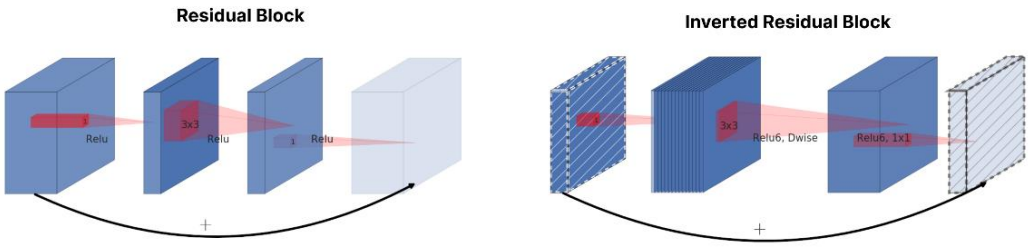
Berikut ilustrasi perbandingannya antara *normal convolution* dan *depthwise convolution* pada gambar 2.8. Gambar yang dimasukkan berdimensi 12x12x3 dan 256 kernel yang berukuran 5x5x3.




Gambar 2. 8 Perbandingan normal convolution dan depthwise separable convolution

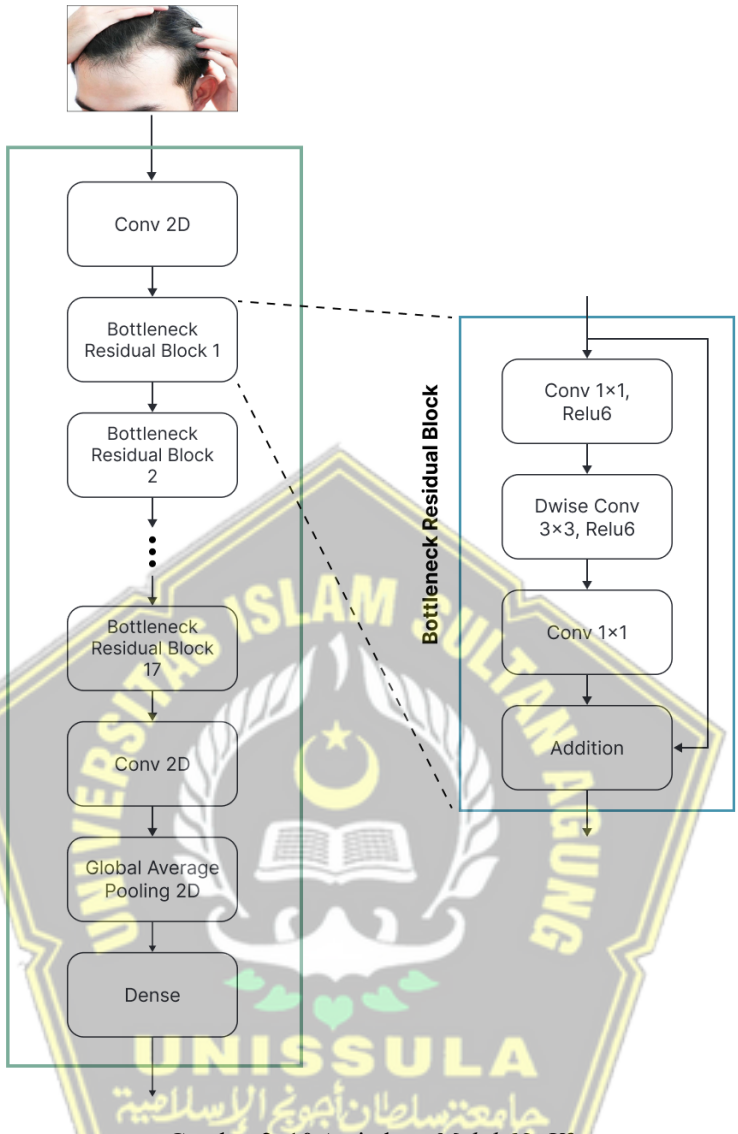
Dalam normal convolution, filter berinteraksi dengan seluruh channel input gambar dan menghasilkan feature map. Namun dalam depthwise convolution, setiap filter hanya berinteraksi dengan satu channel pada input gambar, menghasilkan peta fitur dengan jumlah channel yang sama dengan jumlah filter. Setelah itu, untuk mencapai output yang sama dengan konvolusi normal, dilakukan pointwise convolution dengan filter 1x1. Filter ini memiliki kedalaman yang sama dengan jumlah channel pada peta fitur sebelumnya.

Dalam perbandingan jumlah komputasi antara normal convolution dan depthwise separable convolution pada gambar 2.10, dengan 256 filter berukuran 5x5x3 dan pergeseran sebanyak 8x8 kali, total perkalian selama proses konvolusi adalah 1.228.800. Namun, untuk konvolusi separabel dalam kedalaman, tahap konvolusi dalam kedalaman menggunakan 3 filter berukuran 5x5x1 dengan pergeseran yang sama menghasilkan hanya 4.800 perkalian. Kemudian, pada tahap konvolusi titik, dengan 256 filter berukuran 1x1x3 dan pergeseran yang sama, menambahkan 49.152 perkalian. Keseluruhan, jumlah perkalian pada *depthwise separable convolution* adalah 52.952, yang setara dengan sekitar 22,7% dari jumlah perkalian pada normal *convolution*. Oleh karena itu, *depthwise separable convolution* membuktikan efisiensi komputasi yang jauh lebih baik dibandingkan dengan normal *convolution* (Raihan dkk., 2021).




Gambar 2. 9 Perbedaan Residual Block dan Inverted Residual Block

Dalam *MobileNetV2* terdapat penambahan dua ide utama yaitu *inverted residual* dan *linear bottleneck*. *Linear bottleneck* adalah desain *residual block* di mana *output* dari konvolusi terakhir dalam *residual block* dapat dihubungkan ke *input* awal menggunakan *shortcut connection*. *Inverted residual block* sendiri sebenarnya mirip dengan *residual block*, seperti terlihat pada gambar 2.11. Perbedaan dari *inverted residual block* dalam *MobileNetV2* adalah bahwa ia menerapkan pendekatan sempit → lebar → sempit mengenai jumlah channel, ini seperti membalikkan urutan operasi dalam *residual block* konvensional. Dalam *inverted residual block*, langkah pertama adalah memperluas jaringan menggunakan *convulation* 1x1. Kemudian dilakukan *depthwise convolution* 3x3, diikuti oleh fungsi aktivasi ReLU. Pada akhirnya jumlah *channel* ditingkatkan kembali dengan menggunakan *convulation* 1x1 lainnya. (Sandler dkk., 2018)




Gambar 2. 10 Arsitektur *MobileNetV2*

Dengan menggabungkan konsep *inverted residual* dan *linear bottleneck* menghasilkan arsitektur *MobileNetV2* yang terdiri dari serangkaian layer yang dapat dilihat pada gambar 2.12. Pada proses pertama, citra akan dimasukkan menggunakan layer *convulation* 2D, proses selanjutnya pada layer *bottleneck residual block*. Disinilah konsep *linear bottleneck* dan *inverted residual block* diterapkan. Setelah melewati *bottleneck residual block*, akan masuk pada layer *Global Average Pooling*(GAP) agar menghasilkan *output* model tensor 2D. Output model tensor 2D tersebut lalu akan masuk ke dalam layer *Dense*. Fungsi dari layer *Dense* adalah untuk mengubah matriks 2D tersebut menjadi representasi yang lebih

padat dan kompleks, sehingga model dapat mempelajari pola-pola yang lebih detail dari data yang diberikan.

### 2.2.4 *Tensorflow*

Gambar 2. 11 Tensorflow

*TensorFlow* merupakan suatu kerangka kerja *open-source* yang dikembangkan oleh Google untuk keperluan komputasi numerik dan *Artificial Intelligence* (AI). *TensorFlow* menyediakan alat dan sumber daya untuk membangun dan melatih model *neural networks*, yang digunakan untuk berbagai tugas, termasuk pengenalan gambar, pengenalan suara, dan pemrosesan bahasa alami. (Baihaqi dkk., 2022).

Sedangkan *TensorFlow Lite* adalah versi ringan dari *TensorFlow* yang dirancang khusus untuk perangkat bergerak, seperti ponsel pintar (smartphone) dan perangkat *Internet of Things* (IoT). Tujuan dari *TensorFlow Lite* adalah memberikan kemampuan pengenalan dan inferensi model *machine learning* dengan ukuran dan penggunaan daya yang lebih efisien di perangkat dengan sumber daya terbatas (Baihaqi dkk., 2022). Dalam konteks pengembangan aplikasi *mobile*, *TensorFlow Lite* menjadi pilihan yang baik karena dapat diintegrasikan dengan mudah pada perangkat Android dan iOS.

### 2.2.5 Android

Android merupakan sebuah sistem operasi perangkat *mobile* berbasis Linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Dirancang sebagai platform yang *open-soutce*, android memberikan fleksibilitas kepada para pengembang untuk menciptakan dan mengembangkan berbagai aplikasi. Sistem operasi ini menawarkan lingkungan yang mendukung aplikasi yang beragam, mulai dari permainan hingga aplikasi pembelajaran seperti yang dijelaskan dalam penelitian di atas (Azis dkk., 2020).

Gambar 2. 12 Kotlin

Kotlin adalah bahasa pemrograman yang dikembangkan oleh *JetBrains*, yang juga merupakan perusahaan yang mengembangkan IDE Android Studio. Bahasa ini dirancang untuk menjadi lebih ekspresif, ringkas, dan aman dibandingkan dengan bahasa pemrograman Java. Kotlin berjalan di atas *Java Virtual Machine* (JVM) dan mendukung interoperabilitas dengan Java, sehingga memungkinkan pengembang untuk menggunakan keduanya dalam satu proyek. Kelebihan Kotlin meliputi penanganan yang lebih baik terhadap *NullPointerException*, penulisan kode yang lebih ringkas dan mudah dibaca, serta dukungan penuh dari IDE (Sibarani dkk., 2018). Selain itu, Kotlin didukung secara resmi oleh Google untuk pengembangan aplikasi Android, sejak diumumkan pada Google I/O 2017, memperkuat posisinya sebagai bahasa pemrograman yang sangat relevan dan didukung untuk pengembangan aplikasi Android.

### 2.2.6 *Flowchart*

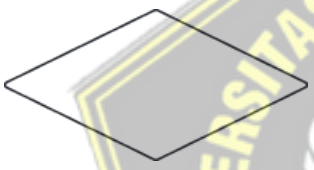
*Flowchart* adalah gambaran diagram alir dari algoritma dalam suatu program. Dengan menggunakan *flowchart*, penyelesaian suatu masalah khususnya yang memerlukan pemahaman dan evaluasi lebih lanjut dapat lebih mudah. Untuk membuat *flowchart* secara efektif, penting untuk memperhatikan beberapa pedoman berikut:

Simbol-simbol *flowchart* yang biasanya digunakan menjelaskan secara rinci berbagai aspek dalam pembuatan diagram alir. Diagram alir biasanya dibuat mengikuti prosesnya dari atas ke bawah dan dari kiri ke kanan. Aktivitas yang digambarkan harus dirinci dengan cermat dan mudah dimengerti oleh pembaca. Setiap aktivitas perlu memiliki penentuan waktu awal dan akhir yang jelas. Dalam menggambarkan setiap langkah aktivitas, harus dijelaskan menggunakan kata kerja. Penting juga untuk menyusun urutan langkah-langkah dari setiap aktivitas secara benar. Selain itu, lingkup dan rentang dari aktivitas yang digambarkan harus diperiksa dengan cermat. Percabangan yang memotong aktivitas yang sedang diilustrasikan tidak selalu perlu dimasukkan dalam flowchart yang sama. Penggunaan simbol penghubung dan pengaturan percabangan yang tepat dapat membantu mengelola percabangan tersebut. Percabangan yang tidak relevan dengan sistem dapat dihilangkan atau ditempatkan di halaman terpisah. Dengan mengaplikasikan simbol-simbol flowchart standar, alur logika program dapat diilustrasikan secara efektif (Zalukhu dkk., 2023).

Tabel 2.1 Akan menjelaskan simbol-simbol *flowchart* yang biasanya digunakan:

