IMPLEMENTASI *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI TUBERKULOSIS PADA CITRA RONTGEN DADA

MUHAMMAD ALIM MA'ARIJ

H071201010

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR

2024

IMPLEMENTASI *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI TUBERKULOSIS PADA CITRA RONTGEN DADA

MUHAMMAD ALIM MA'ARIJ

H071201010

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2024

IMPLEMENTASI *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI TUBERKULOSIS PADA CITRA RONTGEN DADA

MUHAMMAD ALIM MA'ARIJ
H071201010

Skripsi

sebagai salah satu syarat untuk mencapai gelar sarjana

Program Studi Sistem Informasi

pada

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2024

SKRIPSI

IMPLEMENTASI *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI TUBERKULOSIS PADA CITRA RONTGEN DADA

MUHAMMAD ALIM MA'ARIJ

H071201010

Skripsi,

telah dipertahankan di depan Panitia Ujian Sarjana Sistem Informasi Pada 17 September 2024 dan dinyatakan telah memenuhi syarat kelulusan pada

Program Studi Sistem Informasi

Departemen Matematika

Fakultas Matematika Dan Ilmu Pengetahuan Alam

Universitas Hasanuddin

Makassar

Mengesahkan:

Pembimbing tugas akhir,

Dr. Eng. Armin Lawi, S.Si., M. Eng.

NIP. 197204231995121001

Mengetahui:

Ketua Program Studi

Prof. Dr. Jeffry Kusuma

NIP. 196411121987031002

PERNYATAAN KEASLIAN SKRIPSI DAN PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa, skripsi berjudul "Implementasi *Convolutional Neural Network* untuk Klasifikasi Tuberkulosis pada Citra Rontgen Dada" adalah benar karya saya dengan arahan dari pembimbing (Dr. Eng. Armin Lawi, S.Si., M. Eng.) Karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuk apapun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka skripsi. Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini adalah karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut berdasarkan aturan yang berlaku.

Dengan ini saya melimpahkan hak cipta (hak ekonomis) dari karya tulis saya berupa skripsi ini kepada Universitas Hasanuddin.

Makassar, 17 September 2024

MUHAMMAD ALIM MA'ARIJ

H071201010

UCAPAN TERIMA KASIH

Alhamdulillah, segala puji hanya bagi Allah SWT. Dengan rahmat dan kasih-Nya, saya dapat menjalani setiap detik kehidupan ini dengan penuh syukur dan keikhlasan. Nikmat-Nya yang tak terhingga telah mengiringi perjalanan saya sebagai mahasiswa hingga gelar sarjana semakin dekat. Semua ini terwujud atas karunia-Nya, yang membimbing dan menguatkan saya di setiap ujian dan tantangan yang saya hadapi.

Penelitian ini dan penyelesaian skripsi ini tidak akan terlaksana tanpa bimbingan, arahan, dan diskusi dari **Dr. Eng. Armin Lawi, S.Si., M.Eng.** Saya mengucapkan terima kasih yang mendalam dan tulus kepada beliau atas segala waktu dan ilmunya. Ucapan penghargaan yang sebesar-besarnya juga saya sampaikan kepada Bapak **Dr. Khaeruddin, M.Sc**. dan **Edy Saputra Rusdi, S.Si., M.Si.**, yang kritik dan sarannya telah membantu meningkatkan kualitas penelitian ini.

Terima kasih yang dalam kepada kawan-kawan seperjuangan; **Fudhol**, **Hamsa**, **Efendy**, **Hajid**, **Nawwaf**, **Eja**, **Nur**, **Elsah**, **Ara**, **Febi**, **Pia**, **Ekki**, **Halima**, **Napus**, **Talitha**, **Ika**, **Raka**, **Wardah, Indah** dan **Adil**. Kalian bukan hanya teman, tapi saudara yang selalu ada di setiap langkah perjalanan ini. Bersama kalian, tantangan yang berat terasa lebih ringan, dan setiap keberhasilan menjadi lebih bermakna. Terima kasih atas setiap dukungan, tawa, dan bahkan kesulitan yang kita hadapi bersama. Kalian adalah bagian penting dari pencapaian ini, dan saya sangat bersyukur bisa melewati semua ini bersama kalian. Semoga kita semua selalu diberi kesuksesan dan kebahagiaan di masa depan. Terima kasih untuk segalanya.

Tak lupa, saya ucapkan terima kasih yang sebesar-besarnya kepada **Ummul**, **Awang**, **Rizki**, **Mufti**, dan **Rahmatullah**, yang telah membantu dan mendukung saya dalam menyusun skripsi ini. Terima kasih atas diskusi, bantuan moril, dan dorongan semangat yang selalu menguatkan saya di setiap langkah.

Terima kasih juga untuk keluarga BOBA: **Ghazwul**, **Ince**, **Mita**, **Icha**, **Nayah**, **Hazyim**, **Aldi**, **Nisa**, dan **Bunda Emje**, atas kebersamaan dan dukungan kalian. Dan terima kasih kepada teman-teman **HORI20NTAL**, **MIPA 2020**, **BE Himatika FMIPA Unhas 2022-2023**, dan **BEM FMIPA Unhas 2023/2024**. Pengalaman bersama kalian sangat berarti dan memperkaya perjalanan ini.

Akhirnya, kepada kedua orang tua tercinta, saya mengucapkan terima kasih yang tak terhingga dan sembah sujud atas doa, pengorbanan, dan motivasi mereka selama saya menempuh pendidikan.

Penulis,

Muhammad Alim Ma’arij

ABSTRAK

MUHAMMAD ALIM MA'ARIJ. **Implementasi *Convolutional Neural Network* untuk Klasifikasi Tuberkulosis pada Citra Rontgen Dada** (dibimbing oleh Dr. Eng. Armin Lawi, S.Si., M. Eng.)

Latar belakang. Tuberkulosis (TBC) adalah penyakit menular serius yang disebabkan oleh bakteri *Mycobacterium tuberculosis*, yang berdampak signifikan pada kesehatan global, terutama di negara berkembang seperti Indonesia. Metode deteksi TBC yang ada saat ini, seperti pemeriksaan dahak dan rontgen dada, memiliki keterbatasan dalam hal kecepatan, biaya, dan akurasi, sering kali menghasilkan sensitivitas dan spesifisitas yang rendah. Untuk mengatasi tantangan ini, diperlukan pendekatan baru yang lebih cepat, akurat, dan terjangkau dalam mendeteksi TBC, terutama pada tahap awal penyakit. **Tujuan**. Penelitian ini bertujuan untuk meningkatkan akurasi deteksi TBC pada citra rontgen dada dengan menggunakan model *Convolutional Neural Network* (*CNN*) terbaru dan mengembangkan aplikasi web untuk klasifikasi TBC secara otomatis. **Metode**. Data rontgen dada yang dikumpulkan melalui proses *pre-processing*, termasuk penyesuian ulang dan augmentasi data untuk memastikan konsistensi input. Beberapa model *CNN*, seperti *EfficientNetV2B0*, *MobileNetV3*, *NASNetMobile*, dan *DenseNet121*, dilatih menggunakan *transfer learning* dan dievaluasi untuk menentukan model terbaik. Model dengan kinerja terbaik kemudian diimplementasikan dalam aplikasi web menggunakan *Streamlit*, yang memungkinkan pengguna untuk mengunggah citra rontgen dada dan mendapatkan hasil klasifikasi secara otomatis. **Hasil**. Di antara model yang diuji, *MobileNetV3* menunjukkan kinerja terbaik dengan akurasi 98,10%, presisi 100%, dan *F1-Score* 99,84%, menjadikannya pilihan yang unggul untuk deteksi TBC. **Kesimpulan**. *MobileNetV3* terbukti efektif dalam mendeteksi TBC pada citra rontgen dada, dan implementasi model ini dalam aplikasi web menghasilkan alat yang praktis dan dapat diandalkan untuk membantu dalam diagnosis klinis TBC.

Kata kunci: *Convolutional Neural Network*; Klasifikasi Rontgen Dada; *Transfer Learning*; Evaluasi Model; Aplikasi Web.

ABSTRACT

MUHAMMAD ALIM MA'ARIJ. **Implementation of Convolutional Neural Network for Tuberculosis Classification on Chest X-ray Images** (supervised by Dr. Eng. Armin Lawi, S.Si., M. Eng.)

Background. Tuberculosis (TB) is a serious infectious disease caused by the bacterium Mycobacterium tuberculosis, which has a significant impact on global health, especially in developing countries such as Indonesia. Current TB detection methods, such as sputum examination and chest X-ray, have limitations in terms of speed, cost, and accuracy, often resulting in low sensitivity and specificity. To overcome these challenges, new approaches are needed that are faster, more accurate, and more affordable in detecting TB, especially in the early stages of the disease. **Objectives**. This study aims to improve the accuracy of TB detection in chest X-ray images using the latest Convolutional Neural Network (CNN) model and develop a web application for automatic TB classification. **Methods**. The collected chest X-ray data went through pre-processing, including re-scaling and data augmentation to ensure input consistency. Several CNN models, such as EfficientNetV2B0, MobileNetV3, NASNetMobile, and DenseNet121, were trained using transfer learning and evaluated to determine the best model. The best performing model was then implemented in a web application using Streamlit, which allows users to upload chest X-ray images and get classification results automatically. **Results**. Among the tested models, MobileNetV3 showed the best performance with 98.10% accuracy, 100% precision, and 99.84% F1-Score, making it a superior choice for TB detection. **Conclusion**. MobileNetV3 proved effective in detecting tuberculosis in chest X-ray images, and the implementation of this model in a web application resulted in a practical and reliable tool to aid in the clinical diagnosis of tuberculosis.

Keywords: Convolutional Neural Network; Chest X-ray Classification; Transfer Learning; Model Evaluation; Web Application.

DAFTAR ISI

Halaman

DAFTAR TABEL

Nomor urut Halaman

DAFTAR GAMBAR

Nomor urut Halaman

BAB I

Pendahuluan

1.1 Latar Belakang

Tuberkulosis (TBC) adalah penyakit menular yang disebabkan oleh bakteri *mycobacterium tuberculosis*. TBC dikenal sebagai penyakit menular yang signifikan dan bertanggung jawab atas angka kematian yang tinggi. Penyakit ini biasanya menyerang paru-paru tetapi juga dapat menjangkiti beberapa area tubuh lain seperti tulang, usus, saluran kemih, dan kulit. Menurut Laporan *World Health Organization* (WHO) dalam *Global Tuberculosis Report 2023*, jumlah orang yang terdiagnosis TBC secara global mencapai 7,5 juta kasus pada tahun 2022. Ini adalah angka tertinggi sejak WHO memulai pemantauan TBC global pada tahun 1995. Di Indonesia, diperkirakan terdapat sekitar 750.000 kasus TBC pada tahun 2022, menjadikan Indonesia sebagai negara penyumbang penderita TBC terbanyak kedua di dunia setelah India. Oleh karena itu, deteksi penyakit ini pada tahap awal sangat penting untuk mendapatkan pengobatan yang tepat.

Saat ini, metode klasifikasi yang tersedia seringkali mahal, memakan waktu lama, dan hasilnya tidak selalu akurat, menunjukkan sensitivitas dan spesifisitas yang rendah (Briceno, Sergent, Benites, & Alocilja, 2019). Pengenalan yang akurat terhadap TBC menjadi kunci penting dalam upaya untuk menghilangkan penyakit ini sepenuhnya (Uplekar et al., 2015). Beberapa tes telah dikembangkan untuk mengklasifikasikan TBC, termasuk tes kulit, tes dahak, *interferon-gamma release assay* (IGRA), dan *Chest X-ray* (rontgen dada). Meskipun demikian, tes kulit tidak selalu dapat mengidentifikasi TBC secara akurat, sementara tes dahak memakan waktu dan bergantung pada kemampuan pasien untuk memberikan sampel (Satheeshkumar & Arunachalam, 2020). IGRA memiliki biaya tinggi dan sensitivitas yang rendah. Di sisi lain, rontgen dada merupakan metode yang lebih ekonomis dan cepat, meskipun keakuratannya tergantung pada kemampuan diagnosis dokter yang dapat dipengaruhi oleh kelelahan dan kesalahan manusia lainnya (Lu, Chen, Yu, & Chen, 2016). Namun, dengan diagnosis yang tepat, pengobatan yang sesuai dapat diberikan untuk memfasilitasi pemulihan yang cepat.

