DETEKSI DAN IDENTIFIKASI JENIS LUKA LUAR BERDASARKAN IMAGE *FEATURE* MENGGUNAKAN CNN DENGAN VARIASI *PRE-TRAINED MODEL*

SKRIPSI

ZAHRA RAMADHANTI
F1E119101

PROGRAM STUDI SISTEM INFORMASI
JURUSAN TEKNIK ELEKTRO DAN INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS JAMBI
2024

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi ini benar-benar karya saya sendiri. Sepanjang pengetahuan saya tidak terdapat karya atau pendapat yang ditulis atau diterbitkan orang lain kecuali sebagai acuan atau kutipan dengan mengikuti tata penulisan karya ilmiah yang telah lazim.

Tanda tangan yang tertera dalam halaman pengesahan ini adalah asli. Jika tidak asli, saya siap menerima sanksi sesuai dengan peraturan yang berlaku.

Jambi, 27 Desember 2023

Yang menyatakan,

Zahra Ramadhanti

DETEKSI DAN IDENTIFIKASI JENIS LUKA LUAR BERDASARKAN IMAGE *FEATURE* MENGGUNAKAN CNN DENGAN VARIASI *PRE-TRAINED MODEL*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh Gelar Sarjana pada Program Studi Sistem Informasi

ZAHRA RAMADHANTI
F1E119101

PROGRAM STUDI SISTEM INFORMASI
JURUSAN TEKNIK ELEKTRO DAN INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS JAMBI
2024

PENGESAHAN

Skrpsi dengan judul "**DETEKSI DAN IDENTIFIKASI JENIS LUKA LUAR BERDASARKAN *IMAGE FEATURE* MENGGUNAKAN CNN DENGAN VARIASI *PRE-TRAINED MODEL*"** yang disusun oleh **ZAHRA RAMADHANTI, NIM: F1E119101** telah dipertahankan di depan penguji pada tanggal 15 Desember 2023 dan dinyatakan Lulus.

Susunan Tim Penguji:

Ketua	:	Ulfa Khaira, S.Kom., M.Kom.
Sekretaris	:	Pradita Eko Prasetyo Utomo, S.Pd., M.Cs.
Anggota	:	1. Dedy Setiawan, S.Kom., M.IT.
		2. Benedika Ferdian Hutabarat, S.Komp., M.Kom.
		3. Zainil Abisin, S.T., M.Eng.

Disetujui:

Pembimbing Utama	Pembimbing Pendamping
Ulfa Khaira, S.Komp., M.Kom.	Pradita Eko Prasetyo Utomo, S.Pd., M.Cs.
NIP. 198912292019032018	NIP. 198710282019031010

Diketahui:

Dekan Fakultas Sains dan Teknologi	Ketua Jurusan Teknik Elektro dan Informatika
Drs. Jefri Marzal, M.Sc., D.I.T.	Nehru, S.Si., M.T.
NIP.196806021993031004	NIP.197602082001121002

RINGKASAN

Luka luar dapat terjadi di mana saja dan kapan saja akibat faktor eksternal, namun dampaknya terhadap kesehatan tidak selalu meningkatkan kesadaran masyarakat karena kurangnya efisiensi layanan kesehatan. Dengan banyaknya masyarakat pengguna *smartphone* dan gawai pintar lainnya, diperlukan solusi yang memanfaatkan teknologi terutama kecerdasan buatan, seperti algoritma *Convolutional Neural Network* (CNN), untuk mengklasifikasikan gambar luka ringan dan memberikan rekomendasi pertolongan pertama. Model yang dibangun pun menggunakan metode *transfer learning* pada CNN untuk mempersingkat waktu pelatihan dan diharapkan dapat memberikan hasil akurasi yang baik, dengan perbandingan tiga *pre-trained model* yaitu VGG-16, Inception-v3, dan ResNet50 dengan tambahan lapisan model dan *hyperparameter* yang sama seperti inisiasi *learning rate* awal yaitu 0,001, *optizimer* Adam, fungsi *loss* 'categorial cross entrtopy', dua fungsi *callbacks* yaitu *EarlyStopping* dan *ReduceLROnPlateu*, dengan jumlah *epoch* maksimal 100 dan *batch* pelatihan sebanyak 16. Hasil pelatihan dan pengujian menunjukkan bahwa model dengan *pre-trained* VGG-16 memiliki nilai terbaik, dengan akurasi validasi, *precision*, *recall*, dan *f1-score* mencapai 97%, serta akurasi uji sebesar 97%. Model ini pun berhasil diimplementasikan pada web untuk membantu pengguna yang terluka.