Tabel 2. 1 Simbol-simbol *Flowchart*

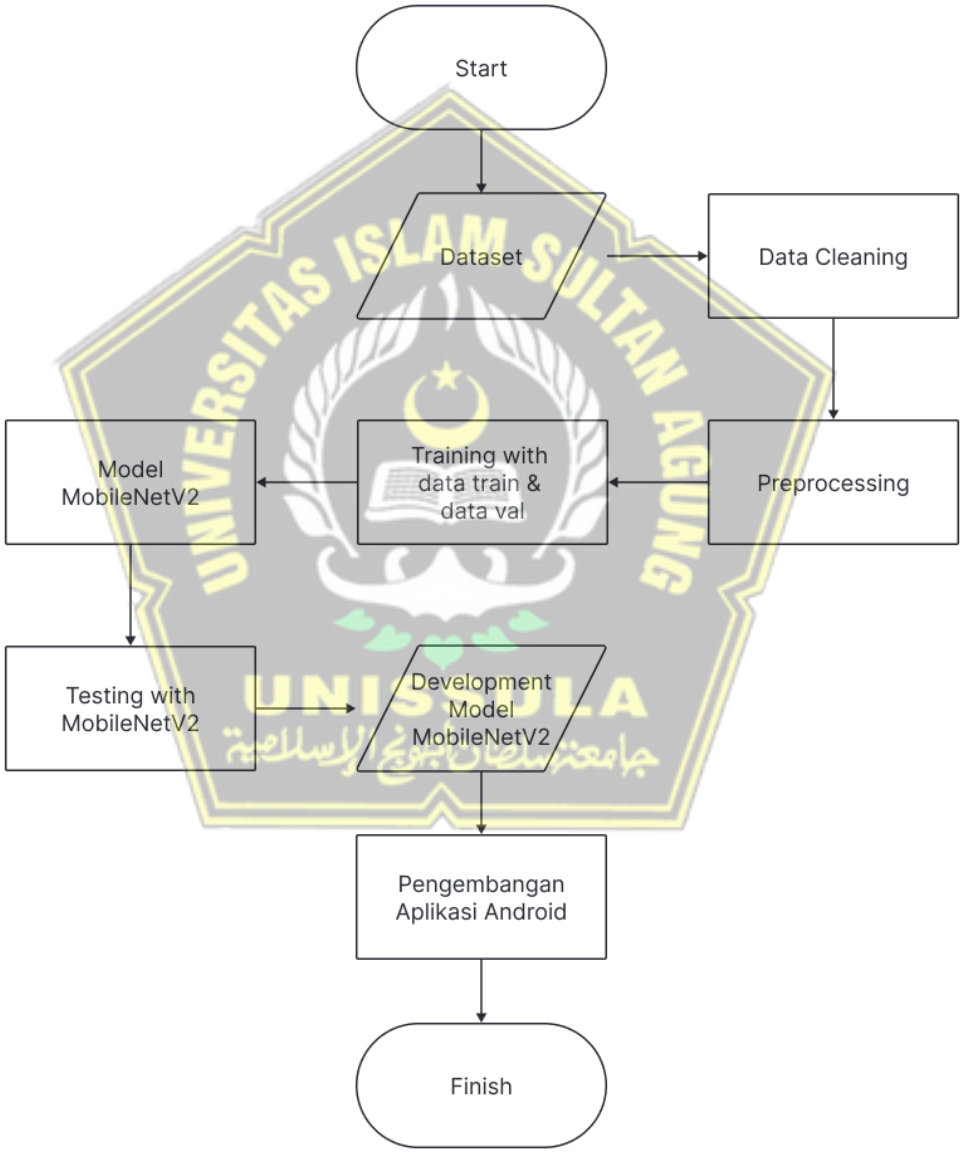
| Simbol | Fungsi |
|---|---|
|  | ***Terminator Symbol***<br>Permulaan atau akhir dari suatu program. |
| | |

| | |
|---|---|
|  | ***Processing Symbol***<br>Proses yang dilakukan oleh sistem. |
|  | ***Input-Output***<br>Menunjukkan proses *input* dan *output*, terlepas dari jenis peralatan yang digunakan. |
|  | ***Predefine Proses***<br>Simbol untuk menjalankan sub-program atau prosedur. |
|  | ***Decision***<br>Simbol yang menandakan suatu keadaan tertentu, di mana menghasilkan dua pilihan. |
|  | ***Connector Symbol***<br>Keluar-masuk atau menghubungkan proses di dalam lembar / halaman yang sama. |
|  | ***Flow***<br>Juga disebut dengan *connecting line*, untuk menghubungkan satu simbol dengan simbol lainnya. |

# BAB III
# METODE PENELITIAN

## 3.1 Meode Penelitian

Metodologi penelitian pada penelitian ini, memiliki beberapa tahapan yang perlu dilakukan. Alur metodologi penelitian dapat dilihat pada gambar 3.1.




Gambar 3. 1 Flowchart Metode Penelitian

### 3.1.1 Dataset

Dataset penyakit kulit kepala yang digunakan berasal dari *kaggle* yang dapat diakses melalui tautan (https://www.kaggle.com/datasets/sundarannamalai/hair-diseases/data). Jumlah data citra adalah 12000 dengan 10 *class* jenis panyakit kulit kepala. Dari dataset ini, dipilih 3 jenis penyakit kulit kepala yaitu *Alopecia Areata, Folliculitis, Head Lice.* Masing masing class memiliki 1200 gambar, sehingga jumlah dataset saat ini adalah 3600. Gambar 3.2 memperlihatkan dataset yang akan digunakan dalam penelitian ini.


Gambar 3. 2 Data Gambar Penyakit Kulit Kepala

### 3.1.2 Data *Cleaning*

Pada tahapan ini, dilakukan pembersihan citra yang mengidentifikasikan duplikasi, citra yang tidak valid, dan citra yang tidak akurat. Dataset ini mengandung banyak citra duplikasi, kondisi ini dapat menyebabkan model mengalami overfitting. Oleh karena itu diperlukan penghapusan citra yang merupakan duplikat.

### 3.1.3 *Preprocessing*

Tahap *Preprocessing* dalam penelitian ini melibatkan tiga proses utama, yaitu *resize* data dan data *augmentation*.

1) *Resize Data*

   *Resize data* adalah proses di mana gambar yang ada diubah ukurannya untuk memenuhi persyaratan *input* model. Dalam penelitian ini, semua gambar dalam dataset disesuaikan ke ukuran 224 piksel x 224 piksel sebelum diberikan ke model.

2) *Normalization*

*Normalization* atau normalisasi dilakukan dengan tujuan untuk mengubah nilai fitur ke dalam skala yang seragam.. Dalam penelitian ini, normalisasi dilakukan menggunakan metode *rescaling citra*. Rescale memiliki tujuan untuk mengubah ukuran citra dengan melakukan pembagian nilai piksel terkecil dengan nilai piksel terbesar. Penerapan dalam penelitian ini adalah melakukan transformasi pada ukuran piksel gambar RGB dari rentang nilai 0 hingga 255 sehingga nilai terkecil menjadi 0 dan nilai terbesar menjadi 1.

3) *Data Augmentation*

Dengan melakukan data cleaning, kemungkinan besar jumlah dataset akan berkurang secara signifikan. Sehingga hal ini dapat menyebabkan risiko *overfitting* pada model. *Overfitting* terjadi ketika model mampu memprediksi data *training* dengan sangat akurat, tetapi kinerjanya menurun secara signifikan saat diujikan dengan data testing yang belum pernah dilihat sebelumnya oleh model. Hal ini dapat menghambat kemampuan model untuk melakukan generalisasi dengan efektif.

Karena itu, perlu dilakukan augmentasi data untuk memperluas dataset yang terbatas dan mengurangi risiko udaerfitting/overfitting pada model. Augmentasi mengacu pada proses menciptakan variasi baru dari dataset gambar dengan melakukan transformasi atau manipulasi. Beberapa metode umum dalam melakukan augmentasi data gambar mencakup *rotation*, *cropping*, *translation*, *brightness*, *flip horizontal* atau *vertical*, dan lain-lain.

Dalam penelitian ini, metode augmentasi yang digunakan memanfaatkan beberapa paramenter pada *ImageDataGenerator* dari Keras, yakni:

1. Rotation range

   Rotation digunakan untuk memanipulasi gambar dengan melakukan memutar citra sebesar sudut yang ditentukan dalam rentang yang telah ditentukan. Rotation range yang diatur bernilai 15.

2. Zoom range

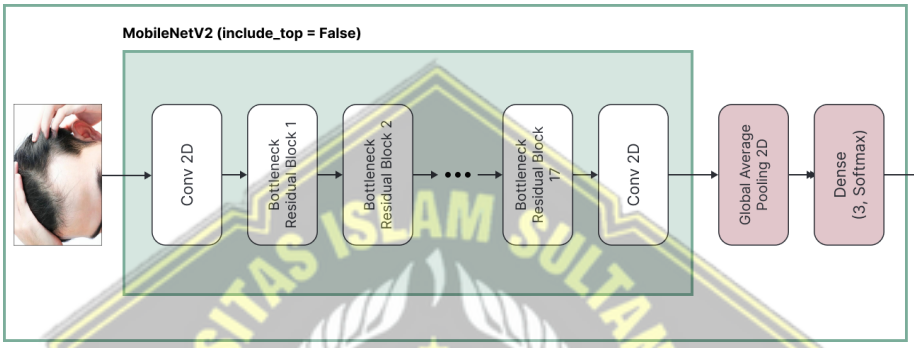
   Zoom range digunakan untuk memodifikasi gambar dengan mengatur besarnya atau kecilnya. Zoom range mengontrol seberapa jauh gambar

akan diperbesar atau diperkecil. Nilai yang digunakan dalam zoom range ini adalah 0.1.

3. Horizontal flip

   Dengan horizontal flip, gambar dapat dibalik secara horizontal sehingga sisi kiri dan kanan menjadi terbalik.

### 3.1.4 Training




Gambar 3. 3 Arsitektur MyModel

Dalam arsitektur *MobileNetV2* diperlukan beberapa perubahan pada lapisan *fully connected* untuk menyesuaikan keluaran. Karena penelitian ini fokus pada klasifikasi 3 kategori penyakit kulit kepala, lapisan *Dense* menggunakan fungsi aktivasi *softmax* dengan 3 node. Dengan menggunakan konfigurasi parameter *include_top = False* sehingga menghilangkan lapisan *fully connected* bawaan dari *MobileNetV2*. Kemudian diperlukan penambahan lapisan *Global AveragePooling* 2D dan lapisan *Dense* dengan 3 *output* dan aktivasi *softmax*. Model baru ini diberi nama mymodel, dengan detail arsitektur dapat dilihat pada gambar 3.3.

Dataset yang digunakan dalam tahap ini terdiri dari data training dan data validation. Langkah awal melibatkan model baru mymodel diberikan data training untuk melakukan proses pembelajaran sehingga mengenali pola pada gambar-gambar tersebut. Kemudian mengukur akurasi model menggunakan data validation. Jika akurasi masih belum tercapai, model akan terus disesuaikan parameter-parameternya dan di-train ulang dengan menggunakan data training. Proses ini dilakukan berulang kali hingga mencapai tingkat akurasi yang diinginkan. Beberapa konfigurasi yang akan digunakan dalam penelitian ini dijelaskan dalam tabel 3.1.

Tabel 3. 1 Konfigurasi Pada Model

| | ***Optimizer*** | ***Epoch*** | ***Learning Rate*** | ***Batch Size*** |
|---|---|---|---|---|
| Konfigurasi 1 | Adam | 10 | 0.001 | 10 |
| Konfigurasi 2 | Adam | 10 | 0.001 | 30 |
| Konfigurasi 3 | Adam | 10 | 0.0001 | 10 |
| Konfigurasi 4 | Adam | 10 | 0.0001 | 30 |

### 3.1.5 Testing

Untuk memastikan bahwa model memiliki kinerja yang baik, dilakukan testing model dengan menggunakan data testing, yang merupakan data yang belum pernah dilihat oleh model sebelumnya. Dalam penghitungan akurasi model, digunakan metode confusion matrix yang memberikan informasi tentang akurasi, presisi, recall, dan F1-Score. Berikut ini adalah penjelasan tentang rumus dari masing-masing fungsi yang akan digunakan dalam tahap testing ini:

**a. *Confusion matrix***

*Confusion matrix* adalah ialah sebuah tabel yang dipakai untuk menilai performa suatu model dalam klasifikasi atau deteksi. Evaluasi ini dilaksanakan dengan membandingkan hasil prediksi yang dihasilkan oleh model dengan nilai aktual dari data pengujian. Rincian tabel *confusion matrix* dapat dilihat pada tabel 3.1.

Tabel 3. 2 Confusion Matrix

| | **Predicted Positif** | **Predicted Negatif** |
|---|---|---|
| **Actually Positif** | TP | FN |
| **Actually Negatif** | FP | TN |

*Confusion Matrix* memiliki 4 komponen utama yang mencakup empat kemungkinan hasil dari prediksi:

1) *True Positive* (TP): Sampel positif yang benar-benar diklasifikasikan oleh model sebagai positif.
2) *True Negative* (TN): Sampel negatif yang benar-benar diklasifikasikan oleh model sebagai negatif

3) *False Positive* (FP): Sampel negatif yang keliru diprediksi oleh model sebagai positif.
4) *False Negative* (FN): Sampel positif yang keliru diprediksi oleh model sebagai negatif.

**b. *Accuracy***

*Accuracy* merupakan *matrix* evaluasi yang digunakan untuk menilai seberapa baik model dapat mengklasifikasikan data secara tepat pada seluruh data testing. Secara matematis, nilai akurasi dihitung dengan membagi jumlah prediksi yang benar dengan total data keseluruhan. Berikut rumus *accuracy*:

$$Accuracy = \frac{TP}{total\ data} \quad (2)$$

**c. *Precission***

*Precission* adalah *Matrix* yang mengukur seberapa banyak dari semua prediksi positif model yang sebenarnya benar positif adalah *precission*. Precision adalah ukuran yang memberikan informasi tentang seberapa tepat model dalam mengidentifikasi data sebagai positif. Semakin tinggi nilai precision, semakin sedikit hasil yang false positive yang dihasilkan oleh model. *Precission* dihitung dengan membagi total data *True Positive* dengan total prediksi positif (TP dan FP). Berikut rumus *Precission*:

$$Precission = \frac{TP}{TP + FP} \quad (3)$$

**d. *Recall***

Recall mengukur efektivitas model dalam mendeteksi kasus positif yang sebenarnya dengan benar. Dalam hal ini, semakin tinggi nilai recall, semakin rendah kemungkinan bahwa model akan melewatkan kasus true positif. Dalam konteks klasifikasi, recall dapat didefinisikan sebagai berikut:

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

**e. F1 - *Score***

F1-score adalah metrik evaluasi yang menggabungkan precision dan recall menjadi satu nilai tunggal. F1-Score berguna ketika ingin mencapai keseimbangan antara precission dan recall, terutama ketika dataset tidak seimbang. Secara matematis, F1-Score didefinisikan sebagai harmonic mean (rata-rata harmonik) dari precision dan recall:

$$F1 = 2 \times \frac{TP}{TP + FN} \quad (5)$$

### 3.2.1 Analisis Kebutuhan

Pada bagian analisis kebutuhan, sistem ini akan dipelajari secara mendalam untuk menentukan segala tindakan yang harus dilakukan selama proses masukan untuk menghasilkan keluaran berupa klasifikasi. Selain itu, ada beberapa proses dan fungsi yang harus diterapkan dalam sistem ini, antara lain:

1) Tips penggunaan aplikasi

   Pengguna perlu memahami cara menggunakan aplikasi dengan benar agar dapat menghindari kesalahan saat mengunggah gambar. Disarankan agar gambar yang diunggah hanya menampilkan bagian kulit kepala yang terkena penyakit. Sebisa mungkin, gambar seharusnya di-*crop* sehingga area penyakit kulit kepala berada di tengah gambar.

2) *Upload* gambar

   Dalam fitur *upload* gambar, terdapat dua opsi yang tersedia, yaitu meng-upload gambar dari galeri atau mengambil gambar langsung dari kamera.