Dalam upaya untuk meningkatkan keakuratan diagnosis TBC, *Artificial Intelligence* (AI) dalam layanan kesehatan telah menjadi komponen krusial dalam sistem layanan kesehatan modern. Banyak penelitian telah dilakukan untuk mengembangkan sistem *computer-aided diagnosis* (CAD) guna meningkatkan kemampuan dokter dan ahli radiologi dalam membuat keputusan yang efektif dalam memberikan perawatan medis kepada pasien (Kotei & Thirunavukarasu, 2022). *Deep learning*, yang merupakan cabang dari AI, telah mengalami perkembangan pesat baru-baru ini dan memberikan solusi yang luar biasa untuk banyak tugas CAD. *Deep learning* merupakan cabang dari *machine learning* yang terinspirasi dari kortex

manusia dengan menerapkan jaringan syaraf buatan yang memiliki banyak *hidden layer* (Santoso, 2018). Dalam konteks pengembangan sistem CAD untuk klasifikasi TBC, *Convolutional Neural Network* (*CNN*) telah menjadi fokus utama. *CNN* telah terbukti efektif dalam mengekstraksi fitur dari citra rontgen dada dan mengklasifikasikannya sebagai normal atau abnormal.

Banyak penelitian telah dilakukan untuk mengklasifikasikan TBC pada rontgen dada dengan menggunakan pendekatan *transfer learning*. Pendekatan ini mengacu pada penggunaan model yang telah dilatih sebelumnya pada kumpulan data tertentu sebagai titik awal untuk membangun model baru dengan menggunakan kumpulan data dan atribut yang berbeda. *Transfer learning* adalah strategi umum di mana model yang dibuat untuk satu tugas digunakan sebagai titik awal untuk mengembangkan model untuk tugas yang berbeda (Gupta, Pathak, & Kumar, 2022). Misalnya, studi oleh Rahman et al., (2020) menggunakan model *ChexNet* yang telah dilatih sebelumnya pada *dataset ImageNet* dan berhasil mencapai akurasi deteksi TBC sebesar 96,47%. Demikian pula, penelitian oleh Kotei & Thirunavukarasu (2022) mengadopsi pendekatan serupa menggunakan model *VGG19*, yang menghasilkan akurasi sebesar 92,86% pada *dataset* rontgen dada mereka. Penelitian lainnya oleh Aulia & Hadiyoso (2022) menggunakan model *VGG16* dan menghasilkan akurasi sebesar 99,76%.

Oleh karena itu, penelitian ini bertujuan untuk menguji beberapa arsitektur *CNN* terbaru menggunakan *transfer learning* dan menerapkan pendekatan yang diperbarui dalam rangka meningkatkan akurasi dan efisiensi deteksi TBC pada citra rontgen dada. Dengan menerapkan *CNN* untuk identifikasi penyakit TBC pada rontgen dada, diharapkan dapat memberikan kontribusi signifikan dalam upaya global untuk mengatasi beban penyakit TBC dan meningkatkan kesehatan masyarakat secara keseluruhan.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana mekanisme *pre-processing* data yang dibutuhkan untuk membangun model klasifikasi TBC menggunakan citra rontgen dada?
2. Bagaimana membangun model *CNN EffiicientNet, MobileNet, NASNetMobile,* dan *DenseNet* menggunakan data citra rontgen dada?
3. Bagaimana mengevaluasi kinerja model *CNN* yang telah dibangun menggunakan citra rontgen dada?
4. Bagaimana implementasi model *CNN* yang menghasilkan aplikasi web?

1.3 Tujuan dan Manfaat

Berdasarkan uraian pada latar belakang, dapat disimpulkan tujuan penelitian sebagai berikut:

1. Memahami dan menerapkan mekanisme *pre-processing* data yang tepat dalam membangun model klasifikasi TBC menggunakan citra rontgen dada.
2. Membangun model *CNN* dengan arsitektur *EffiicientNet, MobileNet, NASNetMobile, dan DenseNet* untuk klasifikasi TBC pada citra rontgen dada.
3. Mengevaluasi dan menentukan model *CNN* terbaik yang telah dibangun untuk deteksi TBC pada citra rontgen dada.
4. Membangun aplikasi web untuk klasifikasi penyakit TBC hasil implementasi model *CNN*.

Sedangkan manfaat dari penelitian sebagai berikut:

1. Memberikan panduan tentang langkah-langkah *pre-processing* data, pembangunan model *CNN*, evaluasi kinerja, dan implementasi ke sistem informasi.
2. Memberikan wawasan tentang performa berbagai arsitektur *CNN* terbaru dalam deteksi TBC pada citra rontgen dada.
3. Menyediakan metode yang lebih efektif untuk menklasifikasi TBC pada tahap awal, memungkinkan pengobatan yang lebih cepat dan tepat.
4. Memberikan manfaat signifikan dalam layanan kesehatan dengan membangun aplikasi web untuk klasifikasi penyakit TBC.

1.4 Batasan Masalah

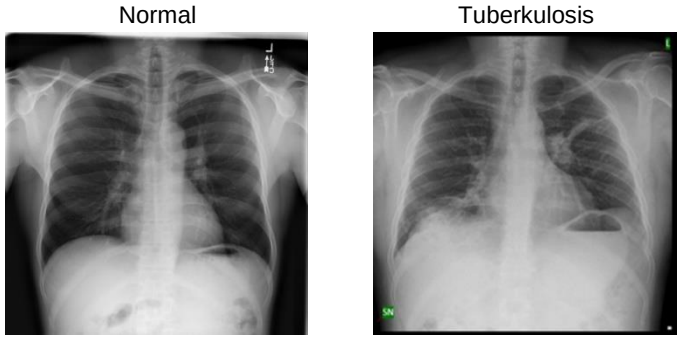
Batasan masalah pada penelitian ini adalah:

1. Penelitian ini hanya akan menggunakan citra rontgen dada sebagai data input untuk mendeteksi TBC.
2. Penelitian akan menggunakan arsitektur *CNN* yang telah ditentukan dan tidak akan mencakup pengembangan arsitektur *CNN* baru.
3. Data yang digunakan dalam penelitian ini akan dibatasi pada *dataset* yang tersedia secara publik atau yang diizinkan untuk digunakan dalam penelitian.
4. Evaluasi performa model akan dibatasi pada metrik akurasi, *recall*, *F1-Score*, presisi, dan *Confusion Matrix*.

1.5 Teori

1.5.1 Tuberkulosis (TBC)

Tuberkulosis (TBC) merupakan penyakit kronik, menular, yang disebabkan oleh *mycobacterium tuberculosis*, ditandai dengan terbentuknya jaringan yang rusak dan mati sebagai respons terhadap bakteri tersebut (Sejati & Sofiana, 2015). TBC menyebar melalui udara ketika seseorang dengan TBC batuk, bersin, berbicara, dan melepaskan droplet kecil yang mengandung bakteri ke udara. *Chest X-ray* atau rontgen dada adalah salah satu metode yang digunakan untuk mendeteksi dan mendiagnosis TBC. Rontgen dada digunakan untuk melihat kerusakan atau perubahan pada paru-paru yang disebabkan oleh TBC.



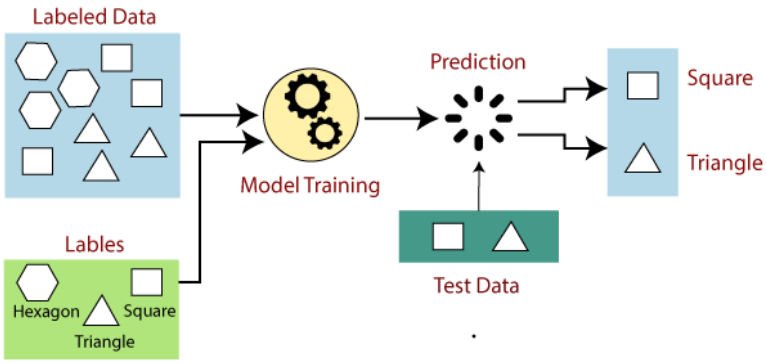
Gambar 1. Perbedaan rontgen dada normal dan TBC

Dapat dilihat dari Gambar 1, rontgen dada yang terkena TBC menunjukkan bintik-bintik, rongga, atau area yang tidak normal di paru-paru. Rontgen dada tidak hanya membantu dalam mendiagnosis TBC, tetapi juga berguna untuk memantau respons terhadap pengobatan, mendeteksi komplikasi seperti efusi pleura, dan mengevaluasi kondisi paru-paru pasien secara keseluruhan. Dengan demikian, rontgen dada merupakan alat diagnostik yang penting dalam manajemen dan pengobatan TBC, membantu dokter dalam merencanakan dan menyesuaikan strategi pengobatan yang tepat.

1.5.2 *Image Classification*

Image classification (klasifikasi citra) adalah salah satu cabang dari pengenalan pola yang bertujuan untuk mengidentifikasi dan mengkategorikan objek dalam citra digital menjadi beberapa kelas. Klasifikasi citra adalah salah satu topik penelitian yang sedang populer dalam bidang *Computer Vision* (CV) dan proses ini biasanya terdiri dari tiga bagian penting: *pre-processing*, *feature extraction* dan *classification* (Xin & Wang, 2019). Illustrasi prosesnya dapat dilihat pada Gambar 2.

a. Tahap *pre-processing* bertujuan untuk meningkatkan kualitas citra dengan mengurangi noise, meningkatkan kontras, dan menormalkan ukuran serta orientasi citra, sehingga lebih sesuai untuk analisis.
b. Tahap *feature extraction* dilakukan untuk mengekstrak informasi penting dari citra, seperti tekstur, bentuk, dan pola, yang dapat digunakan untuk membedakan antara berbagai kelas.
c. Tahap *classification* menggunakan model *machine learning* atau *deep learning*, seperti *CNN* untuk mengkategorikan citra berdasarkan fitur yang diekstraksi.



Gambar 2. Proses *image classification*
https://www.javatpoint.com/supervised-machine-learning

Klasifikasi citra memiliki peran penting dalam berbagai aplikasi, termasuk pengenalan wajah, deteksi objek, diagnosis medis, dan penginderaan jauh, serta didukung oleh perkembangan teknologi *machine learning* dan *deep learning* yang semakin meningkatkan akurasi dan efisiensi proses ini.

1.5.3 *Artificial Intelligence*

Artificial Intelligence (AI) adalah cabang dari ilmu komputer yang fokus pada otomasi perilaku yang cerdas (Luger & Stubblefield, 1993). AI dapat bekerja sendiri atau digabung dengan teknologi lain seperti sensor, lokasi geografis, dan robotika untuk melakukan tugas yang lainnya memerlukan kecerdasan manusia atau intervensi. Contoh aplikasi AI yang umum ditemui dalam kehidupan sehari-hari termasuk asisten digital, panduan GPS, kendaraan otonom, dan alat generatif AI seperti *Open AI's Chat GPT*. Secara garis besar, AI dapat dibedakan menjadi empat kategori:

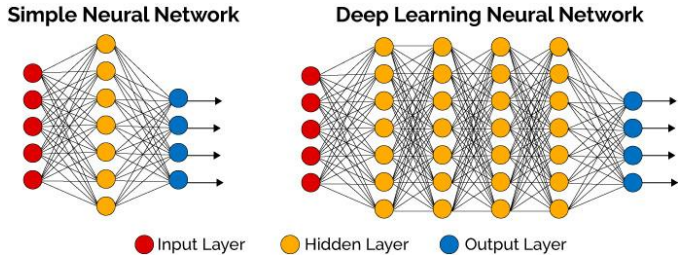
d. *Thinking Humanly*: Mencoba menangkap pemikiran manusia melalui introspeksi dan penelitian psikologi.

e. *Acting Humanly*: Mengacu pada uji Turing yang dirancang oleh Alan Turing pada tahun 1950 untuk menguji apakah komputer dapat meniru kemampuan manusia dalam berkomunikasi melalui teks jarak jauh.
f. *Thinking Rationally*: Membuat pengetahuan informal menjadi formal melalui notasi logika, dengan fokus pada pemecahan masalah dalam dunia nyata.
g. *Acting Rationally*: Membuat inferensi logis sebagai bagian dari agen rasional, yang bertujuan untuk mencapai hasil yang rasional dan logis dalam tindakan

Dalam konteks deteksi penyakit seperti tuberkulosis, AI dapat digunakan untuk menganalisis gambar radiografi paru-paru guna mendeteksi tanda-tanda tuberkulosis, seperti perubahan pada paru-paru yang mencurigakan. Selain itu, AI menggunakan algoritma *deep learning* untuk memprediksi kemungkinan adanya infeksi tuberkulosis berdasarkan data riwayat medis dan hasil pemeriksaan.