External wounds can occur anywhere and at any time due to external factors, but their impact on health does not always increase public awareness due to the lack of efficiency of health services. With so many people using smartphones and other smart devices, a solution is needed that utilizes technology, especially artificial intelligence, such as the Convolutional Neural Network (CNN) algorithm, to classify images of minor injuries and provide first aid recommendations. The model built also uses the transfer learning method on CNN to shorten training time and is expected to provide good accuracy results, with a comparison of three pre-trained models, namely VGG-16, Inception-v3, and ResNet50 with additional model layers and the same hyperparameters as initial learning rate initiation of 0.001, Adam optizimer, 'categorial cross entrtopy' loss function, two callbacks functions namely EarlyStopping and ReduceLROnPlateu, with a maximum number of epochs of 100 and training batches of 16. The training and testing results show that the model with pre-trained VGG- 16 has the best score, with validation accuracy, precision, recall and f1-score reaching 97%, and test accuracy of 97%. This model has also been successfully implemented on the web to help injured users.

RIWAYAT HIDUP

Zahra Ramadhanti lahir di Jambi pada tanggal 13 Desember 2001. Penulis merupakan anak terakhir dari 4 bersaudara dari pasangan M. Zuharfan dan Nurlaila Herawati. Penulis berhasil menjalani jenjang pendidikan formal yaitu:

1. SD Islam Al-Falah Kota Jambi (2007-2013)
2. SMP Negeri 1 Kota Jambi (2013-2016)
3. SMA Negeri 1 Kota Jambi (2016-2019)

Yang mana pada tahun 2019, penulis mulai menempuh pendidikan strata 1 di Universitas Jambi melalui jalur SMMPTN Barat. Selama menempuh pendidikan, penulis termasuk aktif dalam berkegiatan baik di dalam kampus maupun luar kampus. Seperti Himpunan Mahasiswa Sistem Informasi sebagai Koordinator Divisi Dana Usaha dan kepanitiaan lainnya pada acara kampus. Selain di dalam kampus, penulis juga mengikuti studi independen melalui program Merdeka Belajar Kampus Merdeka (MBKM) yaitu pada program Bangkit *Academy* oleh Google, Goto, dan Traveloka dan berfokus pada bidang machine learning. Melalui program Bangkit, penulis pun memanfaatkan ilmu yang dipelajari dan digunakan dalam studi kasus industri pada projek akhir yang berhasil mendapatkan dana inkubasi pengembangan program selama 6 bulan sebanyak 140 juta rupiah oleh Google dan DIKTI.

PRAKATA

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, yang telah memberikan jalan kepada penulis dalam menyelesaikan penulisan skripsi berjudul "DETEKSI DAN IDENTIFIKASI JENIS LUKA LUAR BERDASARKAN *IMAGE FEATURE* MENGGUNAKAN CNN DENGAN VARIASI *PRE-TRAINED MODEL*." Tanpa bimbingan, arahan, serta bantuan lainnya dari berbagai pihak, skripsi ini akan menjalani proses yang sulit. Oleh karena itu, penulis hendak berterima kasih kepada seluruh pihak yang terlibat secara langsung maupun tidak, yaitu:

1. Ayah dan Alm. Ibu yang semasa hidupnya tak pernah berhenti mendoakan dan memberi dorongan kepada saya dalam setiap langkah yang saya ambil.
2. Bapak Drs. Jefri Marzal, M.Sc., D.I.T., selaku Dekan Fakultas Sains dan Teknologi Universitas Jambi.
3. Bapak Edi Saputra, S.T., M.Sc., selaku Ketua Program Studi Sistem Informasi.
4. Ibu Ulfa Khaira, S.Komp., M.Kom dan Bapak Pradita Eko Prasetyo Utomo, S.Pd., M.Cs selaku Dosen Pembimbing Tugas Akhir yang bersedia meluangkan waktunya untuk membimbing saya.
5. Thom selaku *partner* yang selalu memberikan semangat, dorongan, doa, dan tak pernah mengeluh selama menemani perjalanan saya dalam menyelesaikan skripsi ini.
6. THADA's family yang selalu bersedia menjawab pertanyaan saya dan tak ragu membantu saya.
7. Serta keterlibatan pihak lainnya yang tidak bisa saya sebutkan satu per satu namun turut membantu saya dalam menyelesaikan skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna sehingga kritik dan saran yang diberikan akan sangat membantu agar skripsi ini dapat memberikan dampak positif kepada siapapun yang membacanya baik di lingkungan Universitas Jambi maupun khalayak umum.