3) *Cropping* gambar

   Dalam fitur *Cropping* gambar, digunakan untuk menghasilkan gambar yang siap untuk klasifikasi. Pengguna dapat melakukan pemotongan gambar sesuai dengan tips penggunaan aplikasi yang telah disampaikan sebelumnya.

4) Melakukan klasifikasi

   Klasifikasi merupakan fitur utama dalam aplikasi ini. Pengguna dapat menekan tombol "*Scan*" sehingga gambar yang sudah di-*upload* sebelumnya dapat

diklasifikasi. Sistem akan melakukan klasifikasi menggunakan model *mobileNetV2* yang di-*training* sebelumnya.

5) Menampilkan hasil klasifikasi

   Fungsi ini adalah *output* akhir dari aplikasi. Sistem akan menampilkan hasil klasifikasi gambar yang diberikan pengguna. Ketika tombol *scan* ditekan, output klasifikasi akan keluar dengan menampilkan keterangan jenis penyakit kulit kepala yang dideteksi.

### 3.1.6 Deployment Model MobileNetV2

Model CNN yang telah dilatih untuk klasifikasi penyakit kulit kepala, akan dikonversi menjadi format *TensorFlow Lite*, untuk dapat diintegrasikan ke dalam aplikasi perangkat android. Proses konversi model *MobileNetV2* menjadi *TensorFlow Lite* bertujuan untuk mengoptimalkan ukuran model agar dapat berjalan dengan cepat dan efisien pada perangkat *mobile*.

### 3.1.7 Pengembangan Aplikasi Android

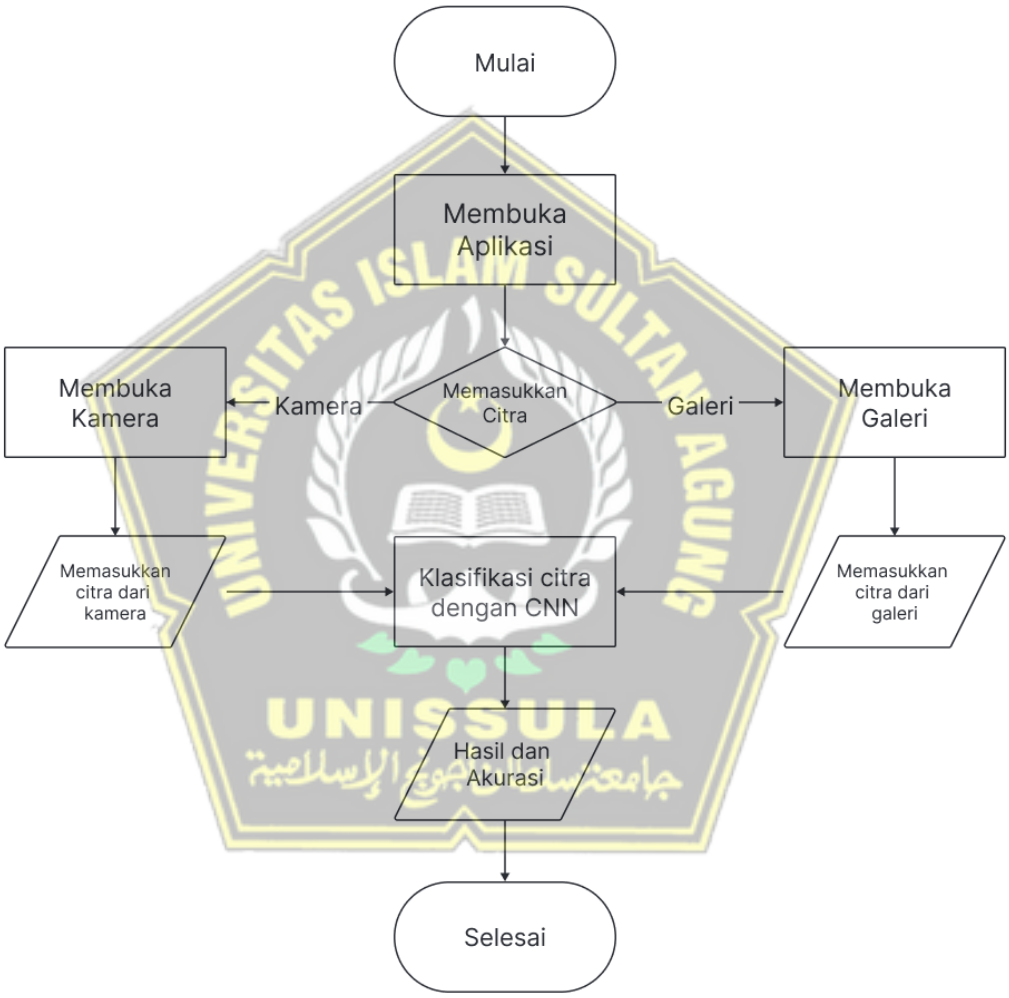
Pada tahap pengembangan aplikasi Android, Android Studio digunakan sebagai platform pengembangan utama. Proses pengembangan melibatkan integrasi model CNN dengan arsitektur *MobileNetV2* yang sudah berformat *TensorFlow Lite* ke dalam aplikasi.

Selanjutnya, dilakukan implementasi *interface* pengguna yang dirancang untuk memfasilitasi pengguna dalam mengambil gambar kulit kepala dan melihat hasil deteksi. Antarmuka ini memungkinkan pengguna berinteraksi dengan aplikasi secara intuitif dan menyediakan pengalaman yang mudah dipahami.

## 3.2 Perancangan Alur Sistem

### 3.2.2 Analisis Alur Sistem

Dalam analisis alur sistem ini, terdapat *flowchart* yang menjelaskan langkah-langkah urutan operasi dari sistem penelitian ini. Mulai dari proses input data hingga *output* berupa klasifikasi yang dihasilkan sistem. *Flowchart* ini dapat dilihat pada gambar 3.4.




Gambar 3. 4 Flowchart Sistem

Analisis sistem *flowchart* aplikasi dimulai dengan langkah pengguna membuka aplikasi. Pengguna akan diarahkan ke halaman *home*. Terdapat tombol "*Upload*" untuk menambahkan gambar, Sedangkan tombol "*Delete*" digunakan untuk menghapus gambar. Ketika pengguna menekan tombol *Upload* terdapat pilihan "Galeri" dan "Kamera". Pengguna dapat memilih tombol galeri untuk

menambahkan gambar dari galeri. Atau pengguna juga dapat memilih tombol kamera untuk mengambil gambar dengan kamera *smartphone*. Setelah pengguna memasukkan gambar, dilakukan proses *cropping* untuk menyesuaikan area yang ingin dilakukan deteksi, setelah dirasa cukup pengguna dapat menekan tombol ceklis. Kemudian menekan tombol "Scan" untuk proses deteksi penyakit kulit kepala. Selesai proses deteksi, model CNN memberikan output berupa klasifikasi jenis penyakit kulit kepala. Hasil deteksi tersebut kemudian ditampilkan pada antarmuka pengguna berupa hasil klasifikasi.

### 3.2.3 Analisis Sistem

Pada tahap analisis sistem, kebutuhan berbagai alat dan perangkat yang diperlukan untuk pengembangan sistem ini akan diidentifikasi. Informasi terperinci mengenai alat dan perangkat yang akan digunakan tersedia dalam tabel 3.1.

Tabel 3. 3 Analisis Sistem

| No | Tool | Version |
|---|---|---|
| 1 | Laptop Acer Aspire 5 A514-54G-524Q | Windows 11 |
| 2 | Smartphone Samsung Galaxy A32 | Android 13 |
| 4 | Google Collab | - |
| 6 | Android Studio | Giraffe, 2022.3.1 |
| 7 | Kotlin | 1.9.0 |
| 8 | Gradle | 8.0 |
| 9 | SDK | 34 |

**a.** Python

Bahasa pemrograman Python dipilih untuk membuat model *machine learning* karena merupakan bahasa tingkat tinggi yang mudah dibaca dan dipahami. Kode Python dieksekusi baris per baris oleh interpreter tanpa adanya kompilasi terlebih dahulu. Dukungan dari berbagai *library* yang kuat untuk ma*chine learning*, menjadikan Python sebagai pilihan utama untuk pengembangan model *machine learning*.

**b.** Tensorflow

TensorFlow merupakan *library open-source* yang populer digunakan dalam pelatihan model *machine learning* yang dikembangkan oleh Google. Fungsi-fungsi pada TensorFlow dapat digunakan untuk mengelola data, melatih model, mengevaluasi performa model serta melakukan inferensi pada data baru.

**c.** Keras

Keras merupakan sebuah perpustakaan *library deep learning open-source* yang dirancang untuk mempermudah pembuatan model jaringan saraf tiruan, termasuk dalam penelitian ini adalah CNN.

**d.** Tensorflow Lite

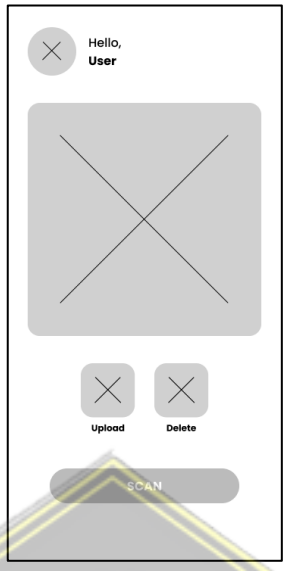
TensorFlow Lite merupakan versi TensorFlow yang lebih ringan, dirancang secara khusus untuk perangkat *mobile*. TensorFlow Lite mengubah model menjadi ukuran yang lebih kecil melalui beberapa teknik optimasi dan kompresi. Tujuannya agar memungkinkan diimplementasikan pada berbagai platform edge termasuk Android dengan performa optimal.

## 3.3 Perancagan User Interface

Dalam proses perancangan antarmuka pengguna, dilakukan desain *mockup* yang akan diimplementasikan ke dalam sistem. Berikut adalah tampilan antarmuka pengguna yang telah dirancang untuk sistem "Deteksi Penyakit kulit kepala Menggunakan *MobileNetV2* Berbasis Android".

### 3.3.1 Halaman Home

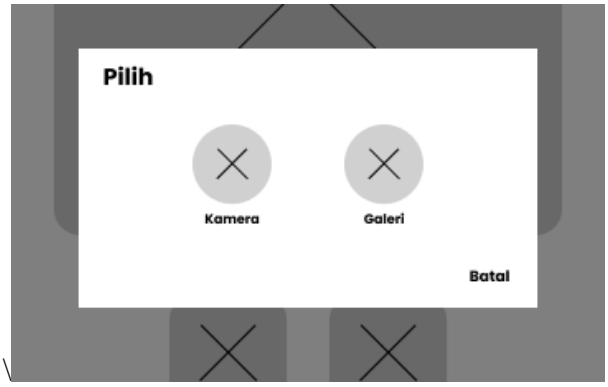
Halaman utama merupakan tampilan pertama dalam aplikasi ini. Pada halaman ini terdapata *frame* untuk gambar klasifikasi. Ketika belum mengupload gambar, teradapat keterangan "*No Image*". Kemudian terdapat tombol *upload* untuk memasukkan gambar pada aplikasi, delete untuk menghapus *image* pada *frame*, serta tombol *scan* untuk memulai proses klasifikasi. Sedain tampilan halama *Home* dapat dilihat pada gambar 3.5.



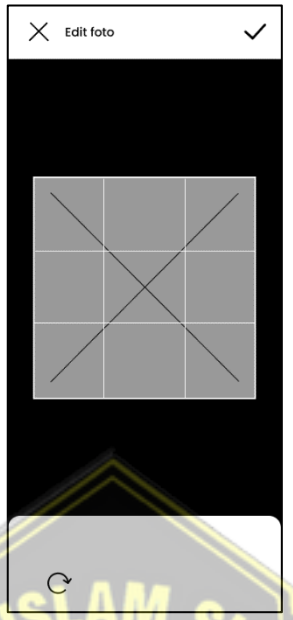

Gambar 3. 5 Halaman Home

### 3.3.2 Upload Gambar

Ketika tombol *upload* ditekan, akan ada *pop up* tampilan pilihan galeri dan kamera. Pengguna dapat memilih galeri ketika ingin meng-*upload* gambar dari galeri seperti terlihar pada gambar 3.6. Dan pengguna dapat memilih kamera untuk mengambil gambar melalui kamera *smartphone* mereka. Setelah pengguna memilih gambar, langkah selanjutnya adalah meng-*crop* gambar. Pastikan area penyakit yang akan diklasifikasi berada di tengah gambar. Tampilan fitur *cropping* dapat dilihat pada gambar 3.7. Setelah dirasa sudah sesuai, pengguna dapat menekan tombol ceklis di kanan atas. Lalu pengguna akan dipindahkan ke tampilan Home dengan gambar yang sudah ada di *frame*.



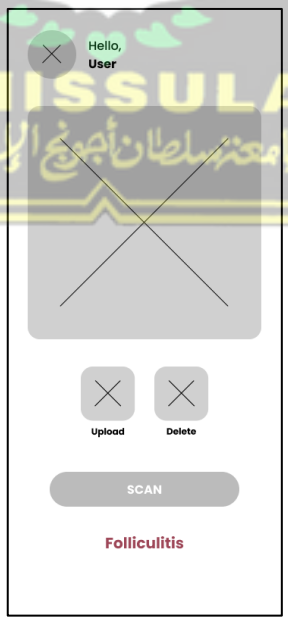

Gambar 3. 6 Dialog Box Tombol Upload




*Gambar 3. 7 Meng-cropping Gambar*

### 3.3.3 Output Klasifikasi

Untuk memunculkan hasil klasifikasi, pengguna perlu menekan tombol *scan*. Sistem akan memulai proses klasifikasi dan akan menampilkannya di bawah halaman *home*. *Mockup* bisa dilihat pada gambar 3.8.