1.5.4 *Deep Learning*

Deep learning adalah subbidang dari AI yang mencoba mempelajari pola-pola kompleks dalam data dengan memanfaatkan arsitektur hirarkis yang terdiri dari banyak *layer* untuk menemukan dan memahami fitur-fitur yang rumit dalam data (Guo et al., 2016). Setiap *layer* dalam jaringan *deep learning* mengambil fitur-fitur yang ditemukan oleh *layer* sebelumnya, memungkinkan untuk pembuatan hierarki fitur yang kompleks dan lebih abstrak.



Gambar 3. *Deep learning layers*
Kesse et al., 2020. Metals. https://doi.org/10.3390/met10040451

Pada Gambar 3 menunjukkan perbandingan *Simple Neural Network* (SNN) dan *Deep Neural Network* (DNN), menyoroti perbedaan jumlah *layer* tersembunyi yang ada dalam setiap proses. Gambar tersebut menunjukkan bahwa DNN memiliki struktur yang lebih dalam, dengan *layer-layer* yang ditumpuk secara bertingkat, sementara SNN terlihat lebih sederhana dengan sedikit *layer* tersembunyi. Dengan memiliki lebih banyak *layer* tersembunyi, DNN dapat mempelajari representasi yang semakin abstrak dan kompleks dari data dengan mudah (Lecun, Bengio, & Hinton, 2015). Proses ini memungkinkan DNN untuk menangkap pola yang lebih kompleks

dan menjadikan mereka lebih efektif dalam berbagai tugas klasifikasi, deteksi objek, dan pengenalan ucapan.

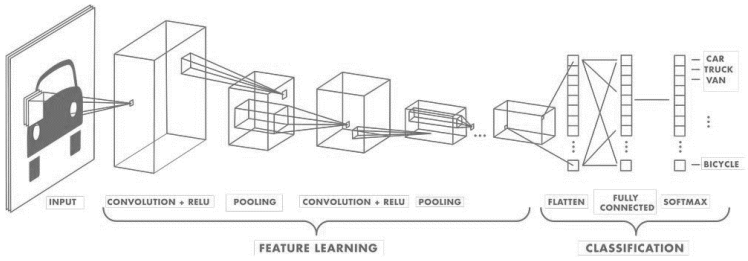
Deep learning juga dikategorikan ke dalam beberapa jenis model berdasarkan arsitektur, fungsionalitas, dan aplikasinya. Setiap jenis model memiliki algoritma dan fungsi uniknya sendiri, yang memungkinkannya unggul dalam tugas-tugas tertentu. Jenis-jenis ini meliputi:

a. *Convolutional Neural Network* (*CNN*), model ini sangat efektif untuk tugas-tugas pengenalan gambar karena kemampuannya untuk secara otomatis mempelajari fitur-fitur dari gambar. *CNN* umumnya digunakan dalam aplikasi seperti pengenalan wajah, deteksi objek, dan klasifikasi gambar.
b. *Recurrent Neural Network* (RNN), model ini dirancang untuk menangani data berurutan dan sangat berguna untuk tugas-tugas seperti pengenalan suara, pemodelan bahasa, dan pembuatan teks. RNN mampu mempertahankan kondisi internal yang menangkap informasi tentang input sebelumnya, sehingga cocok untuk tugas-tugas yang melibatkan hubungan temporal.
c. *Artificial Neural Network* (ANN), model ini adalah jenis jaringan saraf yang paling sederhana dan biasanya digunakan untuk tugas klasifikasi dan regresi. Jaringan ini tidak serumit jenis jaringan saraf lainnya, tetapi masih bisa efektif untuk banyak kasus.

Berdasarkan penjelasan diatas, *Convolutional Neural Network* (*CNN*) merupakan model *deep learning* yang paling sesuai untuk penelitian ini karena dirancang secara khusus untuk identifikasi gambar dan deteksi objek sesuai dengan tujuannya.

1.5.5 *Convolutional Neural Network*

Convolutional Neural Network (*CNN*) adalah model *deep learning* yang memproses data gambar sebagai input, menentukan bobot dan bias yang dapat dipelajari untuk berbagai elemen dalam gambar, dan berfungsi untuk mengidentifikasi serta membedakan antara objek yang berbeda (Azmi, Defit, & Sumijan, 2023).



Gambar 4. Arsitektur *convolutional neural network*
Mien et al., 2022. Journal of Robotics and Control. https://doi.org/10.18196/jrc.v3i4.14978

CNN merupakan jaringan syaraf memiliki arsitektur yang umumnya terdiri dari empat jenis *layer*, yaitu *convolution*, *pooling*, *flatten*, dan *fully connected*. Setiap jenis *layer* ini memiliki peran yang berbeda (Guo et al., 2016). Terlihat pada Gambar 4, *CNN* terdiri dari dari 2 *layer* arsitektur, yakni *feature learning* dan *classification layer*. Pada tahap *feature learning*, *convolution* dan *pooling layer* menerima input gambar dan menghasilkan *feature map* yang merepresentasikan citra dalam bentuk matriks. *Feature map* ini kemudian digunakan pada tahap *classification*, di mana beberapa *fully connected layer* menerima input dari *layer output feature learning*. Proses ini melibatkan *flatten layer* dan beberapa *hidden layer* yang memproses data untuk menghasilkan *output* berupa akurasi klasifikasi untuk setiap kelas.

a. *Convolution Layer*

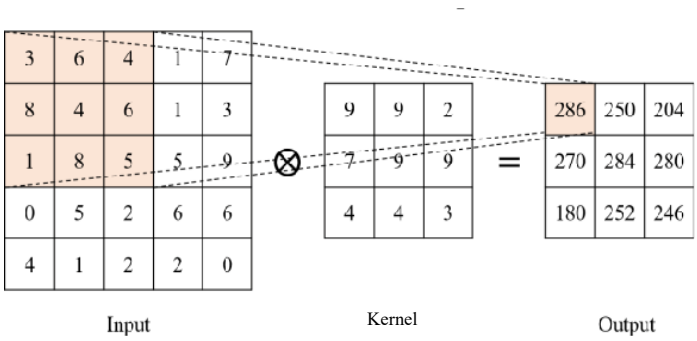
Convolution Layer merupakan *layer* yang pertama kali menerima input citra langsung pada arsitektur (Zufar, Setiyono, & Matematika, 2016). *Layer* ini bertugas untuk mengekstraksi fitur-fitur dari citra input dengan menerapkan filter atau kernel yang bergerak melintasi seluruh citra. Setiap kernel bertanggung jawab untuk mendeteksi berbagai fitur seperti tepi, tekstur, dan pola lainnya dalam citra. Tujuan dari konvolusi pada citra adalah untuk mengidentifikasi dan mengekstraksi fitur dari citra input sehingga fitur ini dapat digunakan oleh *layer* berikutnya untuk klasifikasi atau tugas lain (Azmi et al., 2023).

Penerapan konvolusi dalam *CNN* melibatkan operasi matematika di mana kernel diterapkan pada data input untuk mengekstraksi fitur. Operasi konvolusi dapat ditunjukkan pada persamaan (1).

$$s_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} I_{i+a,j+b} K_{a,b} \quad (1)$$

Dimana I adalah input dan K adalah kernel. Input *convolution layer* merupakan citra yang direpresentasikan menjadi sebuah matriks. Operasi konvolusi menghasilkan s di berbagai tempat pada *feature map*. Posisi tertentu dari s adalah hasil perkalian antara setiap nilai dalam kernel dan nilai piksel citra yang tumpang tindih dengan kernel tersebut (Khan, Rahmani, Shah, & Bennamoun, 2018). Operasi konvolusi ini menghasilkan *feature map* yang menunjukkan keberadaan dan lokasi fitur-fitur yang terdeteksi.

Dapat dilihat pada Gambar 5, setiap kernel pada *convolution layer* mengkonvolusi area kecil dari citra input, yang disebut *receptive field*, dan menghasilkan sebuah nilai dalam *feature map* (*output*). Nilai-nilai ini kemudian ditransfer ke *layer* berikutnya untuk pemrosesan lebih lanjut.

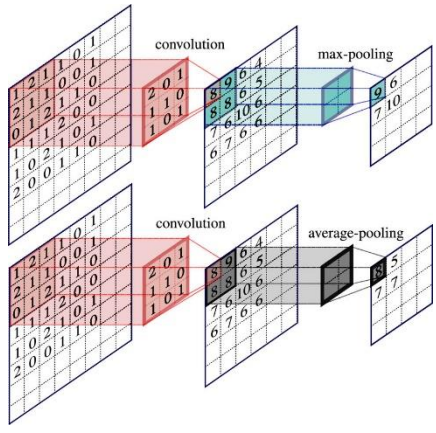


Gambar 5. Operasi dari *convolution layer*
Zhao et al., 2019. International Conference. https://doi.org/10.1145/3378891.337889

b. *Pooling Layer*

Pooling Layer adalah *layer* yang melakukan operasi *downsampling* pada *feature map* yang dihasilkan oleh *layer* sebelumnya. *convolution layer* berfungsi untuk mendeteksi fitur lokal dari citra, seperti tepi atau tekstur, dengan menggunakan kernel. Sementara itu, *pooling layer* menggabungkan fitur-fitur yang serupa untuk menyederhanakan data dan mengurangi dimensi fitur (Lecun et al., 2015).

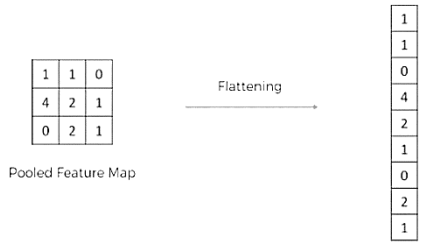
Operasi ini dapat berupa *max pooling*, *average pooling*, atau lainnya, tergantung pada jenis *pooling* yang digunakan. *Max pooling* mengambil nilai maksimum dari setiap area *pooling*, sementara *average pooling* mengambil rata-rata nilainya. Tujuan dari *downsampling* ini adalah untuk mengurangi dimensi data, sehingga mengurangi jumlah parameter yang diperlukan dalam model, dan secara efektif mengontrol *overfitting*. Untuk illustrasinya dapat dilihat pada Gambar 6.



Gambar 6. Operasi dari *max pooling layer*
Teo et al., 2021. New Journal of Physics. https://doi.org/10.1088/1367-2630/ac1fcb

c. *Flatten Layer*

Flatten merupakan metode untuk mengubah data matriks hasil dari *Pooling Layers* n-dimensi menjadi 1-dimensi (Ahmad, 2021). Proses ini diperlukan sebelum data dapat dimasukkan ke *dalam fully connected layer*. Setelah melewati beberapa *layer* konvolusi yang mengidentifikasi fitur-fitur dalam citra dan *pooling layer* yang mengurangi dimensi spasialnya, *output*nya berupa matriks atau kubus multidimensi.