Jambi, 27 Desember 2023

Yang menyatakan,

Zahra Ramadhanti

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

I. PENDAHULUAN

1.1 Latar Belakang

Luka pada kulit, atau yang umumnya disebut luka luar, merupakan suatu kondisi yang sering terjadi di tengah masyarakat, terutama pada anak-anak, dimana kondisi fisiknya masih lemah dan belum terlalu mengenal keadaan lingkungannya sehingga sering mengalami kecelakaan seperti saat berlarian, memanjat pohon, dan sebagainya (Biomi & Swandewi, 2020). Kecelakaan yang terjadi bisa disebabkan oleh banyak faktor yaitu, kebakaran, terjatuh, tertusuk benda tajam, dan masih banyak lagi. Luka yang termasuk penyakit kulit akan semakin parah jika kebiasaan masyarakat dan lingkungannya tidak bersih. Luka bisa terinfeksi bakteri, jamur dan virus. Serta kerugian yang diakibatkan oleh luka yang terinfeksi antara lain adalah rasa sakit, kesulitan dalam beraktivitas, isolasi sosial, kecemasan, perpanjangan tinggal di rumah sakit, morbiditas kronis atau bahkan kematian jika ditangani dengan salah (Agustina et al., 2016). Guna menghindari memburuknya atau terinfeksinya luka, penanganan pertama pada satu jam pertama terjadinya luka adalah hal yang penting (Anggraini et al., 2018). Kecelakaan yang dapat terjadi kapanpun ini pun menjadikan penangan luka menjadi keterampilan yang dapat digunakan tanpa batas waktu (Wijaya, 2018).

Menurut data Badan Pusat Statistik Indonesia, tingkat terjadinya luka ringan di Indonesia meningkat dari tahun ke tahun dengan total 117.913 luka ringan pada tahun 2021 (Badan Pusat Statistik, 2023). Namun pada prakteknya, angka tersebut tidak membuat kesadaran akan luka di masyarakat Indonesia sebanding dengan jumlah terjadinya luka. Dibuktikan dengan survei yang telah dilakukan oleh Badan Penelitian dan Pengembangan Kesehatan Kementerian Kesehatan RI, data dari Riset Kesehatan Dasar menunjukkan bahwa tingkat kesadaran masyarakat akan kesehatan masih tergolong rendah dengan kisaran perhitungan 20% (Badan Strategi Kebijakan Dalam Negeri, 2018). Hal ini menandakan bahwa hanya 53,54 juta dari total 267,7 juta penduduk Indonesia tahun 2018 yang mengerti akan kesehatan sedangkan 214,16 juta lainnya tidak mengerti. Kesadaran adalah hal paling mendasar dan penting dalam perubahan kebiasaan hidup seseorang. Hal ini menunjukkan bahwa kesadaran akan kesehatan ini yang terbentuk pada setiap individu ini lah yang menjadi salah satu faktor penting yang dapat menaikkan derajat kesehatan masyarakat yang optimal (Muttaqien et al., 2019).

Kesadaran masyarakat yang tergolong rendah ini dimungkinkan oleh banyak hal. Beberapanya dikarenakan sulitnya akses fasilitas kesehatan yang ada. Menurut penelitian yang dilakukan oleh Taber pada tahun 2015, dari 1.369 responden, lebih dari 50% menyatakan kendala dalam penanganan medis seperti tingginya biaya layanan kesehatan, tidak adanya kepemilikan asuransi kesehatan, dan kendala pada lamanya waktu penanganan (Taber et al., 2015).

Ditambah lagi, perkembangan ekonomi global yang tidak seimbang menyebabkan kurangnya pemerataan kualitas penangangan kesehatan. Indonesia yang menjadi salah satu negara berkembang ini pun ikut terdampak. Perkembangan ekonomi global yang tidak seimbang mengartikan bahwa sebagian besar pasien cedera atau terluka di negara berkembang atau daerah pedesaan tidak memiliki akses ke diagnosis luka yang tepat, pedoman perawatan berbasis bukti yang ada, teknologi tepat guna, atau keahlian tenaga medis yang diperlukan untuk menjalankan proses penyembuhan yang optimal (Jennett et al., 2003).

Pembuatan teknologi luka yang tepat dapat bermanfaat dalam banyak hal, antara lain adalah mengurangi beban kerja tenaga medis, waktu yang dibutuhkan untuk pergi ke fasilitas kesehatan, dan beban biaya yang dikeluarkan. Hal ini didukung oleh data dari Statista bahwa terdapat 6,64 miliar pengguna *smartphone* di seluruh dunia, atau sekitar 83,32% dari total populasi manusia di bumi (Taylor, 2023). Dengan banyaknya pengguna, manfaat penggunaan teknologi yang disebutkan tadi juga dapat dirasakan oleh banyak orang. Salah satu pemanfaatan teknologi yang tepat adalah terciptanya kecerdasan buatan yang dapat mendeteksi luka.