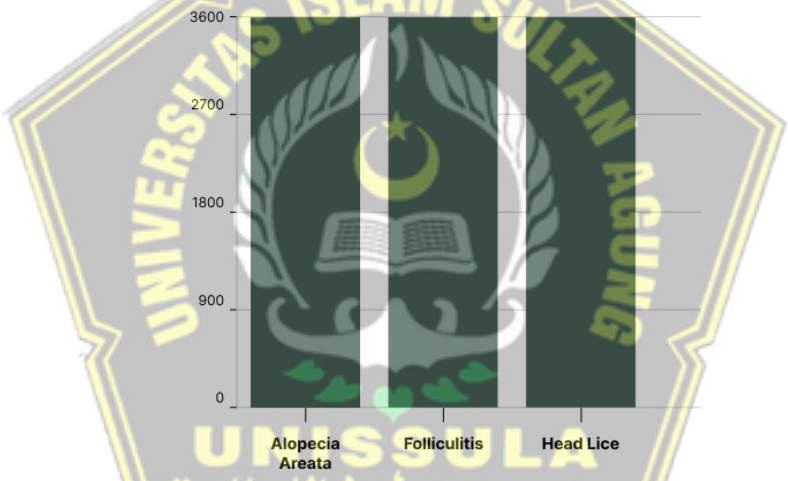

Gambar 3. 8 Output Klasifikasi

# BAB IV
# HASIL DAN ANALISIS PENELITIAN

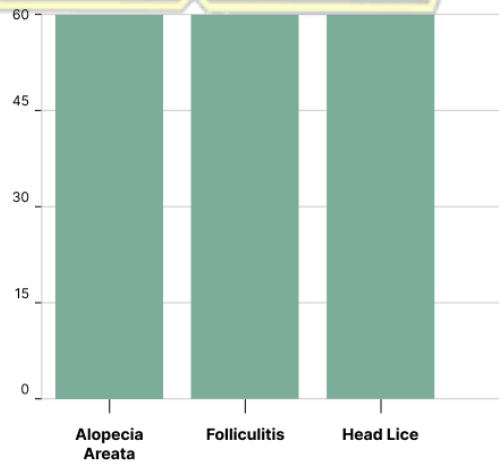
## 4.1 Proses Pengolahan Training dan Validation

### 4.1.1 Data Cleaning

Proses ini bertujuan untuk membersihkan dataset citra dengan mengidentifikasi dan menghapus data duplikasi, data yang tidak valid, dan data yang tidak akurat. Gambar 4.1 menampilkan dataset sebelum dilakukan *data cleaning*. Setelah melalu proses *data cleaning* jumlah dataset akhir berjumlah 180, dengan masing-masing *class* memiliki 60 citra, pembagian akhir dataset ini dapat dilihat pada gambar 4.2.

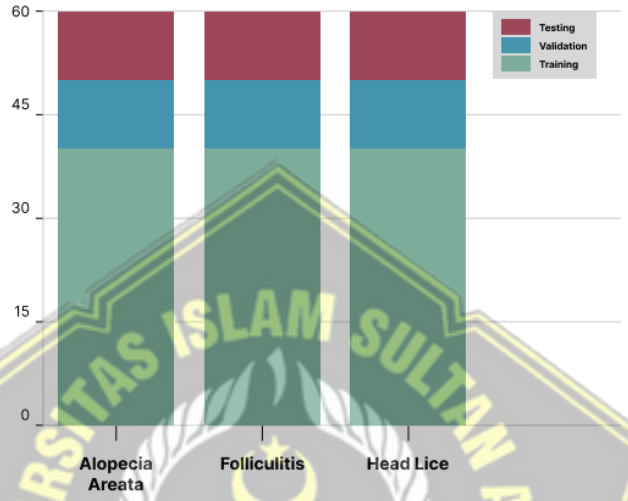



Gambar 4. 1 Jumlah Dataset Sebelum Proses Data Cleaning




Gambar 4. 2 Jumlah Dataset Setelah Proses Data Cleaning

Dengan dataset yang telah melalui proses *data cleaning*, sekarang dataset tersebut dapat dibagi menjadi *data training*, *data validation*, dan *data testing*. *Data training* terdiri dari 40 citra, *data validation* terdiri dari 10 citra, dan *data testing* terdiri dari 10 citra. Pembagian ini dapat diilustrasikan seperti yang terlihat pada Gambar 4.3.




Gambar 4. 3 Dataset Splitting

### *4.1.2 Preprocessing*

*Preprocessing* bertujuan untuk mempersiapkan data sehingga data tersebut siap untuk digunakan dalam proses *training* dalam pembangunan model. Dalam penelitian ini *preprocessing* dilakukan dengan 3 proses, yaitu:

1) *Resize Data*

   Resizing dilakukan untuk mengubah ukuran citra menjadi 224 piksel x 224 piksel. Ukuran ini umum digunakan dalam berbagai arsitektur CNN.

2) *Normalization*

   *Normalization* dilakukan untuk menstandarisasi rentang nilai dari fitur memiliki skala yang seragam. Dalam penelitian ini normalisasi dilakukan dengan metode *rescaling*.

3) *Data Augmentasi*

   Augmentasi dilakukan untuk menghasilkan variasi baru dari dataset gambar dengan melakukan transformasi atau manipulasi. Data yang akan diaugmentasi

adalah *data training*. Gambar 4.4 menjelaskan jumlah *data training* sebelum dilakukan proses augmentasi. Masing-masing class memiliki 40 gambar. Setelah dilakukan proses augmentasi dengan metode *rotation range* bernilai 15, *zoom range* 0.1, dan *horizontal flip,* jumlah *data train* hasil augmentasi menjadi 240. Dengan masing-masing class memiliki 80 gambar. Gambar 4.5 menjelaskan detail jumlah *data training* setelah proses augmentasi.

```
Jumlah file alopecia dalam folder: 40
Jumlah file folliculitis dalam folder: 40
Jumlah file headlice dalam folder: 40
```

Gambar 4. 4 Data *Training* Sebelum Proses Augmentasi

```
Jumlah file alopecia dalam folder: 80
Jumlah file folliculitis dalam folder: 80
Jumlah file headlice dalam folder: 80
```

Gambar 4. 5 Data *Training* Setelah Proses Augmentasi

## 4.2 Proses Penggunaan dan Cara Kerja Sistem

Pada bagian ini akan dijelaskan tentang proses pengklasifikasian gambar yang diunggah oleh pengguna menggunakan model yang telah dibuat sebelumnya.

### 4.2.1 Upload Gambar

Pertama, pengguna meng-*upload* gambar penyakit kulit kepala yang akan diklasifikasi. Dalam sistem, terdapat dua pilihan untuk melakukan *upload* gambar: mengambil gambar dari kamera *smartphone* pengguna atau memilih gambar dari galeri pengguna Setelah pengguna memilih gambar, mereka dapat melakukan penyesuaian gambar dengan fitur *cropping*. Fitur ini penting agar gambar dapat diatur dengan baik sebelum proses klasifikasi, sehingga menghasilkan klasifikasi yang tepat.

### 4.2.2 Konversi Gambar ke TensorImage

Pada tahap ini sistem akan mengmbil gambar yang telah di-*upload* pengguna dan menyimpannya dalam bentuk *Bitmap*. Kemudian gambar akan diubah menjadi TensorImage. Proses konversi ini dilakukan agar gambar dapat disesuaikan dengan

ukuran yang diharapkan oleh model, yaitu me-*resize* gambar menjadi 224 piksel x 224 piksel.

### 4.2.3 Konversi TensorImage ke TensorBuffer

Selanjutnya sistem akan mengkonversi TensorImage menjadi TensorBuffer. TensorBuffer digunakan untuk menyimpan representasi gambar yang sudah diubah menjadi TensorImage. Kemudian TensorBuffer siap untuk proses klasifikasi oleh model

### 4.2.4 Proses Klasifikasi

Gambar yang sudah menjadi TensorBuffer akan diproses oleh model untuk dilakukan klasifikasi. Hasil klasifikasi merupakan probabilitas prediksi untuk setiap kelas yang telah ditentukan dalam model. Prediksi kelas dengan probabilitas tertinggi kemudian digunakan sebagai hasil akhir dari proses klasifikasi.

## 4.3 Implementasi User Interface

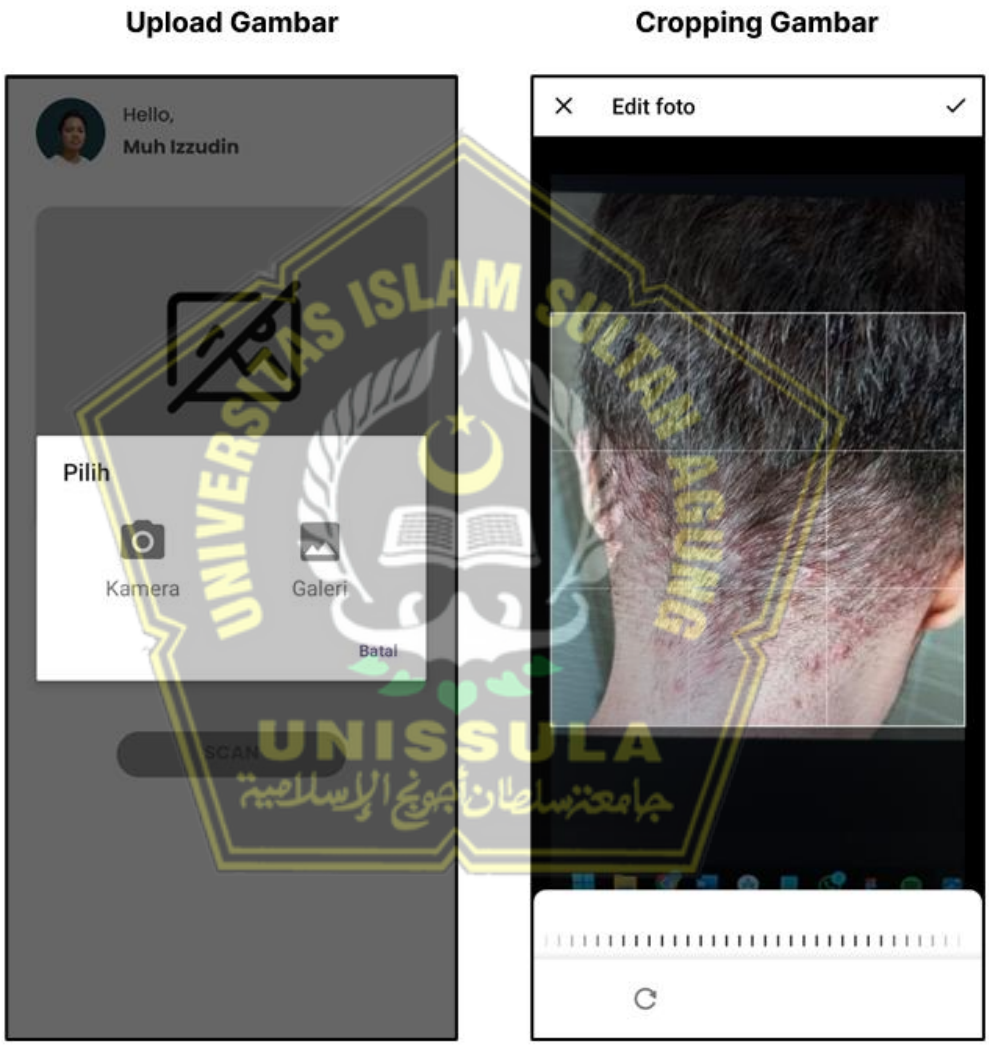
### 4.3.1 Halaman Home

Halaman *Home* merupakan halaman awal yang akan dilihat oleh pengguna pertama kali. Terdapat berbagai tombol dengan gambar dan tulisan untuk memudahkan interaksi pengguna dengan aplikasi. Tampilan home terlihat seperti pada gambar 4.6.


Gambar 4. 6 Tampilan Home

### 4.3.2 *Upload* Gambar

Untuk meng-*upload* gambar pengguna perlu menekan tombol *upload*, sehingga muncul pilihan kamera dan galeri. Ketika selesai memilih gambar, pengguna juga bisa menggunakan fitur *cropping* gambar. Tampilan ini bisa dilihat pada gambar 4.7.

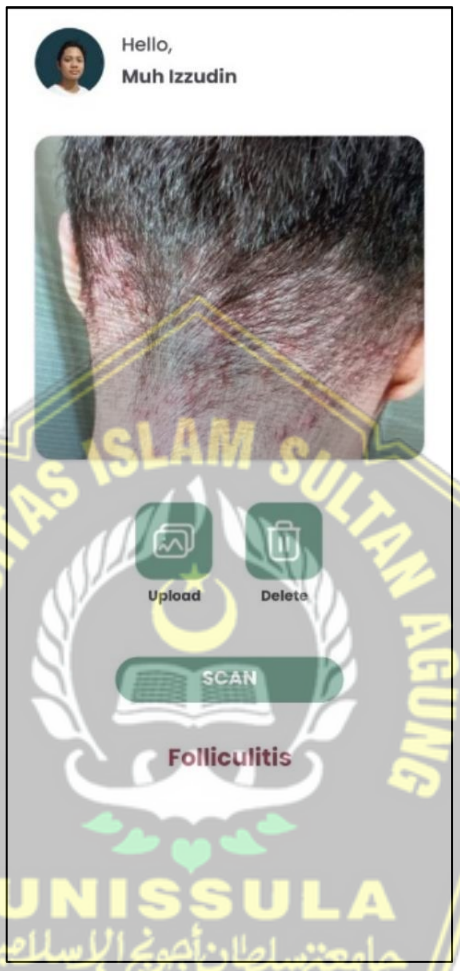



Gambar 4. 7 Tampilan Upload Gambar dan Cropping Gambar

### 4.3.3 Output Klasifikasi

Klasifikasi akan dimulai setelah pengguna meng-*upload* gambar dan menekan tombol *Scan*. Sistem akan memulai proses klasifikasi untuk mengidentifikasi jenis penyakit kulit kepala yang terdapat pada gambar yang

diunggah. Setelah proses klasifikasi selesai, halaman akan menampilkan hasil klasifikasi jenis penyakit kulit kepala di bawah tombol scan. Tampilan ini dapat dilihat pada gambar 4.8.




Gambar 4. 8 Tampilan Output Klasifikasi

## 4.4 Pengujian Sistem

Pada tahap pengujian sistem, digunakan metode *black box testing* yang merupakan pendekatan untuk menguji fungsionalitas sistem tanpa memperhatikan detail struktur atau logika internalnya. Metode ini bertujuan untuk memastikan bahwa sistem berfungsi sesuai dengan spesifikasi yang telah ditentukan. Dalam penelitian ini, *function testing* akan digunakan untuk menguji apakah setiap fungsi beroperasi dengan benar sesuai dengan yang diharapkan. Berikut tabel 4. yang merupakan hasil *function testing* pada sistem ini.