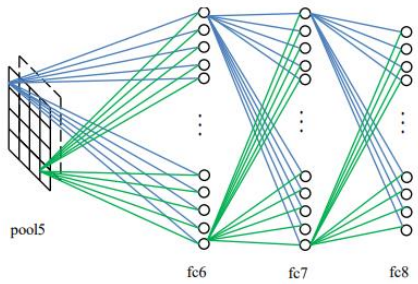


Gambar 7. Proses *flattening* dari hasil pooling pada *feature* map
Chakrabarty, 2019. Conference Paper. https://doi.org/10.1109/ICCMC.2019.8819804

Misalnya, jika *output* dari *pooling layer* adalah matriks berukuran 3 x 3 seperti pada Gambar 7, proses flattening akan mengubahnya menjadi vektor dengan panjang 3 * 3 = 9. Matriks berukuran 3 x 3 diubah menjadi vektor satu dimensi dengan panjang 9 elemen yang merepresentasikan semua nilai dari matriks tersebut secara linear. Vektor ini kemudian dimasukkan ke dalam *fully connected layer*.

d. *Fully Connected Layer*

Fully-connected layer atau lebih dikenal dengan *dense layer* merupakan bagian atau *layer* terakhir dari setiap arsitektur *CNN* yang digunakan untuk klasifikasi, di mana setiap neuron dalam satu *layer* terhubung dengan setiap neuron dari *layer* sebelumnya (Ghosh et al., 2019).



Gambar 8. Operasi dari fully connected *layer*
Guo dkk., 2016. Neurocomputing. https://doi.org/10.1016/j.neucom.2015.09.116

Layer ini bertanggung jawab untuk menggabungkan fitur-fitur yang telah diekstraksi oleh *layer-layer* sebelumnya (seperti *layer convolutional* dan *pooling*) dan menerjemahkannya ke dalam *output* yang dapat digunakan untuk membuat prediksi.

1.5.6 *Pre-trained Model*

Seperti yang kita ketahu sebelumnya, arsitektur *CNN* terdiri dari *convolution*, *pooling*, *flatten*, dan *fully connected layer*. Beberapa arsitektur yang telah dirancang berpartisipasi dalam kompetisi *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), yang menggunakan *dataset ImageNet* untuk mengevaluasi performa mereka. Arsitektur yang mencapai performa terbaik dalam ILSVRC sering dipilih sebagai *base model* atau *pre-trained model* karena memiliki akurasi yang tinggi dan kerugian validasi (*validation loss*) yang rendah (Ahmad, 2021).

Pre-trained model yang dihasilkan dari kompetisi ILSVRC sangat berharga bagi para peneliti dan praktisi. Model-model ini dapat digunakan sebagai titik awal untuk tugas-tugas yang serupa, sehingga mengurangi waktu dan sumber daya yang diperlukan untuk melatih model dari awal. *Pre-trained* model adalah sebuah arsitektur *CNN* yang telah dilengkapi dengan bobot, bias, dan algoritma yang sudah dilatih pada *dataset* besar seperti *ImageNet*, *CIFAR-10*, atau *MNIST* (Pardede & Purohita, 2023). Model ini dirancang untuk menyelesaikan berbagai tugas seperti klasifikasi citra, deteksi objek, atau keterangan citra dengan memanfaatkan fitur-fitur penting yang sudah dipelajari dari *dataset* pelatihan. Dengan memanfaatkan model *pre-trained*, proses pelatihan yang memakan waktu dan sumber daya komputasi dapat dipercepat, karena model sudah memiliki pengetahuan awal yang dapat digunakan untuk aplikasi baru atau spesifik (X. Han et al., 2021).

Dalam penelitian ini, empat arsitektur *pre-trained* yang digunakan adalah *MobileNetV3*, *EfficientNetV2B0*, *NASNetMobile*, dan *DenseNet121*. Setiap arsitektur ini memiliki karakteristik unik dan keunggulan dalam berbagai aspek seperti efisiensi komputasi, akurasi, dan kecepatan inferensi. Detail dari masing-masing arsitektur model ini dapat dilihat pada Tabel 1.

Tabel 1. Arsitektur *pre-trained model*

Arsitektur	Fitur Utama	*Top-1 Accuracy ImageNet* (%)
MobileNetV3	*Depthwise Separable Convolution, Linear Bottleneck, Hard-Swish, Platform-Aware NAS & NetAdapt*	75,2
EfficientNetV2B0	*SE Block, ConvLayer, Inverted Residual Block Combination, & Model Scaling*	78.7

Arsitektur	Fitur Utama	*Top-1 Accuracy ImageNet* (%)
NASNetMobile	*ScheduledDropPath Regularization, Flexible Scalability, & NASNet Search Space*	74.4
DenseNet121	*Dense Connectivity & Dense Block*	75.5

a. *EfficientNetV2B0*

EfficientNet adalah arsitektur *CNN* yang menggunakan pendekatan *compound scaling* untuk mengoptimalkan kedalaman, lebar, dan resolusi gambar secara bersamaan. *EfficientNetV2B0* adalah versi yang lebih baru yang menawarkan peningkatan dalam hal akurasi dan efisiensi komputasi. Pendekatan ini memungkinkan model untuk mencapai kinerja tinggi dengan jumlah parameter yang lebih sedikit. Arsitektur ini telah diimplementasikan dalam aplikasi klasifikasi citra dan menunjukkan peningkatan performa signifikan dalam hal akurasi dan efisiensi sumber daya (Kamal, Shahbudin, & Rahman, 2023).

Dibandingkan dengan arsitektur sebelumnya, *EfficientNetV2B0* memiliki perbedaan signifikan dalam hal pendekatan desain dan optimasi. *EfficientNetV1B0* menggunakan teknik *compound scaling*, yang secara bersamaan menyeimbangkan kedalaman, lebar, dan resolusi gambar untuk mencapai efisiensi model yang optimal (Tan & Le, 2019). Ini menghasilkan model dengan performa tinggi, namun bisa mengalami hambatan performa pada perangkat keras tertentu karena desain *layer* yang lebih sederhana dan berpotensi kurang optimal pada arsitektur modern. Sebaliknya, *EfficientNetV2B0* memperkenalkan beberapa inovasi seperti *Fused-MBConv layers* dan *progressive learning*, yang tidak hanya mempercepat proses *training* tetapi juga mengurangi latensi selama inferensi (Tan & Le, 2021). *Fused-MBConv* menggabungkan *convolution* dan *batch normalization* dalam satu operasi, yang secara langsung mengurangi jumlah operasi komputasi yang diperlukan, membuatnya lebih efisien pada perangkat keras modern. Selain itu, teknik *progressive learning* yang diterapkan pada V2 membantu mengurangi *overfitting*, sehingga model dapat mencapai akurasi yang lebih tinggi dengan ukuran model yang lebih kecil atau setara dibandingkan V1.

EfficientNetV2B0 dipilih karena inovasi dalam desainnya yang meningkatkan efisiensi komputasi dan akurasi. Pendekatan yang diterapkan pada *EfficientNetV2B0* memungkinkan model untuk mencapai performa tinggi dengan jumlah parameter yang lebih sedikit, membuatnya ideal untuk aplikasi yang memerlukan keseimbangan antara akurasi dan efisiensi. *Layer* dari arsitektur *EfficientNetV2B0* dapat dilihat pada Tabel 2.

Tabel 2. Arsitektur *EfficientNetV2B0*
Tan & Le, 2021. arXiv. https://arxiv.org/pdf/2104.00298

Layer	*Output Shape*	*Kernel Size*	*Strides*	*Parameters*
Input	(*None*, 224, 224, 3)	-	-	-
Conv2D	(*None*, 112, 112, 32)	3x3	2	864
BatchNormalization	(*None*, 112, 112, 32)	-	-	128
Swish Activation	(*None*, 112, 112, 32)	-	-	-
MBConv Block	(*None*, 112, 112, 32)	3x3 *depthwise*	1	864
MBConv Block	(*None*, 56, 56, 64)	3x3 *depthwise*	2	3,584
MBConv Block	(*None*, 28, 28, 128)	3x3 *depthwise*	2	12,832
MBConv Block	(*None*, 14, 14, 256)	3x3 *depthwise*	2	49,216
MBConv Block	(*None*, 14, 14, 256)	3x3 *depthwise*	1	72,832
MBConv Block	(*None*, 7, 7, 512)	3x3 *depthwise*	2	205,056
Global Average Pooling	(*None*, 512)	-	-	-
Dense	(*None*, 1280)	-	-	656,640
Dense	(*None*, 1000)	-	-	1,281,000
Output (Softmax)	(*None*, 1000)	-	-	-

b. *MobileNetV3*

MobileNetV3 adalah arsitektur *CNN* yang dirancang untuk perangkat dengan sumber daya terbatas. Arsitektur ini menggunakan inovasi seperti *Hard-Swish* dan *NetAdapt* untuk mengoptimalkan arsitektur *CNN* pada perangkat *mobile* serta *Platform-Aware NAS* memastikan model dioptimalkan untuk perangkat keras tertentu. *MobileNetV3* telah digunakan dalam berbagai aplikasi *mobile* dan menunjukkan efisiensi tinggi dalam penggunaan daya dan komputasi (Howard et al., 2019).

Arsitektur sebelumnya (*MobileNetV2)* mengandalkan teknik *inverted residuals* dan *linear bottlenecks* untuk mencapai efisiensi komputasi, dengan fokus pada peningkatan efisiensi *layer* dan pengurangan jumlah parameter (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018). Meskipun efisien, *MobileNetV2* dapat menghadapi batasan dalam hal akurasi dan kecepatan inferensi pada perangkat dengan sumber daya sangat terbatas. Sebaliknya, *MobileNetV3* memperkenalkan *Hard-Swish*, sebuah fungsi aktivasi non-linear yang lebih efisien dibandingkan dengan fungsi ReLU atau Swish pada perangkat *mobile* (Howard et al., 2019). Selain itu, *MobileNetV3* menggunakan *NetAdapt*, metode optimasi yang memungkinkan penyesuaian arsitektur untuk kebutuhan spesifik perangkat keras, serta *Platform-Aware NAS* untuk memastikan model dioptimalkan secara khusus untuk perangkat yang digunakan. Inovasi ini memungkinkan *MobileNetV3* untuk mencapai kecepatan inferensi yang lebih tinggi dan penggunaan memori yang lebih efisien dibandingkan *MobileNetV2*. Selain itu, *MobileNetV3* telah terbukti memberikan performa yang baik pada berbagai tugas klasifikasi gambar dengan komputasi yang minimal.

Oleh karena itu, *MobileNetV3* dipilih karena efisiensinya dalam hal kecepatan dan penggunaan memori, menjadikannya ideal untuk aplikasi real-time dan perangkat dengan daya terbatas. *Layer* dari arsitektur *MobileNetV3* dapat dilihat pada Tabel 3.

Tabel 3. Arsitektur *MobileNetV3*
Howard et al., 2019. ArXiv. https://arxiv.org/pdf/1905.02244

Layer	*Output Shape*	*Kernel Size*	*Strides*	*Parameters*
Input	(*None*, 224, 224, 3)	-	-	-
Conv2D	(*None*, 112, 112, 32)	3x3	2	864
Depthwise Conv2D	(*None*, 112, 112, 32)	3x3	1	864
Pointwise Conv2D	(*None*, 112, 112, 64)	1x1	1	2,112
Depthwise Conv2D	(*None*, 56, 56, 64)	3x3	2	5,760
Pointwise Conv2D	(*None*, 56, 56, 128)	1x1	1	8,320
Depthwise Conv2D	(*None*, 28, 28, 128)	3x3	2	22,464
Pointwise Conv2D	(*None*, 28, 28, 128)	1x1	1	16,384
Depthwise Conv2D	(*None*, 14, 14, 128)	3x3	2	28,224
Pointwise Conv2D	(*None*, 14, 14, 256)	1x1	1	33,024
Depthwise Conv2D	(*None*, 7, 7, 256)	3x3	2	72,576
Pointwise Conv2D	(*None*, 7, 7, 512)	1x1	1	131,584
Global Average Pooling	(*None*, 512)	-	-	-
Dense	(*None*, 1280)	-	-	656,640
Dense	(*None*, 1000)	-	-	1,281,000
Output (Softmax)	(*None*, 1000)	-	-	-

c. *NASNetMobile*

NASNetMobile adalah arsitektur *CNN* yang dihasilkan dari proses *Neural Architecture Search* (NAS), di mana arsitektur dioptimalkan secara otomatis menggunakan algoritma pencarian. *NASNetMobile* dirancang untuk perangkat *mobile* dan memiliki kinerja yang sangat baik pada berbagai *dataset benchmark*. Pendekatan ini memungkinkan penemuan arsitektur optimal tanpa intervensi manusia yang signifikan, membuatnya sangat fleksibel untuk berbagai tugas klasifikasi. Arsitektur *NASNetMobile* mencapai keseimbangan optimal antara akurasi dan efisiensi komputasi, dan menunjukkan kinerja unggul pada perangkat *mobile* (Tan, Chen, Pang, Vasudevan, & Le, 2018).

Dibandingkan dengan arsitektur sebelumnya, *NASNet* yang dirancang untuk perangkat yang lebih kuat, *NASNetMobile* berfokus pada pengoptimalan arsitektur untuk perangkat dengan sumber daya terbatas. *NASNet* menggunakan *Neural Architecture Search* untuk menemukan blok arsitektur terbaik, yang kemudian diperbesar untuk menghasilkan model dengan performa unggul pada *dataset* yang lebih besar dan komputasi yang lebih intensif. Namun, model ini sering kali tidak optimal untuk perangkat mobile karena kebutuhan komputasi dan penggunaan memori yang lebih besar. Sedangkan *NASNetMobile* dirancang khusus untuk perangkat mobile, mengadopsi arsitektur dari *NASNet* tetapi dengan skala yang lebih

kecil dan efisiensi yang ditingkatkan. Arsitektur ini mengoptimalkan keseimbangan antara akurasi dan efisiensi komputasi dengan mengurangi kompleksitas model tanpa mengorbankan performa signifikan. Proses *Neural Architecture Search* yang digunakan memungkinkan penyesuaian arsitektur secara otomatis untuk memenuhi kebutuhan perangkat dengan daya dan sumber daya komputasi yang terbatas.

NASNetMobile dipilih karena kemampuannya dalam menemukan arsitektur optimal tanpa intervensi manusia yang signifikan, serta kinerja tinggi dengan efisiensi komputasi yang baik. Ini menjadikannya pilihan yang baik untuk tugas-tugas klasifikasi gambar yang membutuhkan fleksibilitas dan adaptabilitas. *Layer* dari arsitektur *NASNetMobile* dapat dilihat pada Tabel 4.

Tabel 4. Arsitektur *NASNetMobile*
Zoph et al., 2017. ArXiv. https://arxiv.org/pdf/1707.07012

Layer	***Output Shape***	***Kernel Size***	***Strides***	***Parameters***
Input	(*None*, 224, 224, 3)	-	-	-
Conv2D	(*None*, 112, 112, 32)	3x3	2	864
Conv2D	(*None*, 112, 112, 32)	3x3	1	9,056
Conv2D	(*None*, 56, 56, 80)	3x3	1	18,560
Conv2D	(*None*, 56, 56, 80)	3x3	2	13,760
Conv2D	(*None*, 28, 28, 160)	3x3	1	32,320
Conv2D	(*None*, 28, 28, 160)	3x3	2	23,360
Conv2D	(*None*, 14, 14, 320)	3x3	1	128,320
Conv2D	(*None*, 14, 14, 320)	3x3	2	80,320
Conv2D	(*None*, 7, 7, 640)	3x3	1	640,320
Conv2D	(*None*, 7, 7, 640)	3x3	2	384,320
Global Average Pooling	(*None*, 640)	-	1	-
Dense	(*None*, 1000)	-	-	641,000
Output (Softmax)	(*None*, 1000)	-	-	-

d. *DenseNet121*

DenseNet (*Dense Convolutional Network*) adalah jenis arsitektur *CNN* yang menghubungkan setiap *layer* ke setiap *layer* lainnya dengan cara *feed-forward*. Setiap *layer* menerima input dari semua *layer* sebelumnya dan mengirimkan *output*nya ke semua *layer* berikutnya. Pendekatan ini membantu mengatasi masalah gradien yang menghilang dan memungkinkan penggunaan parameter yang lebih efisien. *DenseNet121* telah terbukti efektif dalam berbagai tugas klasifikasi gambar termasuk dalam bidang medis seperti deteksi penyakit melalui citra rontgen. *DenseNet121* telah diterapkan untuk klasifikasi penyakit COVID-19 menggunakan citra rontgen dengan akurasi yang sangat tinggi seperti yang dibuktikan dalam penelitian oleh Ezzat et al., 2020.

Dibandingkan dengan versi sebelumnya, seperti *DenseNet169* atau *DenseNet201*, *DenseNet121* menonjol karena keseimbangan antara jumlah

parameter dan akurasi yang dicapai. Meskipun *DenseNet121* memiliki jumlah *layer* yang lebih sedikit dibandingkan dengan *DenseNet169* atau *DenseNet201*, arsitektur ini tetap mampu mempertahankan performa yang tinggi sambil mengurangi kompleksitas komputasi. Versi sebelumnya, seperti *DenseNet201*, mungkin menawarkan sedikit peningkatan dalam hal akurasi pada *dataset* yang sangat kompleks karena jumlah *layer* yang lebih banyak, namun ini juga meningkatkan kebutuhan komputasi dan memori secara signifikan. *DenseNet121* dipilih dalam banyak aplikasi medis karena efisiensinya dalam penggunaan parameter dan kemampuannya untuk mengatasi masalah gradien yang menghilang, sambil tetap memberikan akurasi yang sangat tinggi.

DenseNet121 dipilih karena kemampuannya dalam mengatasi masalah gradien yang menghilang dan penggunaan parameter yang lebih efisien. Arsitektur ini juga cenderung lebih baik dalam memanfaatkan fitur-fitur yang dipelajari sebelumnya, yang dapat meningkatkan akurasi model. *Layer* dari arsitektur *DenseNet121* dapat dilihat pada Tabel 5.

Tabel 5. Arsitektur *DenseNet121*
Huang et al., 2016. ArXiv. https://arxiv.org/pdf/1608.06993

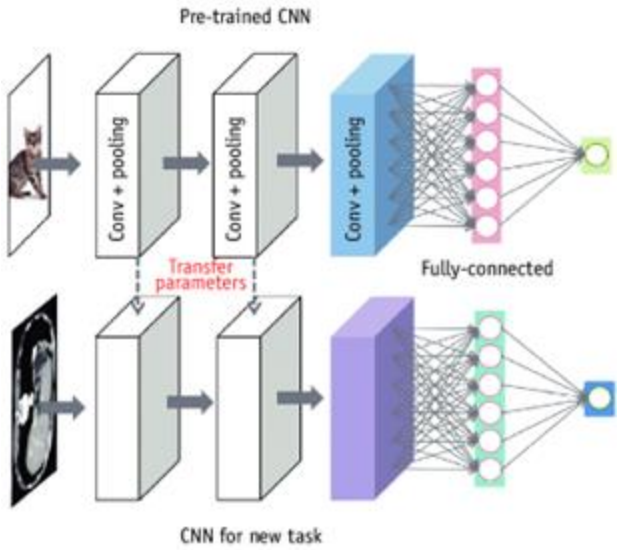
Layer	***Output Shape***	***Kernel Size***	***Strides***	***Parameters***
Input	(*None*, 224, 224, 3)	-	-	-
Conv2D	(*None*, 112, 112, 64)	7x7	2	9,472
BatchNormalization	(*None*, 112, 112, 64)	-	-	256
ReLU Activation	(*None*, 112, 112, 64)	-	-	-
MaxPooling2D	(*None*, 56, 56, 64)	3x3	2	-
Dense Block 1	(*None*, 56, 56, 256)	-	-	66,048
Transition Layer 1	(*None*, 28, 28, 128)	1x1	1	33,536
Dense Block 2	(*None*, 28, 28, 512)	-	-	257,152
Transition Layer 2	(*None*, 14, 14, 256)	1x1	1	131,072
Dense Block 3	(*None*, 14, 14, 1024)	-	-	1,017,600
Transition Layer 3	(*None*, 7, 7, 512)	1x1	1	410,368
Dense Block 4	(*None*, 7, 7, 1024)	-	-	1,679,360
Global Average Pooling	(*None*, 1024)	-	-	-
Dense	(*None*, 1000)	-	-	1,025,000
Output (Softmax)	(*None*, 1000)	-	-	-

1.5.7 *Transfer Learning*

Transfer learning adalah teknik *machine learning* yang memungkinkan penggunaan kembali model yang sudah dilatih (*pre-trained model*) untuk suatu tugas sebagai titik awal dalam pengembangan model untuk tugas-tugas lain yang serupa (Pahlevi, 2023). Teknik ini memanfaatkan pengetahuan yang telah diperoleh model dari tugas awal untuk mempercepat dan meningkatkan kinerja pada tugas baru. Dengan

demikian, *transfer learning* menghemat waktu dan sumber daya, karena model tidak perlu dilatih dari awal dan dapat bekerja dengan *dataset* yang lebih kecil.

Gambar 9 menunjukkan bagaimana *transfer learning* bekerja. Pertama, sebuah model dilatih menggunakan data pertama untuk menyelesaikan tugas pertama dan menghasilkan hasil. Setelah itu, model yang sama digunakan lagi untuk tugas kedua, tetapi hanya bagian akhirnya yang dilatih ulang dengan data kedua untuk menghasilkan hasil baru. Dengan cara ini, pengetahuan dari tugas pertama digunakan kembali untuk tugas kedua, menghemat waktu dan usaha dalam melatih model baru dari awal.



Gambar 9. Alur kerja *transfer learning*
Do et al., 2020. Korean Journal of Radiology. https://doi.org/10.3348/kjr.2019.0312

Salah satu alasan utama penggunaan *transfer learning* adalah efisiensinya dalam hal waktu dan sumber daya, karena model tidak perlu dilatih dari nol. Model yang sudah dilatih pada *dataset* besar dan serupa juga cenderung memiliki representasi fitur yang baik, sehingga mengurangi risiko *overfitting* dan menghasilkan performa yang lebih baik setelah *fine-tuning* pada *dataset* baru. *Transfer learning* sangat berguna ketika *dataset* yang dimiliki kecil, masalah yang akan diselesaikan mirip dengan tugas yang digunakan untuk melatih model sebelumnya, atau ketika ada kebutuhan untuk mengembangkan model dengan cepat dan dengan sumber daya komputasi yang terbatas. Dengan memanfaatkan pengetahuan yang sudah ada dalam model *pre-trained*, *transfer learning* memungkinkan pengembangan model yang lebih cepat dan dengan performa yang tinggi dalam berbagai aplikasi, termasuk klasifikasi gambar dan pemrosesan bahasa alami (Pan & Yang, 2010).

1.5.8 Evaluasi Model

Evaluasi model berguna untuk menguji model yang sudah dilatih serta membantu menemukan pengaturan terbaik untuk model. Evaluasi model juga membantu mengidentifikasi apakah model terlalu fokus pada data pelatihan (*overfitting*) atau terlalu umum (*underfitting*).

a. *Confusion Matrix*

Confusion Matrix adalah sebuah tabel yang digunakan untuk mengukur performa suatu model klasifikasi dalam *machine learning*. Tabel ini membandingkan nilai prediksi dengan nilai aktual, sehingga memungkinkan analisis yang lebih rinci dan akurat tentang keberhasilan model dalam memprediksi kelas yang tepat.