Tabel 4. 1 *Testing Black Box* pada Sistem

| **Skenario Pengujian** | **Kasus Pengujian** | **Hasil Pengujian** | **Kesimpulan** |
|---|---|---|---|
| Membuka aplikasi | Menampilkan halaman home | Sesuai | Normal |
| *Upload* gambar dari kemera | Melakukan *upload* gambar penyakit kulit kepala dari kamera | Sesuai | Normal |
| *Upload* gambar dari galeri | Melakukan *upload* gambar penyakit kulit kepala dari kamera | Sesuai | Normal |
| Menekan tombol *Scan* | Menekan tombol *Scan* untuk menampilkan hasil klasifikasi | Sesuai | Normal |
| *Testing* gambar *alopecia areata* | Melakukan *testing* pada gambar yang menunjukkan jenis penyakit kulit kepala *alopecia areata* | Sesuai | Normal |
| *Testing* gambar *Folliculitis* | Melakukan testing pada gambar yang menunjukkan jenis penyakit kulit kepala *folliculitis* | Sesuai | Normal |
| *Testing* gambar *Head Lice* | Melakukan testing pada gambar yang menunjukkan jenis penyakit kulit kepala *head lice* | Sesuai | Normal |

## 4.5 Hasil dan Analisis

### 4.4.1 *Training* dan *Validation*

Proses *training* dilakukan dengan *data training* yang diaugmentasi dan *data* validation. Data ini terdiri dari 240 citra untuk *data training* (80 citra per *class*) dan 30 citra untuk *data validation* (10 citra per *class*).

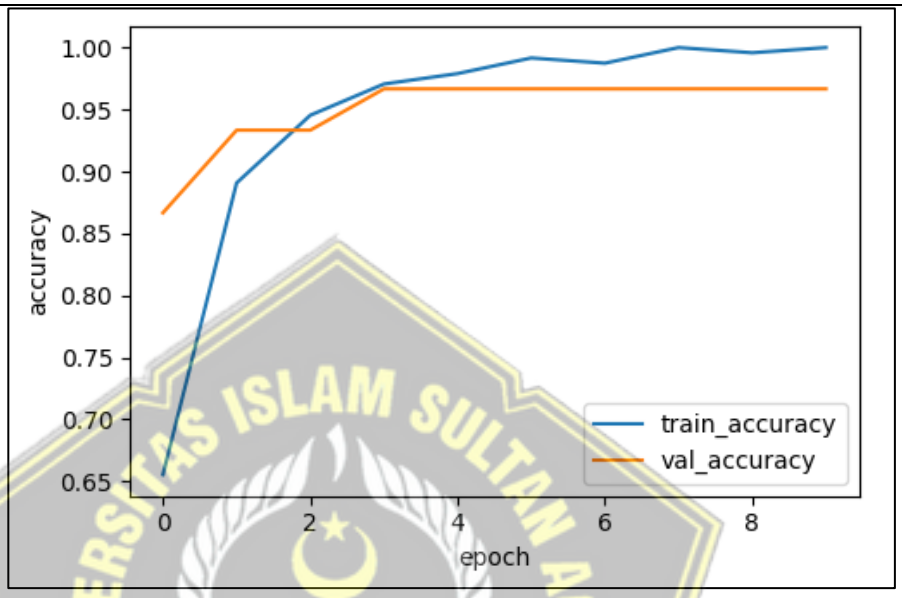
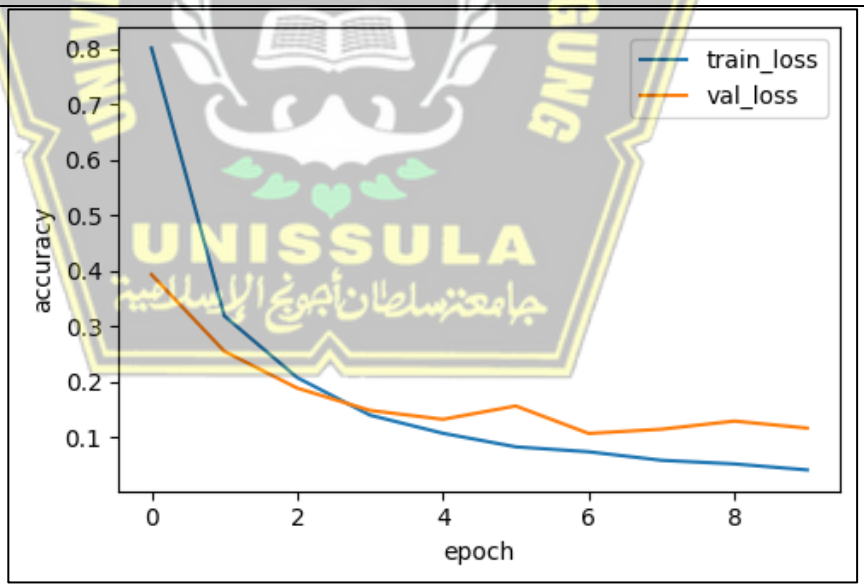
Konfigurasi yang digunakan seperti *Optimizer* yaitu "adam". *Epoch* sebesar 10 dengan menggunakan *early stopping* ketika akurasi mencapai 0.95, proses *training* akan dihentikan jika tidak terjadi peningkatan *accuracy* minimal 0.005. Untuk mencari hasil optimasi yang baik, dilakukan 4 konfigurasi berbeda dengan hyperparameter yang digunakan seperti yang sudah dijelaskan dalam tabel 3.1 sebelumnya.

Tabel 4. 2 Hasil Konfigurasi

| | ***Epoch*** | ***Acc*** | ***Loss*** | ***Val-acc*** | ***Val-loss*** |
|---|---|---|---|---|---|
| **Konfigurasi 1** | 10/10 | 1.0000 | 0.0417 | 0.9667 | 0.1169 |
| **Konfigurasi 2** | 6/10 | 0.9458 | 0.1901 | 0.9333 | 0.1844 |
| **Konfigurasi 3** | 8/10 | 0.8792 | 0.4130 | 0.9000 | 0.4342 |
| **Konfigurasi 4** | 10/10 | 0.7417 | 0.6302 | 0.7000 | 0.6013 |

Pada Tabel 4.2 ditampilkan hasil model terbaik dari setiap konfigurasi dalam proses *training*. Dari kelima konfigurasi tersebut, konfigurasi 1 memiliki akurasi tertinggi, mencapai 1.0000. Namun, model dengan akurasi tinggi tidak selalu berarti model yang baik. Model mungkin saja menghafal data pelatihan dan gagal untuk menggeneralisasi pada data baru. Sehingga penting untuk menganalisis grafik hasil pelatihan seperti grafik *loss* dan *accuracy* untuk memahami kinerja model secara lebih menyeluruh. Pada grafik *accuracy*, peningkatan menunjukkan peningkatan kinerja model. Sedangkan pada grafik *loss* untuk mencapai model dengan kesesuaian yang baik, grafik *loss* harus menurun secara stabil sementara *gap* antara *training loss* dan *validation loss* semakin berkurang dan berdekatan. Berikut tabel 4.3 merupakan grafik accuracy dan loss konfigurasi 1, konfigurasi 2, konfigurasi3, dan konfigurasi 4.

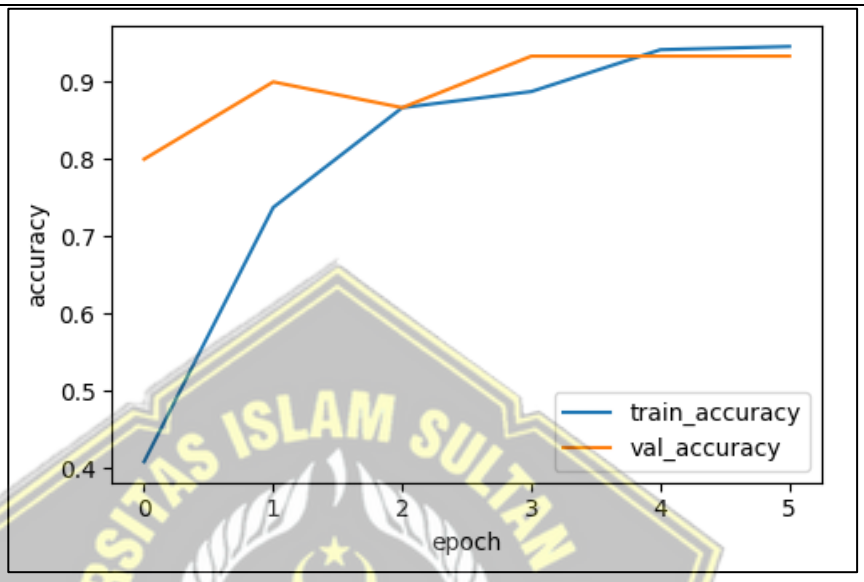
Tabel 4. 3 Grafik Accuracy dan Loss

<table>
<tr><td colspan="2">Konfigurasi 1<ul><li>Learning rate = 0.001</li><li>Batch size = 10</li></ul></td></tr>
<tr><td>Accuracy</td><td> 
</td></tr>
<tr><td>Loss</td><td> 
</td></tr>
</table>