Tabel 6. *Confusion Matrix*

Aktual	Prediksi	
	Positive	***Negative***
Positive	TP	FN
Negative	FP	TN

Dalam Tabel 6 terdapat empat jenis hasil yang penting: *True Positive* (TP), *True Negative* (TN), *False positive* (FP), dan *False Negative* (FN). TP dan TN memberikan informasi ketika pengklasifikasi benar, sementara FP dan FN memberi tahu ketika pengklasifikasi salah (J. Han et al., 2011).

b. Akurasi

Akurasi (*accuracy*) adalah rasio keberhasilan model dalam menebak benar semua kelas dari setiap kasus yang diberikan (Hartato, 2021). Secara intuitif, semakin tinggi nilai akurasi, semakin baik model dalam memprediksi label yang benar. Secara matematis, akurasi didefinisikan dalam persamaan (2).

$$Akurasi = \frac{TP + TN}{TP + FP + TN + LN} \quad (2)$$

Jika akurasi suatu model untuk mendeteksi COVID-19 dari citra rontgen adalah 92%, ini berarti bahwa dari 100 orang yang diuji, model berhasil membuat prediksi yang benar (baik positif maupun negatif) untuk 92 orang. Sisa 8 orang lainnya salah didiagnosis, baik sebagai FP maupun FN.

c. Presisi

Presisi (*precision*) adalah rasio prediksi benar positif (TP) dibandingkan dengan keseluruhan hasil yang diprediksi positif. Presisi menunjukkan seberapa banyak dari prediksi positif model yang benar-benar positif. Presisi yang tinggi berarti bahwa

ketika model memprediksi positif, prediksi tersebut cenderung benar. Secara matematis, presisi didefinisikan dalam persamaan (3).

$$Presisi = \frac{TP}{TP + FP} \tag{3}$$

Jika presisi suatu model untuk mendeteksi COVID-19 dari citra rontgen adalah 85%, ini berarti bahwa dari 100 orang yang diprediksi positif COVID-19 oleh model, 85 orang memang benar-benar positif COVID-19, sementara 15 orang lainnya sebenarnya negatif.

d. *Recall*

Recall atau Sensitivitas menunjukkan bahwa proporsi kasus nyata positif yang diprediksi dengan benar sebagai positif (Mubin, 2021). Nilai *recall* yang tinggi menunjukkan bahwa model memiliki kemampuan yang baik dalam mendeteksi semua kasus yang seharusnya diklasifikasikan sebagai positif. Secara matematis, *recall* didefinisikan dalam persamaan (4).

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Jika sensitivitas suatu model untuk mendeteksi COVID-19 dari citra rontgen adalah 90%, ini berarti bahwa dari 100 orang yang benar-benar terkena COVID-19, model berhasil mendeteksi 90 orang dengan benar.

e. *F1-Score*

F1-Score adalah indikator yang menunjukkan bahwa model yang dibuat memiliki nilai presisi dan *recall* yang baik (Clara et al., 2021). *F1 Score* memberikan satu nilai komprehensif yang mencerminkan keseimbangan antara kemampuan model untuk mendeteksi kasus positif yang sebenarnya dan menghindari kesalahan positif palsu. Secara matematis, *F1-Score* didefinisikan dalam persamaan (5).

$$F1\ score = 2 \times \frac{presisi \times recall}{presisi + recall} \tag{5}$$

Jika *F1-Score* suatu model untuk mendeteksi COVID-19 dari citra rontgen adalah 88%, ini berarti bahwa model memiliki keseimbangan antara presisi dan sensitivitas. Artinya, model ini cukup baik dalam mendeteksi kasus positif (sensitivitas) dan juga cukup akurat dalam memastikan bahwa prediksi positifnya benar (presisi).

f. Spesifisitas

Spesifisitas adalah kemampuan model untuk dengan benar mengidentifikasi semua kasus negatif dalam *dataset*. Spesifisitas mengukur proporsi dari semua contoh negatif yang benar-benar diklasifikasikan sebagai negatif oleh model. Secara matematis, spesifisitas didefinisikan dalam persamaan (6).

$$Spesifisitas = \frac{TN}{TN + FP} \quad (6)$$

Contohnya jika spesifisitas suatu model untuk mendeteksi COVID-19 adalah 85%, ini berarti bahwa dari 100 orang yang tidak terkena COVID-19, model berhasil mengidentifikasi 85 orang dengan benar sebagai tidak terkena.

1.5.9 *Streamlit*

Streamlit adalah framework *Python* yang dirancang untuk memudahkan pembuatan aplikasi web dengan cepat dan mudah tanpa memerlukan pengetahuan mendalam tentang pengembangan web. *Framework* ini memungkinkan para *data scientist* dan pengembang untuk mengubah skrip data menjadi aplikasi web interaktif hanya dengan beberapa baris kode.

Streamlit sangat berguna bagi peneliti dan *data scientist* untuk membagikan hasil penelitian dan data mereka secara interaktif (Nápoles-Duarte et al., 2022). Misalnya, *Streamlit* telah digunakan untuk membangun platform kepercayaan kecerdasan buatan untuk jaringan nirkabel generasi berikutnya. *Framework* ini memungkinkan peneliti untuk mengevaluasi, mempertahankan, mengesahkan, dan memverifikasi model dan aplikasi AI mereka terhadap ancaman adversarial seperti penghindaran, peracunan, ekstraksi, dan gangguan, sehingga meningkatkan kinerja jaringan dan keamanan siber (Kuzlu, Catak, Sarp, Cali, & Gueler, 2022).

1.5.10 *TensorFlow*

TensorFlow adalah sebuah *framework open-source* yang dikembangkan oleh *Google Brain* untuk keperluan *machine learning* dan *deep learning*. *TensorFlow* mempermudah proses pembangunan, pelatihan, dan penerapan model *machine learning* dalam skala yang luas. *Framework* ini dikenal karena fleksibilitasnya dan dukungannya terhadap berbagai platform, mulai dari perangkat *mobile* hingga sistem terdistribusi. *TensorFlow* melakukan pekerjaan yang luar biasa dalam pengenalan pola, terutama dalam konteks gambar, suara, ucapan, bahasa, dan data deret waktu (Manaswi, 2018).

TensorFlow menyediakan *library Keras*, yaitu API tingkat tinggi yang terintegrasi dengan *TensorFlow* untuk mempermudah pembangunan dan pelatihan model *neural*

network. Dalam ekosistem *Keras*, terdapat beberapa *library* dan modul yang menyediakan berbagai fungsi dan fitur untuk memudahkan pengembangan model *machine learning*. Beberapa *library* dan komponen utama yang tersedia dalam *Keras* dapat dilihat pada Tabel 7.

Tabel 7. *Keras TensorFlow library*
Source: https://*Keras*.io/api/

Library	**Deskripsi**
Keras.*layer*s	Modul untuk membuat *layer neural network*, seperti *Dense*, *Conv2D*, *LSTM*, dan lain-lain.
Keras.models	Menyediakan kelas untuk mendefinisikan dan mengelola model, termasuk model *Sequential* dan *Functional API*.
Keras.metrics	Modul untuk mengukur kinerja model selama pelatihan dan evaluasi, seperti *accuracy*, *precision*, *recall*.
Keras.callbacks	Alat untuk mengimplementasikan tindakan yang dilakukan pada titik tertentu selama pelatihan model, seperti *EarlyStopping*, *ModelCheckpoint*.
Keras.preprocessing	Menyediakan alat untuk *preprocessing* data gambar, teks, dan urutan (*sequences*) sebelum memasukkan ke model.
Keras.activations	Fungsi aktivasi yang digunakan dalam *layer neural network*, seperti *relu*, *sigmoid*, *softmax*.
Keras.utils	Berbagai alat bantu utilitas, seperti fungsi untuk mengonversi label menjadi *one-hot encoding*, atau plot model arsitektur.
Keras.regularizers	Alat untuk menerapkan regularisasi pada bobot model, seperti L1, L2, dan *dropout regularization*.
Keras.losses	Menyediakan berbagai fungsi loss yang digunakan dalam pelatihan model, seperti *mean_squared_error*, *categorical_crossentropy*.
Keras.applications	Koleksi *pre-trained model* yang dapat digunakan untuk transfer learning, seperti *VGG16, ResNet, InceptionV3*.

1.5.11 *Hyperparameter*

Hyperparameter adalah parameter yang mengontrol proses pelatihan model dalam *deep learning*. Untuk membangun model *deep learning* yang optimal, diperlukan konfigurasi *hyperparameter* yang tepat. Menyempurnakan *hyperparameter* ini memerlukan pengalaman, intuisi, dan pemahaman yang mendalam tentang model, karena hasil yang memuaskan hanya bisa dicapai dengan mengatur kombinasi nilai *hyperparameter* secara cermat (Iqbal, Qureshi, Ullah, Li, & Mahmood, 2022). Beberapa *hyperparameter* yang perlu untuk diketahui dalam deep learning meliputi:

a. *Learning Rate*, untuk mengontrol seberapa besar perubahan yang dilakukan pada bobot model setelah setiap iterasi pelatihan. *Learning rate* yang terlalu tinggi bisa menyebabkan model tidak stabil, sedangkan yang terlalu rendah bisa memperlambat konvergensi.
b. Jumlah *Epoch*. yaitu jumlah total iterasi yang digunakan untuk melatih model. *Epoch* yang terlalu sedikit bisa membuat model *underfitting*, sementara *epoch* yang terlalu banyak bisa menyebabkan *overfitting*.
c. *Batch Size*, yaitu jumlah contoh data yang diproses sebelum model diperbarui. Ukuran *batch* yang kecil dapat mempercepat pelatihan tetapi dengan fluktuasi yang lebih besar dalam pembaruan model, sedangkan ukuran batch yang besar bisa stabil tetapi lebih memakan memori.
d. *Network Architecture*, yaitu jumlah lapisan tersembunyi dan jumlah neuron di setiap lapisan yang mempengaruhi kapasitas model. Struktur yang terlalu sederhana bisa membuat model tidak cukup kuat, sementara struktur yang terlalu kompleks bisa menyebabkan *overfitting*.
e. *Activation Function*, yaitu fungsi yang menentukan output dari setiap neuron dalam jaringan. Fungsi aktivasi seperti *ReLU*, *Sigmoid*, dan *Tanh* masing-masing memiliki kelebihan dan kekurangan tergantung pada masalah yang dihadapi.
f. *Regularization*, suatu teknik seperti *Dropout* dan *L2 Regularization* yang digunakan untuk mencegah *overfitting* dengan menambahkan penalti pada bobot model atau mengabaikan beberapa neuron selama pelatihan.
g. *Optimizer*: Algoritma yang digunakan untuk memperbarui bobot model selama pelatihan, seperti *Stochastic Gradient Descent* (SGD), *Adam*, atau *RMSprop*. Pilihan optimizer mempengaruhi kecepatan konvergensi dan kualitas akhir model.

BAB II

Metode Penelitian

2.1 Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan dari bulan April sampai dengan Juli 2024.

Tabel 8. Waktu dan tahap penelitian

No.	Kegiatan	2024															
		April				Mei				Juni				Juli			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Pengajuan Topik																
2	Studi Literatur																
3	Pengajuan Judul																
4	Pengumpulan Data																
5	Eksplorasi Data																
6	Penyusunan Draft Proposal (BAB I – III)																
7	Bimbingan dan Revisi																
9	Olah dan Analisis Data																
10	Implementasi dan Pengujian Model																
11	Penyusunan Laporan Akhir																
12	Seminar Hasil																

Tempat penelitian dilakukan di Laboratorium Rekayasa Perangkat Lunak (RPL) Prodi Sistem Informasi, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Kota Makassar, Indonesia.

2.2 Instrumen Penelitian

Adapun instrumen penelitian yang digunakan terdiri dari perangkat keras (*hardware*) pada Tabel 9 dan perangkat lunak (software) pada Tabel 10.

Tabel 9. Instrumen perangkat keras

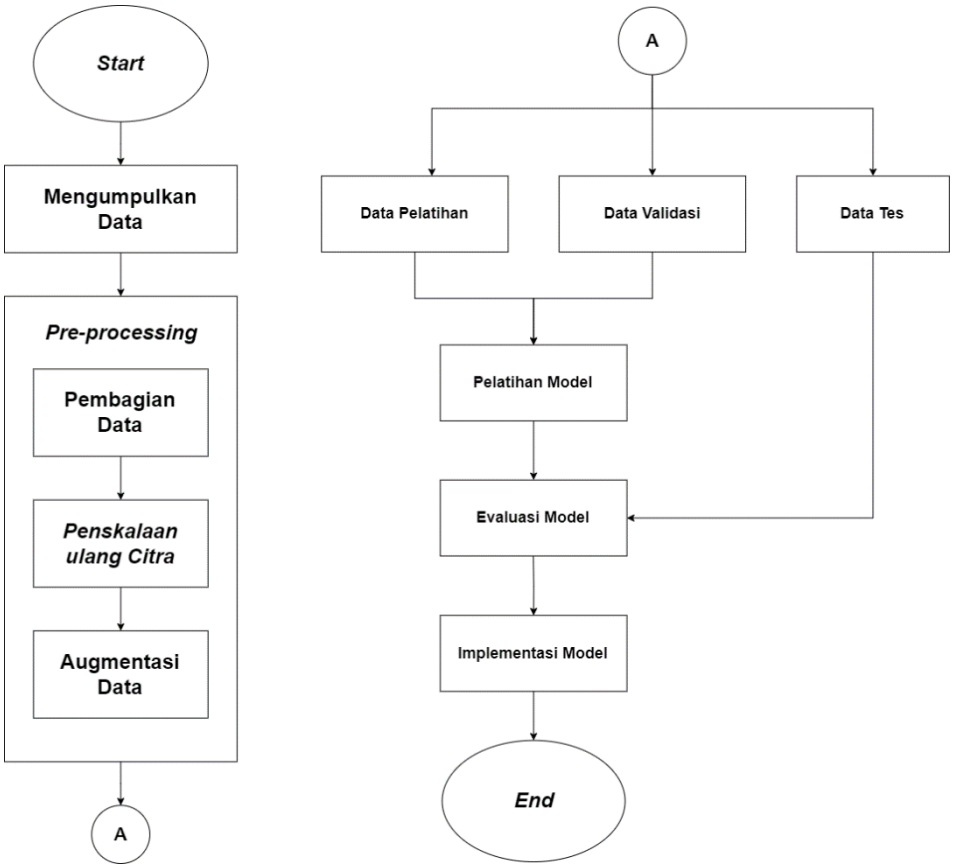
Hardware	**Spesifikasi**
Operating System (OS)	Windows 11 Home
Processor	AMD Ryzen 3 5300U
Memory	8 GB DDR4-3200
Storage	512gb SSD M.2
GPU	AMD Vega 7

Tabel 10. Instrumen perangkat lunak

Software	**Keterangan**
Python	Bahasa pemrograman utama
Visual Studio Code	Digunakan sebagai editor kode
Jupyter Notebook	Aplikasi web open-source yang digunakan untuk menulis dan menjalankan kode dan analisis data.
Streamlit	*Framework* yang digunakan dalam membangun aplikasi web untuk menampilkan hasil klasifikasi.
NumPy	Digunakan untuk komputasi ilmiah dan operasi array
Scikit-learn	Modul untuk mengukur kinerja model
Matplotlib	Digunakan untuk membuat visualisasi data seperti grafik dan plot
Keras TensorFlow	Modul *Keras* yang disertakan dalam *TensorFlow* untuk membangun dan melatih model *deep learning*
splitfolders	Library untuk membagi *dataset* menjadi folder *train*, *validation*, dan *test*.

2.3 Tahap Penelitian

Tahapan penelitian yang akan diambil untuk menyelesaikan penelitian ini meliputi *collecting data*, *pre-processing data,* pelatihan model, evaluasi model*,* dan implementasi model. Ini direpresentasikan dalam Gambar 10 yaitu *flowchart* yang menggambarkan alur kerja penelitian.



Gambar 10. Diagram alur penelitian

Keterangan:

a. **Mengumpulkan Data**, ini merupakan tahap pengumpulan data yang akan digunakan dalam pelatihan model.
b. ***Pre-processing***, proses ini bertujuan untuk mempersiapkan dan mengubah data mentah menjadi format yang dapat digunakan secara optimal oleh model. Berikut adalah komponen-komponen dari pre-processing yang ada pada flowchart:
   - Pembagian Data, data yang telah dikumpulkan dibagi menjadi beberapa subset: data *train* (pelatihan), *data validation* (validasi), dan *data test* (pengujian). Pembagian ini bertujuan untuk memastikan bahwa model dilatih, divalidasi, dan diuji dengan data yang terpisah agar hasil evaluasi lebih objektif.
   - Penyesuian ulang Citra, citra-citra dalam *dataset* diubah ukurannya menjadi ukuran yang konsisten, biasanya ke dimensi yang diperlukan

oleh arsitektur model. Penyesuian ulang ini penting untuk memastikan semua data gambar sesuai dengan input yang diharapkan oleh model.

- Augmentasi Data*,* yaitu proses memperbesar ukuran *dataset* secara efektif dengan membuat variasi baru dari data gambar yang ada. Augmentasi membantu meningkatkan generalisasi model dengan membuatnya lebih robust terhadap variasi pada data.

c. **Data Pelatihan,** data yang diambil dari data yang telah diproses digunakan untuk melatih model. Data ini digunakan untuk mengoptimalkan bobot model selama proses pelatihan.
d. **Data Validasi,** data yang digunakan untuk memvalidasi model selama pelatihan. Validasi dilakukan untuk mengukur kinerja model pada data yang tidak terlihat selama pelatihan, membantu dalam *tuning hyperparameter* dan mencegah *overfitting*.
e. **Data Tes,** data yang digunakan untuk menguji model setelah pelatihan selesai. Data ini sepenuhnya baru bagi model dan digunakan untuk mengevaluasi kinerja akhir dari model secara objektif.
f. **Pelatihan Model***,* yaitu proses pelatihan model menggunakan data pelatihan. Selama tahap ini, model dbelajar dari data dan menyesuaikan bobotnya untuk meminimalkan kesalahan prediksi.
g. **Evaluasi Model***,* yaitu proses evaluasi yang menggunakan data validasi untuk memantau kinerja selama pelatihan dan data tes untuk mendapatkan metrik kinerja akhir. Evaluasi ini mencakup pengukuran seperti akurasi, precision, *recall*, dan lain-lain untuk menentukan seberapa baik model melakukan tugasnya.
h. **Implementasi Model***,* setelah model dievaluasi dan memenuhi kriteria kinerja yang diharapkan, model tersebut diterapkan untuk digunakan dalam aplikasi nyata.

2.4 *Dataset*

Dataset yang digunakan dalam penelitian ini berasal dari berbagai sumber, termasuk National *Library of Medicine* (NLM) yang menyediakan *dataset Montgomery* dan *Shenzhen*, serta *dataset Belarus* yang dikumpulkan untuk studi resistensi obat oleh *National Institute of Allergy and Infectious Diseases*. Selain itu, *dataset RSNA pneumonia detection challenge* juga dimanfaatkan.

Dataset ini terdiri dari *Chest X-ray* (rontgen dada) Normal (3500) dan citra pasien dengan TBC (700). Seluruh *dataset* penelitian ini dapat diakses melalui kaggle.com.

2.5 *Pre-processing*

Dalam penelitian ini, dilakukan *pre-processing* data pada *dataset* yang sebelumnya telah dikumpulkan untuk tujuan identifikasi penyakit Tuberkulosis (TBC) pada citra

rontgen dada. *Dataset* awal, yang tersimpan dalam satu folder, dibagi secara proporsional menjadi tiga subset: data pelatihan, data pengujian, data validasi. Selanjutnya, *dataset* yang sudah dibagi dimuat menggunakan fungsi yang diambil dari *TensorFlow*, dengan mengatur dimensi citra dan ukuran batch. *Dataset* diproses dalam mode label *"categorical".* Berikutnya, untuk meningkatkan generalisasi model dan mengurangi kemungkinan *overfitting*, augmentasi data diterapkan pada *dataset* pelatihan. Dengan langkah-langkah ini, *dataset* telah disiapkan dan siap untuk digunakan dalam pelatihan, validasi, dan evaluasi model klasifikasi citra TBC selanjutnya.

2.6 Pelatihan Model

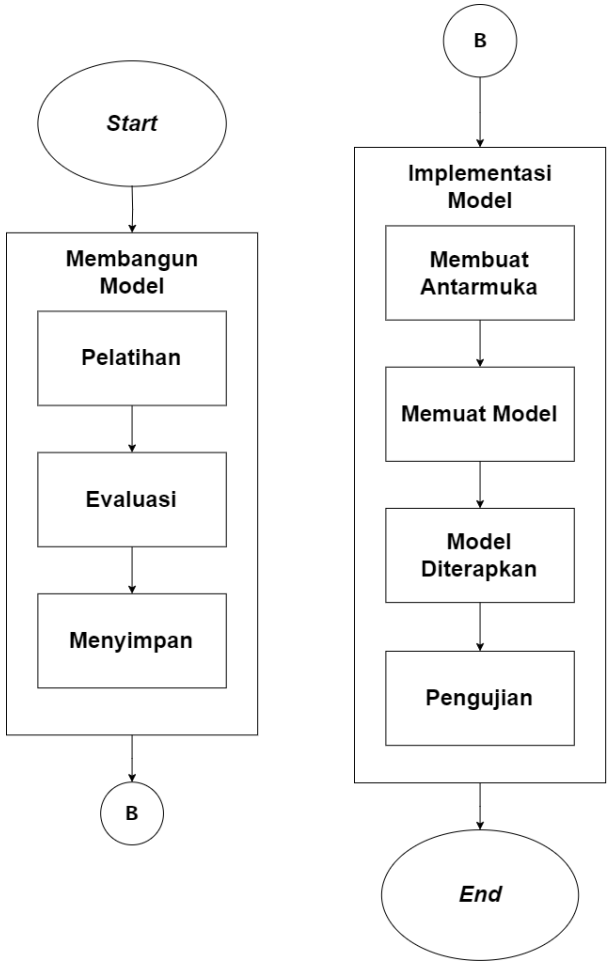
Pada penelitian ini, model dilatih dengan menggunakan arsitektur *EfficientNetV2B0, MobileNetV3, NASNetMobile,* dan *DenseNet121* dengan data pelatihan dan validasi setelah proses *data pre-processing.* Arsitektur-arsitektur ini dipilih karena memiliki keunggulan berupa kinerja tinggi dan latensi rendah dengan bobot yang ringan. *Framework Keras* digunakan untuk memfasilitasi proses pelatihan, dengan memanfaatkan bobot *pre-trained* dari *ImageNet.*

2.7 Evaluasi Model

Model yang dihasilkan dari proses pelatihan akan menghasilkan sejumlah metrik kinerja yang kemudian dievaluasi, termasuk melihat *Confusion Matrix*, akurasi, presisi, *recall*, dan *F1-Score*. Model yang sebelumnya menggunakan berbagai arsitektur berbeda kemudian dibandingkan. Arsitektur dengan kinerja model terbaik akan diimplementasikan ke dalam aplikasi web.

2.8 Implementasi Model

Setelah proses evaluasi model selesai dan model dengan kinerja terbaik telah diidentifikasi, tahap selanjutnya adalah mendistribusikan model tersebut agar dapat diintegrasikan ke dalam sistem informasi yang sesuai. Dalam konteks ini, kami akan menggunakan *framework Streamlit* untuk melaksanakan proses implementasi model dalam bentuk aplikasi web. Diagram proses implementasi model dapat dilihat pada Gambar 11.



Gambar 11. Diagram alur implementasi

Keterangan:

a. Membangun Model, proses di mana model dikonstruksi dan disiapkan untuk dilatih dan dievaluasi. Ini mencakup beberapa tahap penting:

- **Pelatihan,** pelatihan model menggunakan *dataset* yang sudah diproses. Selama fase ini, model belajar mengenali pola dalam data dengan menyesuaikan bobot berdasarkan input yang diberikan.
- **Evaluasi,** evaluasi model dilakukan untuk mengukur kinerjanya menggunakan data validation. Ini melibatkan pengukuran metrik seperti akurasi, *precision*, *recall*, dan *F1-Score* untuk menilai seberapa baik model bekerja pada tugas yang diberikan.
- **Menyimpan,** Setelah pelatihan dan evaluasi, model yang sudah dilatih dan memiliki kinerja yang baik disimpan ke dalam format file tertentu

(seperti .h5, .pb, atau model checkpoint) sehingga dapat digunakan kembali di masa depan tanpa harus melatih ulang.