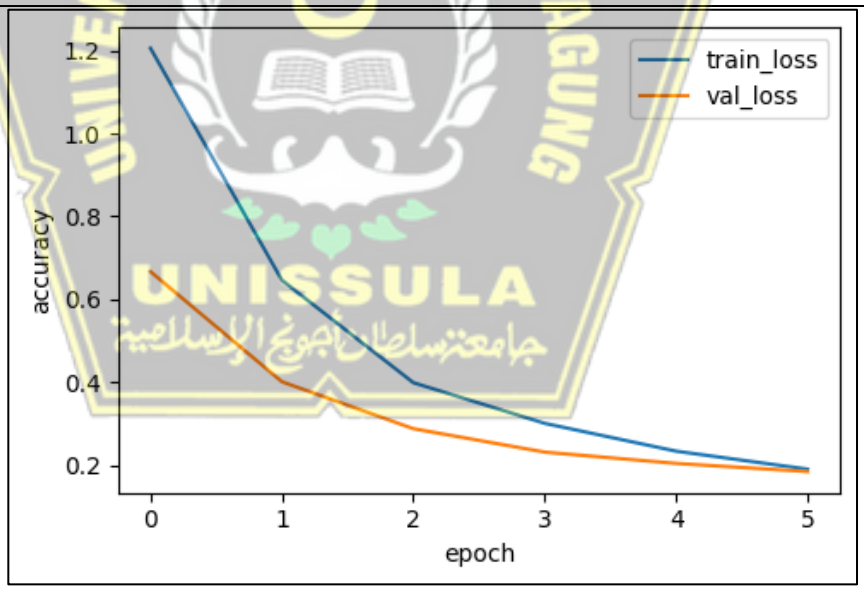
**Konfigurasi 2**

- Learning rate = 0.001
- Batch size = 30

*Accuracy*



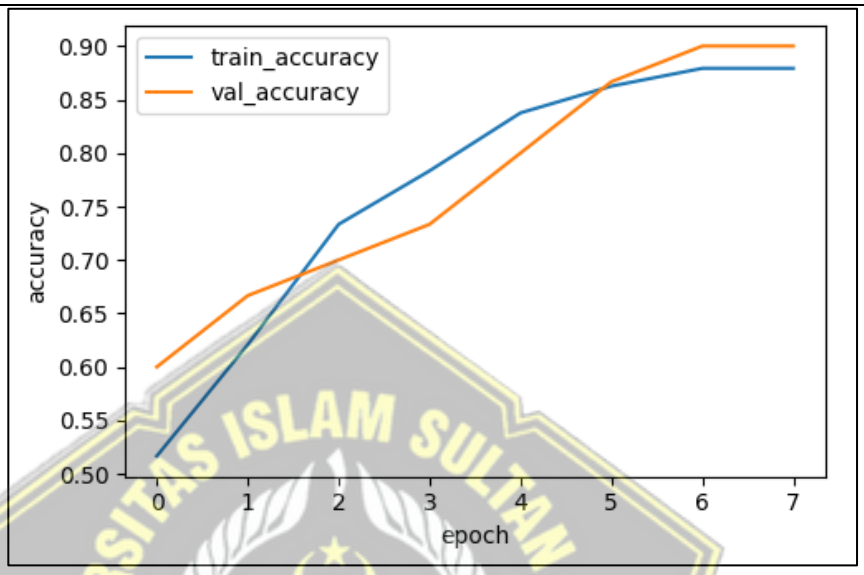

*Loss*



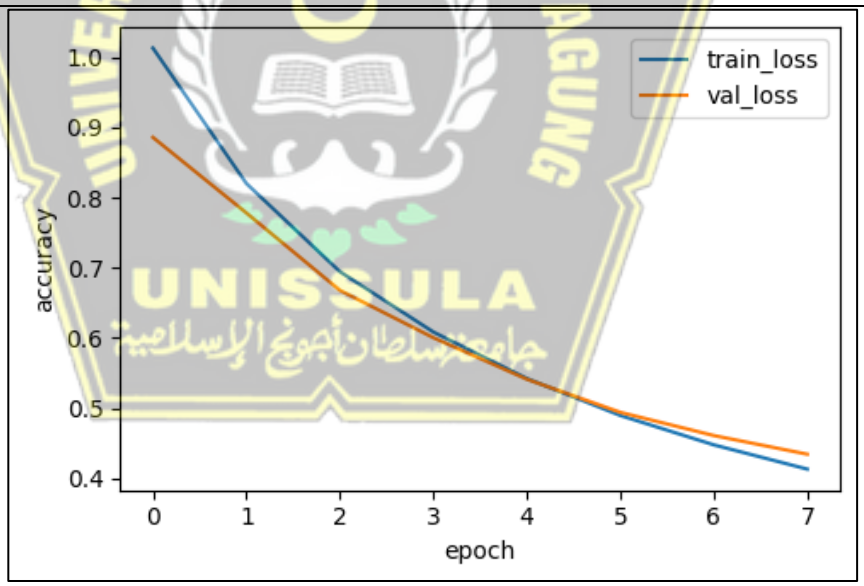
**Konfigurasi 3**

- Learning rate = 0.0001
- Batch size = 10

*Accuracy*



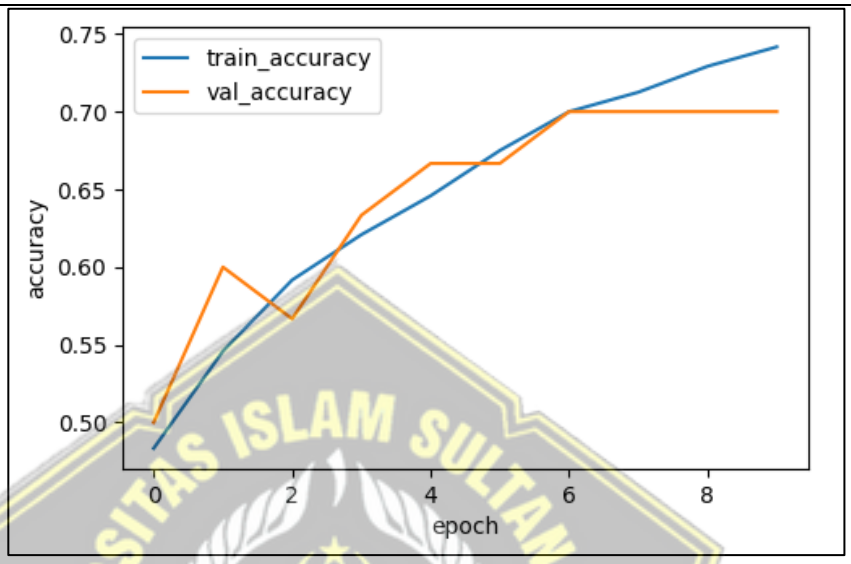

*Loss*



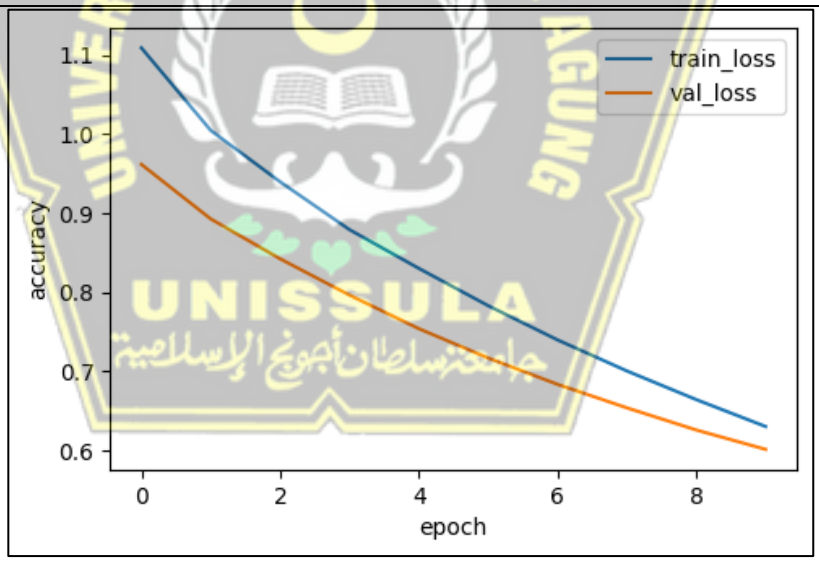
**Konfigurasi 4**

- Learning rate = 0.0001
- Batch size = 30

*Accuracy*




*Loss*

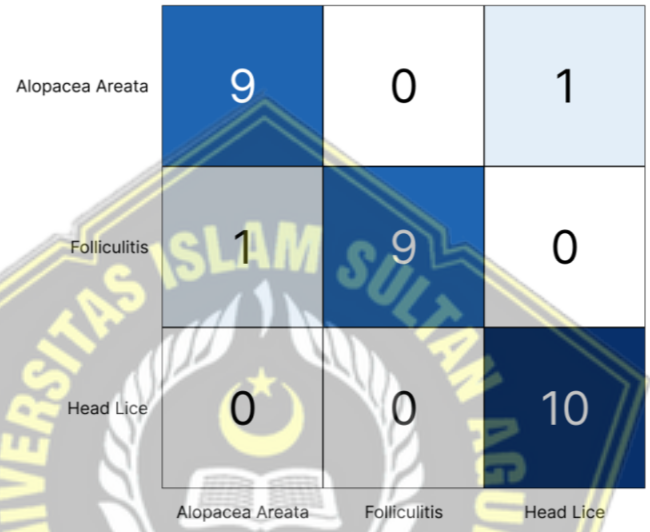



Tabel 4.3 menampilkan grafik *accuracy* dan grafik *loss* untuk setiap konfigurasi. Dengan data-data yang diberikan, konfigurasi 1 menunjukkan adanya *overfitting*. Meskipun model memiliki akurasi 1.00, terdapat *gap* yang signifikan antara grafik *training loss* dan *validation loss*, menandakan bahwa model terlalu adaptif terhadap *data training* dan mungkin telah menghafal pola yang spesifik pada data tersebut. Sehingga konfigurasi 1 sudah tidak perlu dimasukkan ke dalam tahap *testing*.

### 4.4.2 *Testing*

Pada tahap *testing*, dilakukan untuk mengevaluasi seberapa akurat sistem dalam mengklasifikasikan *data testing* yang terdiri dari 3 *class* dan berjumlah 30 citra. *Confusion matrix* digunakan untuk membantu dalam menghitung *accuracy, precission, recall* dan *f1 score*.

**a. Konfigurasi 2**




*Gambar 4. 9 Confusion Matrix Konfigurasi 2*

Berdasarkan *confusion matrix* pada gambar 4.9, sehingga bisa didapatkan nilai *True Positive* (TP), *False Negative* (FN), *False Pos*itive (FP), dan *True Negative* (TN) pada tiap *class* di konfigurasi 2 yang dituliskan pada tabel 4.4.

Tabel 4. 4 Nilai TP, FP, FN, dan TN pada *class* di konfigurasi 2

| | **TP** | **FP** | **FN** | **TN** |
|---|---|---|---|---|
| ***Alopecia Areata*** | 9 | 1 | 1 | 19 |
| ***Folliculitis*** | 9 | 0 | 1 | 20 |
| ***Head Lice*** | 10 | 1 | 0 | 19 |

Kemudian dengan nilai TP, FP, FN, dan TN dapat digunakan untuk menghitung nilai dari *accuracy, precission, recall,* dan *f1-score*. Pada tabel 4.5 menampilkan hasil *accuracy, precission, recall,* dan *f1-score* dari konfigurasi 2. Dan pada tabel 4.6 menampilkan hasil *precission, recall* dan *f1-score* pada tiap *class* di konfigurasi 2.

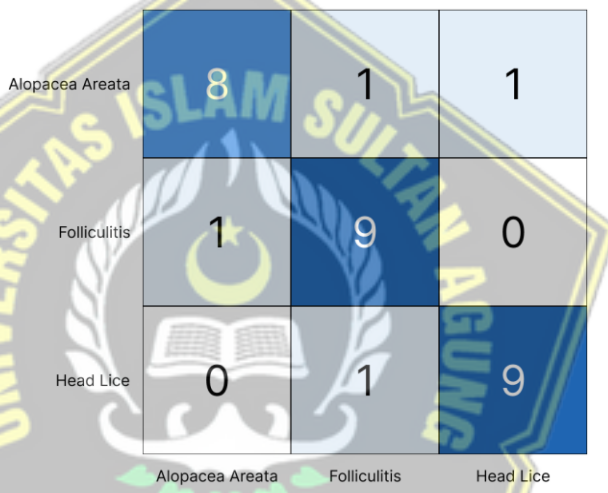
Tabel 4. 5 Accuracy, Precission, recall, dan f1-Score pada konfigurasi 2

| **Konfigurasi 2** | ***Accuracy*** | ***Precission*** | ***Recall*** | ***F1-Score*** |
|---|---|---|---|---|
| | 0.93 | 0.94 | 0.93 | 0.93 |

Tabel 4. 6 *Precission, recall,* dan *f1-score* tiap *class* di konfigurasi 2

| | ***Precission*** | ***Recall*** | ***F1-Score*** |
|---|---|---|---|
| ***Alopecia Areata*** | 0.90 | 0.90 | 0.90 |
| ***Folliculitis*** | 1 | 0.90 | 0.95 |
| ***Head Lice*** | 0.91 | 1 | 0.96 |

**b. Konfigurasi 3**




*Gambar 4. 10 Confusion Matrix Konfigurasi 3*

Berdasarkan *confusion matrix* pada gambar 4.10, sehingga bisa didapatkan nilai *True Positive* (TP), *False Negative* (FN), *False Positive* (FP), dan *True Negative* (TN) pada tiap *class* di konfigurasi 3 yang dituliskan pada tabel 4.7

Tabel 4. 7 Nilai TP, FP, FN, dan TN pada *class* di konfigurasi 3

| | **TP** | **FP** | **FN** | **TN** |
|---|---|---|---|---|
| ***Alopecia Areata*** | 8 | 1 | 2 | 19 |
| ***Folliculitis*** | 9 | 2 | 1 | 18 |
| ***Head Lice*** | 9 | 1 | 1 | 19 |

Kemudian dengan nilai TP, FP, FN, dan TN dapat digunakan untuk menghitung nilai dari *accuracy, precission, recall,* dan *f1-score*. Pada tabel 4.8 menampilkan hasil *accuracy, precission, recall,* dan *f1-score* dari konfigurasi 3. Dan pada tabel 4.9 menampilkan hasil *precission, recall* dan *f1-score* pada tiap *class* di konfigurasi 3.

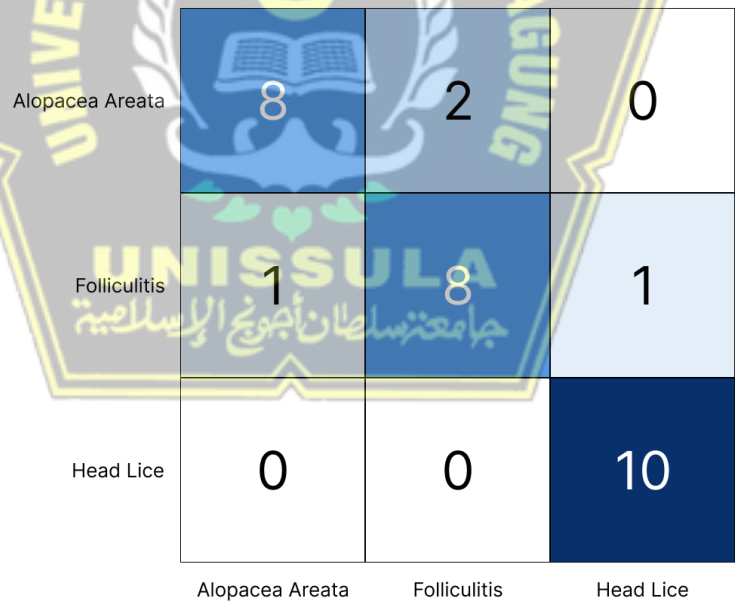
Tabel 4. 8 Accuracy, Precission, recall, dan f1-Score pada konfigurasi 3

| **Konfigurasi 3** | ***Accuracy*** | ***Precission*** | ***Recall*** | ***F1-Score*** |
|---|---|---|---|---|
| | 0.87 | 0.87 | 0.87 | 0.87 |

Tabel 4. 9 *Precission, recall,* dan *f1-score* tiap *class* di konfigurasi 3

| | ***Precission*** | ***Recall*** | ***F1-Score*** |
|---|---|---|---|
| ***Alopecia Areata*** | 0.89 | 0.80 | 0.85 |
| ***Folliculitis*** | 0.82 | 0.90 | 0.86 |
| ***Head Lice*** | 0.90 | 0.90 | 0.90 |

**c. Konfigurasi 4**




Gambar 4. 11 *Confusion Matrix* Konfigurasi 4

Berdasarkan *confusion matrix* pada gambar 4.11, sehingga bisa didapatkan nilai *True Positive* (TP), *False Negative* (FN), *False Positive* (FP), dan *True Negative* (TN) pada tiap *class* di konfigurasi 4 yang dituliskan pada tabel 4.10.

Tabel 4. 10 Nilai TP, FP, FN, dan TN pada class di konfigurasi 4

| | **TP** | **FP** | **FN** | **TN** |
|---|---|---|---|---|
| ***Alopecia Areata*** | 8 | 1 | 2 | 19 |
| ***Folliculitis*** | 8 | 2 | 2 | 18 |
| ***Head Lice*** | 10 | 1 | 0 | 19 |

Kemudian dengan nilai TP, FP, FN, dan TN dapat digunakan untuk menghitung nilai dari *accuracy, precission, recall,* dan *f1-score*. Pada tabel 4.11 menampilkan hasil *accuracy, precission, recall,* dan *f1-score* dari konfigurasi 4. Dan pada tabel 4.12 menampilkan hasil *precission, recall* dan *f1-score* pada tiap *class* di konfigurasi 4.