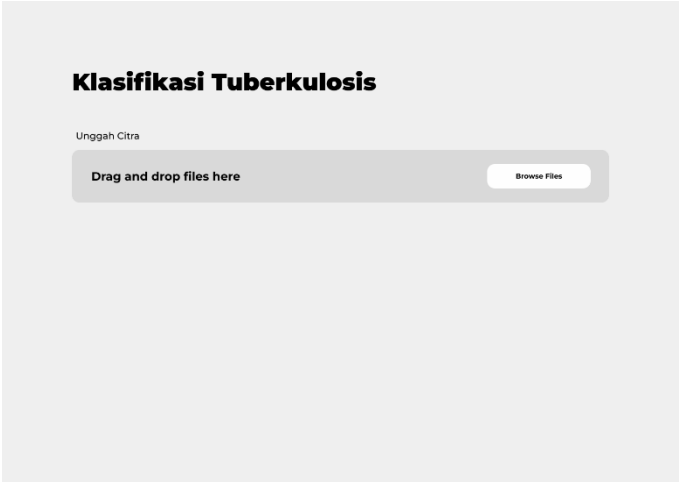
b. Implementasi Model, proses dimana model yang terbaik di-*deploy* dalam bentuk aplikasi web. Ini mencakup beberapa tahap yaitu:

- **Membuat Antarmuka,** proses pembuatan antarmuka pengguna untuk berinteraksi dengan model. Antarmuka ini bisa berupa aplikasi web atau *dashboard* yang memudahkan pengguna untuk menginput data dan melihat prediksi model.
- **Memuat Model,** model yang telah disimpan sebelumnya dimuat kembali ke dalam lingkungan implementasi. Ini memastikan bahwa model siap digunakan untuk melakukan prediksi pada data baru.
- **Model Diterapkan,** model diterapkan ke dalam aplikasi web, sehingga dapat menerima input dari pengguna atau sistem lain dan memberikan *output* prediksi.
- **Pengujian,** setelah implementasi model, pengujian dilakukan untuk memastikan bahwa model berfungsi dengan baik di aplikasi web. Pengujian ini mencakup pengecekan fungsionalitas, kecepatan respons, dan keandalan model dalam memberikan hasil yang akurat.

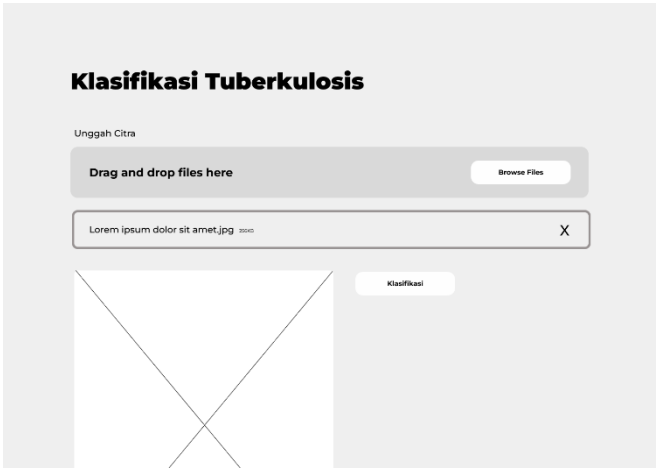
2.9 Desain Sistem

Dalam penelitian ini, aplikasi web dikembangkan dengan menggunakan *framework Streamlit* untuk menciptakan antarmuka pengguna atau *dashboard* yang intuitif. *Dashboard* ini dirancang khusus untuk membantu pengguna dalam mengklasifikasikan tuberkulosis pada citra rontgen dada. Desain sistem ini terdiri dari dua bagian utama:

2.9.1 Halaman Awal



(a) Halaman awal



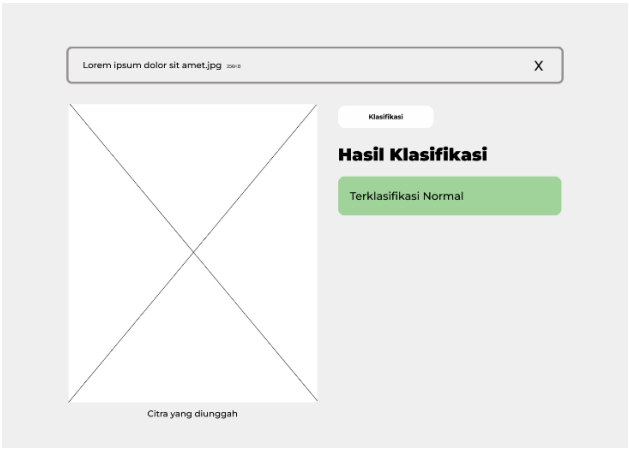
(b) Tampilan setelah mengunggah citra

Gambar 12. Desain tampilan awal

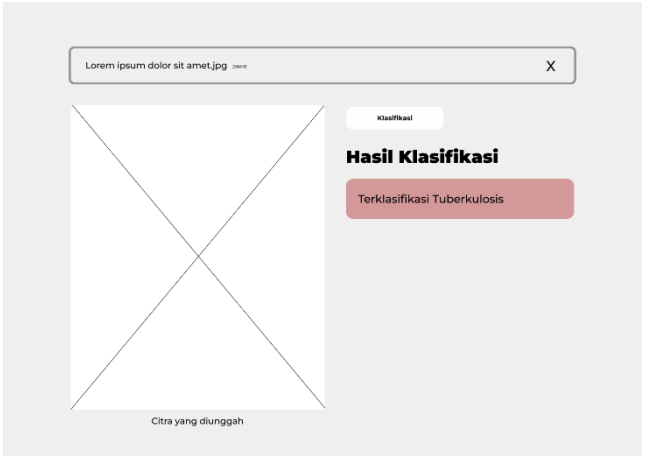
Gambar 12 menunjukkan desain halaman awal, termasuk tampilan sebelum dan setelah citra diunggah oleh pengguna. Halaman ini adalah tampilan pertama yang dilihat oleh pengguna saat membuka aplikasi. Desainnya mencakup antarmuka utama yang menyambut pengguna. Setelah pengguna mengunggah citra, tampilan akan berubah sesuai dengan proses selanjutnya.

2.9.2 Halaman Klasifikasi

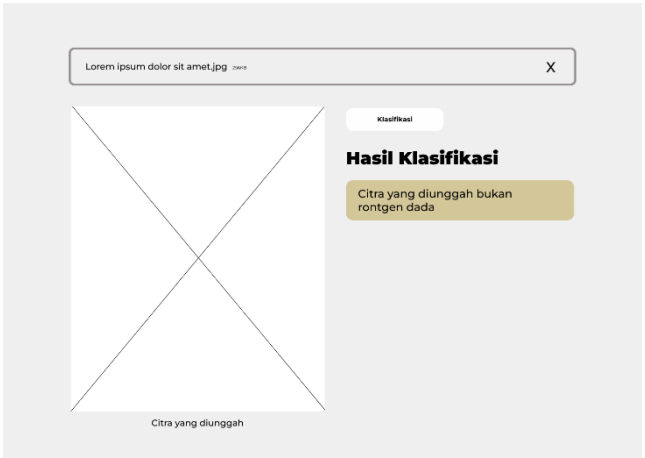
Halaman ini menampilkan tiga kondisi: citra terklasifikasi sebagai normal, terklasifikasi sebagai tuberkulosis, atau menunjukkan bahwa citra yang diunggah bukan merupakan citra rontgen dada. Desain rancangan halaman ini dapat dilihat pada Gambar 13.



(a) Normal



(b) Tuberkulosis

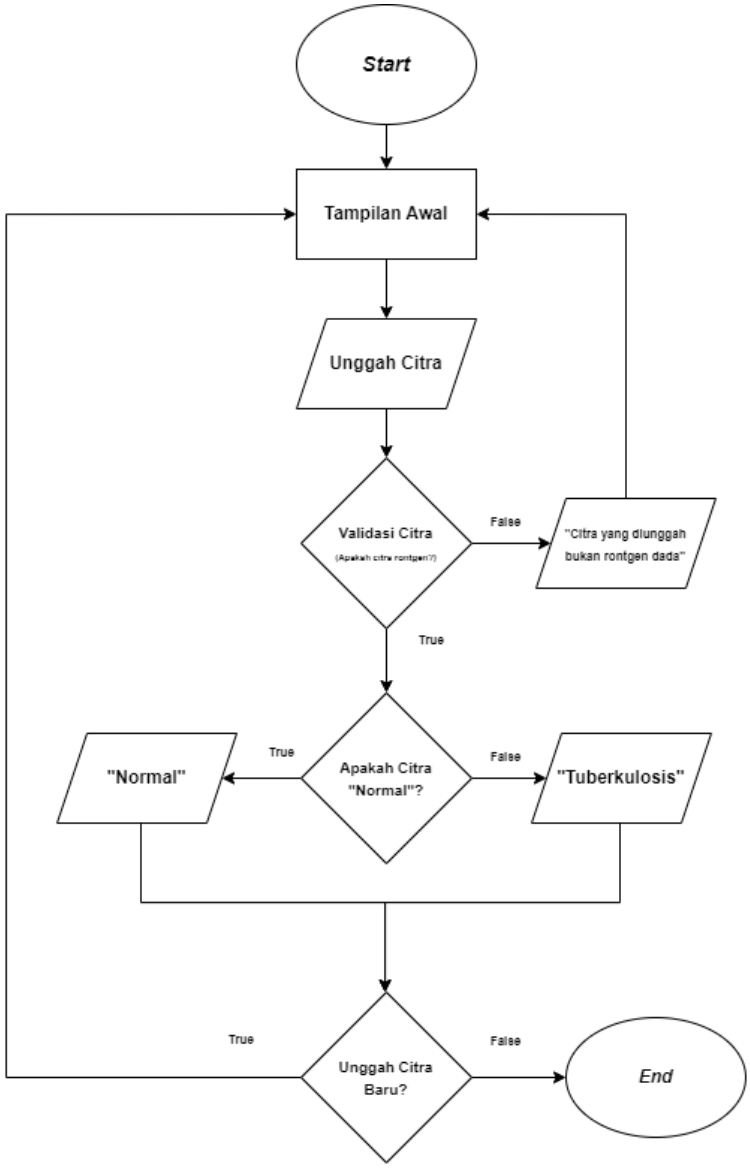


(c) Bukan rontgen dada

Gambar 13. Desain tampilan klasifikasi

2.10 Pengujian Sistem

Tahap terakhir yang dilakukan adalah menguji sistem untuk memastikan bahwa aplikasi web yang dibuat bekerja dengan baik dan sesuai dengan yang diharapkan. Diagram alur pengujian sistem dapat dilihat pada Gambar 14.



Gambar 14. Alur pengujian sistem

Keterangan:

a. **Tampilan Awal**, pengguna berada di halaman utama aplikasi web, di mana mereka dapat mengunggah citra rontgen dada untuk dianalisis
b. **Unggah Citra**, pengguna mengunggah citra yang ingin mereka analisis. Citra ini seharusnya berupa citra rontgen dada.

c. Validasi Citra, aplikasi web memvalidasi apakah citra yang diunggah benar-benar merupakan citra rontgen dada. Ini adalah langkah penting untuk memastikan bahwa citra yang diunggah sesuai dengan apa yang dapat diproses oleh aplikasi web.

- Jika Benar (*True*): Jika citra adalah citra rontgen dada, proses berlanjut ke tahap berikutnya.
- Jika Salah (*False*): Jika citra bukan rontgen dada, aplikasi menampilkan pesan kesalahan, “Citra yang diunggah bukan rontgen dada”, dan kembali ke tampilan awal.

d. **Apakah Citra "Normal"?**, aplikasi web melakukan klasifikasi untuk menentukan apakah citra rontgen yang diunggah menunjukkan kondisi "Normal" atau "Tuberkulosis".

- Jika Benar (*True*): Jika citra diklasifikasikan sebagai "Normal", hasil ini ditampilkan kepada pengguna.
- Jika Salah (*False*): Jika citra diklasifikasikan sebagai "Tuberkulosis", hasil ini ditampilkan kepada pengguna.

e. Unggah Citra Baru?, setelah hasil klasifikasi ditampilkan, aplikasi web terbuka untuk mengunggah citra baru untuk dianalisis.

- Jika Benar (*True*): Pengguna memilih untuk mengunggah citra baru, proses kembali ke tahap "Unggah Citra".
- Jika Salah (*False*): Pengguna memilih untuk tidak mengunggah citra baru, dan proses berakhir (*End*).