Tabel 4. 11 *Accuracy, Precission, recall,* dan *f1-Score* pada konfigurasi 4

| **Konfigurasi 4** | ***Accuracy*** | ***Precission*** | ***Recall*** | ***F1-Score*** |
|---|---|---|---|---|
| | 0.87 | 0.87 | 0.87 | 0.86 |

Tabel 4. 12 *Precission, recall,* dan *f1-score* tiap *class* di konfigurasi 4

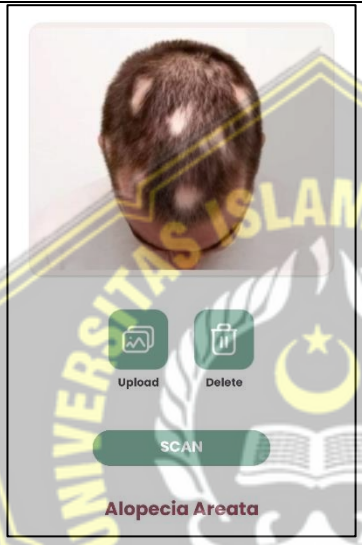
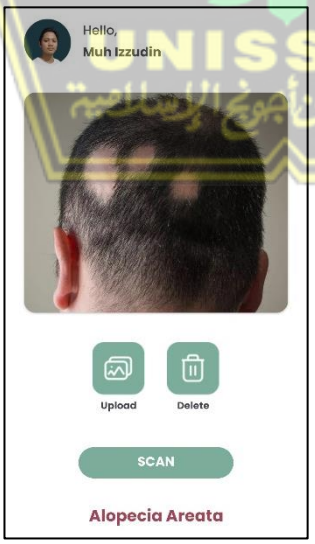
| | ***Precission*** | ***Recall*** | ***F1-Score*** |
|---|---|---|---|
| ***Alopecia Areata*** | 0.89 | 0.80 | 0.85 |
| ***Folliculitis*** | 0.80 | 0.80 | 0.80 |
| ***Head Lice*** | 0.91 | 1 | 0.96 |

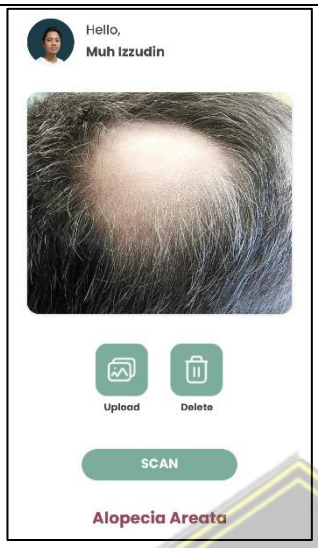
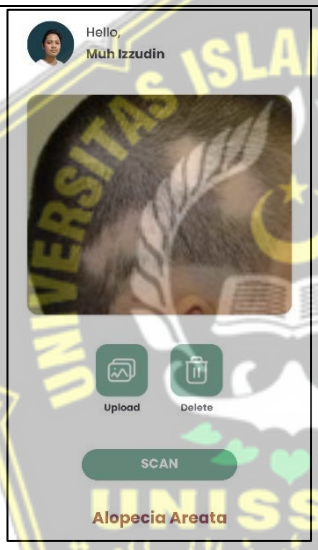
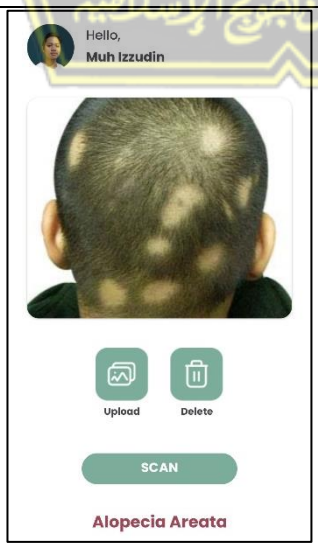
Berdasarkan hasil pengujian *convulation matrix* yang telah dilakukan, konfigurasi 2 menunjukkan kinerja yang paling baik. Dengan *accuracy* mencapai 0.93, *precission* mencapai 0.94, *recall* mencapai 0.93, dan nilai *F1-Score* mencapai 0.93.

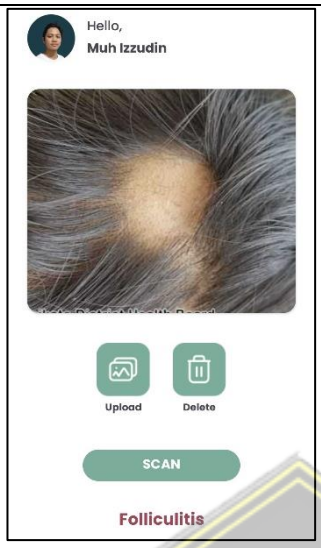
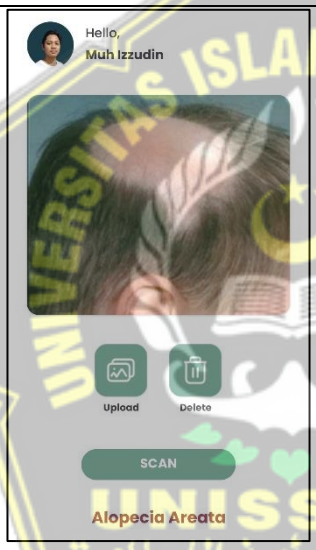
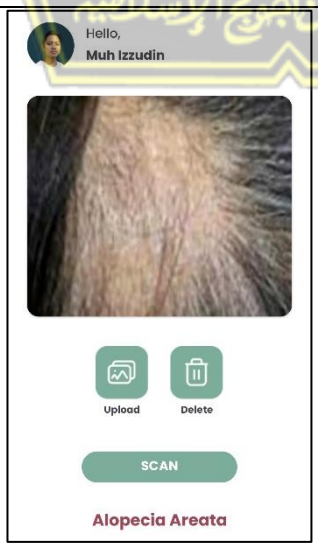
### 4.4.3 Hasil *Testing*

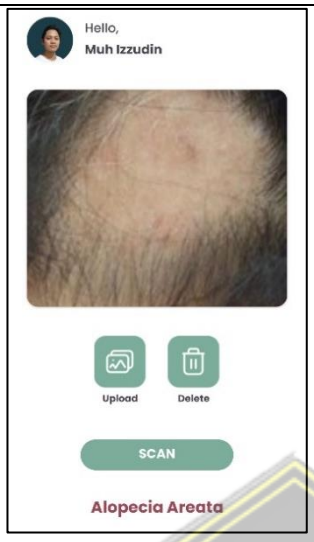
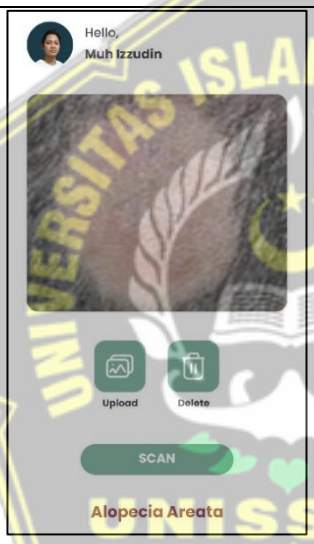
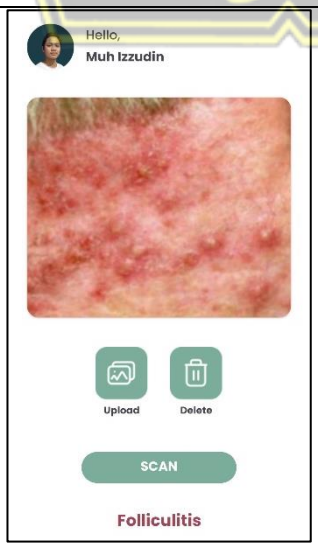
Pada bagian ini akan diperlihatkan hasil *testing* menggunakan model yang terbaik pada penelitian ini yaitu konfigurasi 2 dengan *data test*. Berikut hasil *testing* dengan model konfigurasi 2 dalam android yang dapat dilihat pada tabel 4.13.

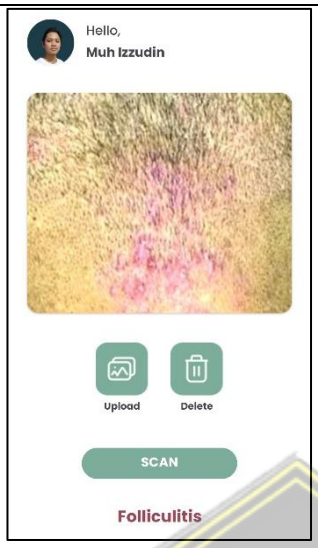
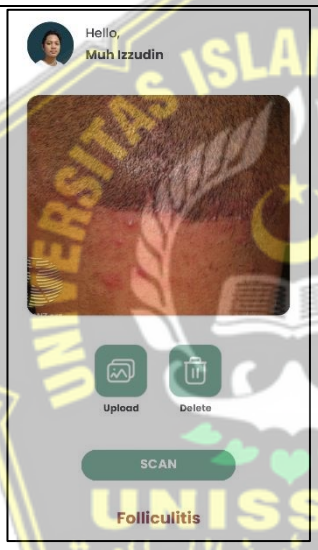
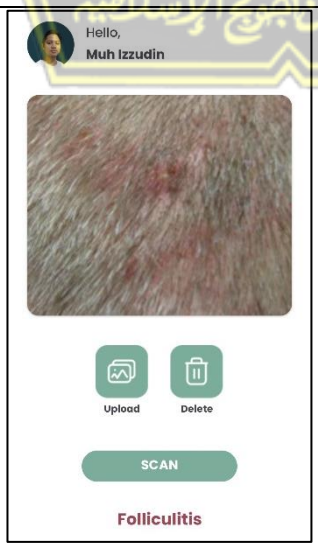
Tabel 4. 13 Hasil Uji Coba

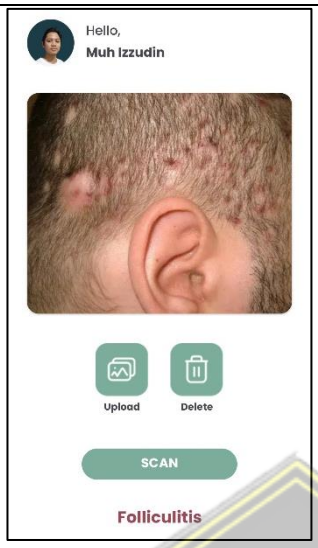
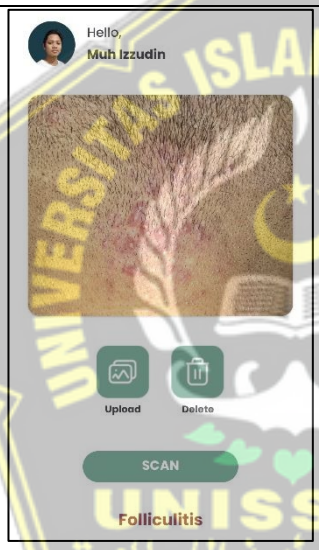
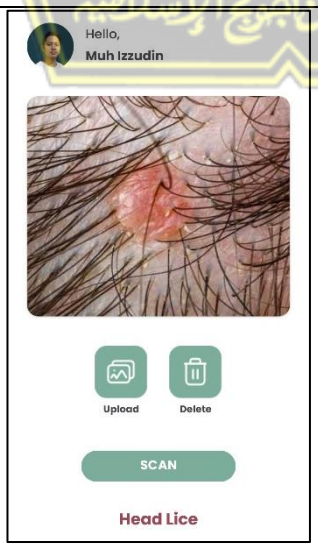
| No | Gambar | Keterangan |
|---|---|---|
| | ***Alopace Areata*** | |
| 1 |  | Sesuai |
| 2 |  | Sesuai |

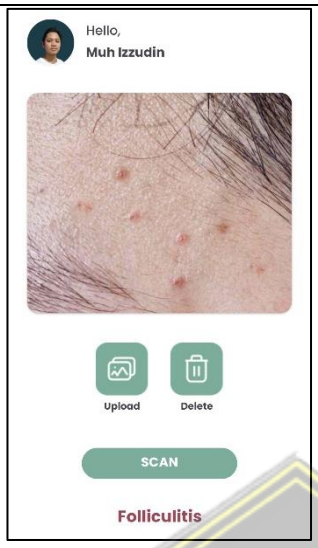
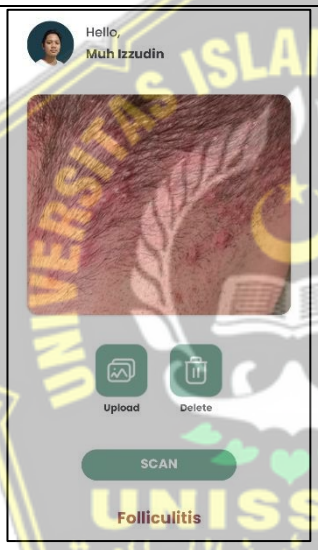
| | | |
|---|---|---|
| 3 |  | Sesuai |
| 4 |  | Sesuai |
| 5 |  | Sesuai |

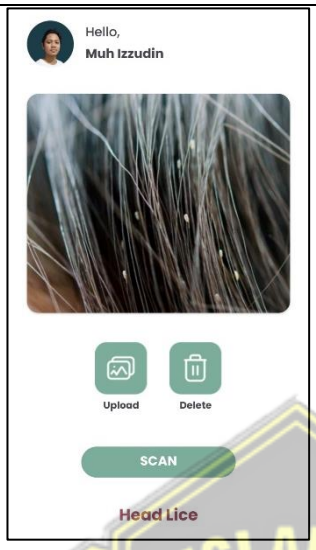
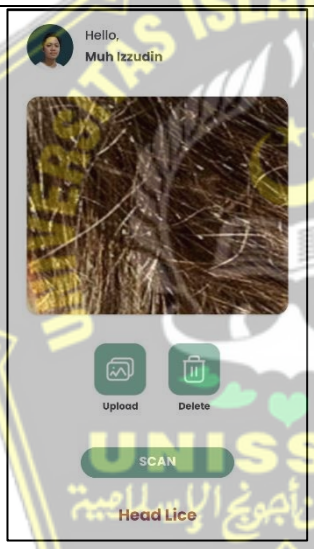
| | | |
|---|---|---|
| 6 |  | Tidak Sesuai |
| 7 |  | Sesuai |
| 8 |  | Sesuai |

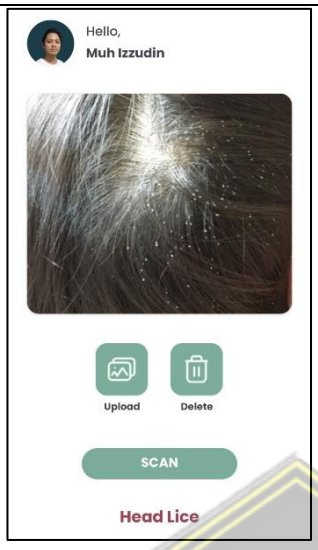
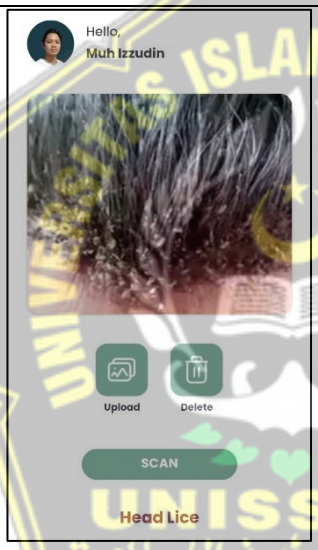
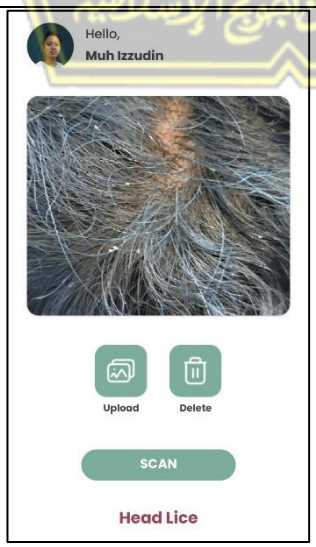
| | | |
|---|---|---|
| 9 |  | Sesuai |
| 10 |  | Sesuai |
| | ***Folliculitis*** | |
| 1 |  | Sesuai |

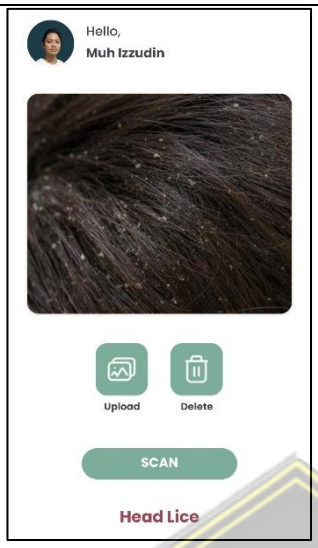
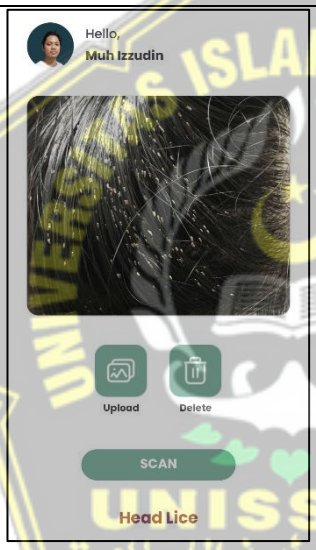
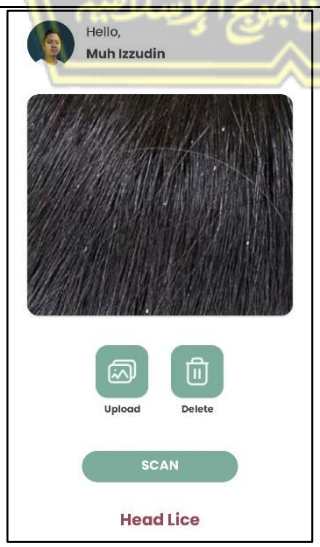
| | | |
|---|---|---|
| 2 |  | Sesuai |
| 3 |  | Sesuai |
| 4 |  | Sesuai |

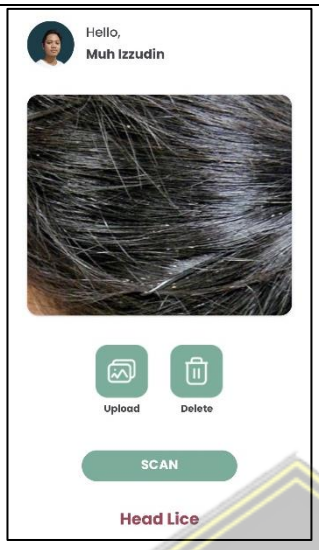
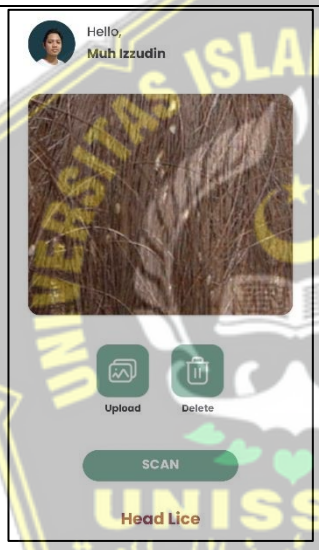
| | | |
|---|---|---|
| 5 |  | Sesuai |
| 6 |  | Sesuai |
| 7 |  | Tidak Sesuai |

| | | |
|---|---|---|
| 8 |  | Sesuai |
| 9 |  | Sesuai |
| 10 |  | Sesuai |

| | Head Lice | |
|---|---|---|
| 1 |  | Sesuai |
| 2 |  | Sesuai |

| | | |
|---|---|---|
| 3 |  | Sesuai |
| 4 |  | Sesuai |
| 5 |  | Sesuai |

| | | |
|---|---|---|
| 6 |  | Sesuai |
| 7 |  | Sesuai |
| 8 |  | Sesuai |

| | | |
|---|---|---|
| 9 |  | Sesuai |
| 10 |  | Sesuai |

Dari hasil uji coba klasifikasi konfigurasi 2 menggunakan 30 data testing penyakit kulit kepala, menunujukkan 28 citra terdeteksi sesuai dan 2 citra terdeteksi tidak sesuai yaitu:

1. Data keenam pada class alopecia areata terdeteksi sistem masuk dalam class folliculitis.
2. Data ketuju pada class folliculitis terdeteksi sistem masuk dalam class head lice.

# BAB IV
# HASIL DAN ANALISIS PENELITIAN

## 5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat ditarik kesimpulan bahwa sistem deteksi penyakit kulit kepala dapat diimplementasikan dengan cukup baik dalam sistem berbasis android. Berikut adalah beberapa poin kesimpulan yang penulis peroleh selama penelitian ini:

1. *MobileNetV2* bisa diadaptasi untuk mendeteksi penyakit kulit kepala dengan hasil yang baik pada sistem berbasis android.
2. Konfigurasi terbaik terdapat pada konfigurasi 2 dengan *optimizer* Adam, *learning rate* 0.001, *batch size* 30, dan *epoch* 6. Menghasilkan *accuracy* 0.9458, *loss* 0.1901, *validation accuracy* 0.9333, dan *validation loss* 0.1844
3. Model arsitektur *MobileNetV2* menunjukkan kinerja yang baik dalam mendeteksi jenis penyakit kulit kepala. Hasil pengujian menggunakan data test menunjukkan *accuracy* 0.93, *precission* 0.94, *recall* 0.93, dan *fi-score* sebesar 0.93.

## 5.2 Saran

Berdasarkan penelitian yang telah dilakukan, hasilnya menunjukkan performa yang cukup memuaskan. Meskipun demikian, untuk penelitian selanjutnya peneliti menyarankan:

1. Untuk meningkatkan kinerja model, disarankan untuk memperluas penggunaan dataset dengan jumlah gambar yang lebih besar dari 60 pada setiap kelas.
2. Disarankan untuk memperluas variasi jenis penyakit kulit kepala yang ada sehingga model dapat mempelajari berbagai karakteristik dan kasus yang lebih beragam.
3. Disarankan untuk meningkatkan arsitektur *MobileNetV2* dengan versi terbaru dari *MobileNet* untuk meningkatkan kemampuan pengenalan fitur yang relevan dan performa keseluruhan model.

# DAFTAR PUSTAKA


Abdi, D. A. (2020). Tinea Capitis. *Wal'afiat Hospital Journal*, *1*(2), 40. https://whj.umi.ac.id/index.php/whj/article/view/45

Ambarwati, N. S. S., Supiani, T., Laksmi, N. A., dan Atmanto, D. (2020). Peningkatan Kesejahteraan Dengan Pemanfaatan Lidah Buaya Untuk Perawatan Kulit Kepala Dan Rambut. *JKKP (Jurnal Kesejahteraan Keluarga dan Pendidikan)*, *7*(02), 117–129. https://doi.org/10.21009/jkkp.072.01

Annur, I. F., Umami, J., Annafii, M. N., Trisnaningrum, N., dan Putra, O. V. (2023). Klasifikasi Tingkat Keparahan Penyakit Leafblast Tanaman Padi Menggunakan MobileNetv2. *Fountain of Informatics Journal*, *8*(1), 7–14. https://doi.org/10.21111/fij.v8i1.9419

Azis, N., Pribadi, G., dan Nurcahya, M. S. (2020). *Analisa dan Perancangan Aplikasi Pembelajaran Bahasa Inggris Dasar Berbasis Android*. *4*(3), 1–5.

Badan Pusat Statistik Indonesia. (2022). *Statistik Telekomunikasi Indonesia 2022*.

Baihaqi, M. B., Litanianda, Y., dan Triyanto, A. (2022). Implementasi Tensor Flow Lite Pada Teachable Untuk Identifikasi Tanaman Aglonema Berbasis Android. *Komputek*, *6*(1), 70. https://doi.org/10.24269/jkt.v6i1.1143

Fachri, B., Nurleni, R., Harahap, R. R., dan Iskandar, I. (2022). Penerapan Metode Forward Chaining Dalam Mendiagnosa Penyakit Kulit Kepala. *Journal of Science and Social Research*, *5*(2), 284. https://doi.org/10.54314/jssr.v5i2.922

Geraldy, C., dan Lubis, C. (2020). Pendeteksian Dan Pengenalan Jenis Mobil Menggunakan Algoritma You Only Look Once Dan Convolutional Neural Network. *Jurnal Ilmu Komputer dan Sistem Informasi*, *8*(2), 197. https://doi.org/10.24912/jiksi.v8i2.11495

Harani, N. H., Prianto, C., dan Hasanah, M. (2019). Deteksi Objek Dan Pengenalan Karakter Plat Nomor Kendaraan Indonesia Menggunakan Metode Convolutional Neural Network (CNN) Berbasis Python. *Jurnal Teknik Informatika*, *11*(3), 47–53. https://ejurnal.ulbi.ac.id/index.php/informatika/article/view/658

Hendri Butar-Butar, R. J., dan Marpaung, N. L. (2023). Deep Learning untuk

Identifikasi Daun Tanaman Obat Menggunakan Transfer Learning MobileNetV2. *Jurnal Informatika: Jurnal Pengembangan IT*, *8*(2), 142–148. https://doi.org/10.30591/jpit.v8i2.5217

Hutauruk, D. J. R., Herlawati, H., dan Hendharsetiawan, A. A. (2022). Pendeteksian Wajah Bermasker Berbasis Android Menggunakan Algoritma Convolutional Neural Network (CNN). *Journal of Students' Research in Computer Science*, *3*(2), 207–216. https://doi.org/10.31599/jsrcs.v3i2.1693

Ibnul Rasidi, A., Pasaribu, Y. A. H., Ziqri, A., dan Adhinata, F. D. (2022). Klasifikasi Sampah Organik dan Non-Organik Menggunakan Convolutional Neural Network. *Jurnal Teknik Informatika dan Sistem Informasi*, *8*(1), 72–81. https://doi.org/10.28932/jutisi.v8i1.4314

Ilahiyah, S., dan Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem dan Teknologi Informasi Indonesia)*, *3*(2), 49–56.

Mellaratna, W. P., Perdana, R. D., dan Penulis, K. (2023). Alopesia Areata Dengan Pengobatan Triamcinolone Acetonide Alopecia Areata Treatment With Triamcinolone Acetonide. *Jurnal Medika Nusantara*, *1*(3), 61–68.

Muslim, F. P., Ridiar, A. F., Handiani, A., dan Pebriani, D. D. (2022). Kajian Pemahaman Generasi Z Terhadap Kutu Rambut ( Pediculus humanus ) Pada Manusia Study of Z Generation Understanding of Head Lice ( Pediculus humanus ) in Humans. *Prosiding SEMNAS BIO*, *1*(2809–8447), 303–321.

Naufal, M. F., dan Kusuma, S. F. (2021). Pendeteksi Citra Masker Wajah Menggunakan CNN dan Transfer Learning. *Jurnal Teknologi Informasi dan Ilmu Komputer*, *8*(6), 1293. https://doi.org/10.25126/jtiik.2021865201

Peryanto, A., Yudhana, A., dan Umar, R. (2020). Rancang Bangun Klasifikasi Citra Dengan Teknologi Deep Learning Berbasis Metode Convolutional Neural Network. *Jurnal Ilmiah Teknik Informatika*, *8*(2), 138. https://doi.org/10.22441/format.2019.v8.i2.007

Raihan, M., Allaam, R., dan Wibowo, A. T. (2021). Klasifikasi Genus Tanaman Anggrek Menggunakan Metode Convolutional Neural Network (CNN). *e-*

*Prceeding of Engineering*, *8*(2), 1–1153.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., dan Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

Sibarani, N. S., Munawar, G., dan Wisnuadhi, B. (2018). Analisis Performa Aplikasi Android Pada Bahasa Pemrograman Java dan Kotlin. In Prosiding Industrial Research Workshop and National Seminar. *Industrial Research Workshop and National Seminar*, *July*.

Sinaga, J. Y., Amalia, F., dan Santoso, E. (2020). Pengembangan Sistem Rekomendasi Produk Perawatan Kulit Berbasis Web Menggunakan Metode AHP. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *4*(11), 4071–4079. http://www.jurnal.kaputama.ac.id/index.php/SENATIKA/article/view/989

Sulistiyana, F., dan Anardani, S. (2023). *Aplikasi Deteksi Penyakit Tanaman Jagung Dengan Convolutional Neural Network dan Support Vector Machine*. 423–432.

Toena, H. T., dan Danarti, R. (2022). Diagnosis Alopesia Areata Pada Anak: Kasus Serial. *Cermin Dunia Kedokteran*, *49*(4), 214. https://doi.org/10.55175/cdk.v49i4.1824

Triyono, L., Thohari, A. N. A., Hestiningsih, I., dan Yobioktabera, A. (2008). *Klasifikasi Penyakit Kulit Menggunakan Metode Convolutional*. 282.

Winnarto, M. N., Mailasari, M., dan Purnamawati, A. (2022). Klasifikasi Jenis Tumor Otak Menggunakan Arsitekture Mobilenet V2. *Jurnal SIMETRIS*, *13*(2), 1–12.

Zalukhu, A., Purba, S., Darma, D., Zalukhu[1], A., Purba[2], S., Darma[3], D., Teknik Informatika, M., dan Industri, F. T. (2023). Perangkat Lunak Aplikasi Pembelajaran Flowchart. *Jurnal Teknologi, Informasi dan Industri*, *4*(1), 61–70. https://ejurnal.istp.ac.id/index.php/jtii/article/view/351