Kecerdasan buatan (AI) adalah bidang yang berkembang pesat dan menghasilkan harapan untuk mengatasi masalah yang membingungkan. Penelitian ini memiliki akar permasalahan berupa klasifikasi citra, dimana klasifikasi citra merupakan bagian dari *Computer Vision*. Tujuannya adalah untuk mengklasifikasikan citra yang dimasukkan ke dalam kategori tertentu. Salah satu solusi untuk masalah ini adalah dengan menggunakan ilmu *Machine Learning* (ML). *Machine Learning* (ML) adalah teknik pembelajaran di mana teknologi memungkinkan algoritma untuk memproses citra atau gambar, suara, dan gerakan. Ini dapat diintegrasikan ke dalam alat digital kehidupan sehari-hari dan alur kerja profesional otomatis (Fourcade & Khonsari, 2019).

Saat ini metode algoritma *machine learning* sedang marak digunakan, salah satunya adalah metode *deep learning*. Metode *deep learning* menjadi sangat populer karena meningkatnya jumlah data beranotasi karena dapat mengekstraksi fitur tanpa aturan yang dirancang manusia (Anisuzzaman et al.,

2022). Metode ini juga telah menjadi kata kunci saat ini karena hasil canggih yang diperoleh dalam domain klasifikasi citra/gambar, deteksi objek, dan pemrosesan bahasa alami (Pathak et al., 2018). Deteksi objek merupakan salah satu masalah paling kritis dan kompleks dalam visi komputer (*computer vision*). Selama dekade ini, dengan pesatnya perkembangan *deep learning*, para peneliti telah bereksperimen dalam skala besar dan membantu meningkatkan deteksi objek dan tugas terkait seperti klasifikasi objek, lokalisasi, dan segmentasi menggunakan model mendalam yang mendasarinya (Diwan et al., 2023). Pengenalan/dan kualifikasi objek atau gambar ini mengalami kemajuan yang luar biasa. Salah satunya dikarenakan ketersediaan kumpulan data beranotasi berskala besar dan jaringan saraf convolutional (CNN) yang dalam (Kido et al., 2018).

Terdapat beberapa algoritma selain CNN yang dapat melakukan klasifikasi gambar, seperti SVM, *Random Forest*, dan KNN. Namun, perbedaan mendasar antara CNN dan algoritma lainnya terletak pada kemampuan CNN untuk mengekstraksi fitur otomatis dari gambar, sementara algoritma seperti Random Forest dan KNN memerlukan ekstraksi fitur manual sebelum dilatih. CNN juga unggul dari SVM karena kemampuannya dalam mengekstraksi fitur secara bertingkat, sebagai contoh, lapisan pertama CNN mengidentifikasi tepian, lapisan kedua mengidentifikasi bentuk, dan seterusnya. Penelitian membandingkan SVM, KNN, dan CNN dalam klasifikasi gambar cuaca menunjukkan bahwa CNN memiliki akurasi tertinggi (94%), di atas SVM (86%) dan KNN (76%) (Naufal, 2021). Dalam penelitian lain mengenai pengenalan penyakit tanaman, CNN juga unggul dengan akurasi 97,8%, diikuti oleh *Random Forest* (87,4%), SVM (78,6%), dan KNN (76,9%) (Hatuwal et al., 2020). Karena efisiensi dan kemampuannya dalam tugas klasifikasi gambar, baik dari awal maupun dengan transfer learning, CNN menjadi pilihan utama dalam membangun model *machine learning* untuk klasifikasi gambar.

Pembangunan model deteksi atau klasifikasi menggunakan algoritma CNN ini dapat dilakukan dalam 2 teknik: 1) Membangun dan melatih model CNN dari awal, dan 2) Menggunakan metode *transfer learning* dengan menggunakan model yang telah ada (siap pakai) tanpa pelatihan ulang. *Transfer learning* ini adalah metode yang popular digunakan dalam membangun sebuah model. Hal ini dikarenakan kemampuannya yang dapat mengurangi waktu serta usaha pengembang dalam membangun model yang diinginkan. Metode yang digunakan adalah menggunakan kembali pengetahuan dari *pre-trained model* yang telah ada untuk tugas yang lain. Metode *transfer learning* dapat digunakan untuk berbagai kasus seperti klasifikasi, regresi, dan *clustering* (Tammina,

2022). Terdapat berbagai *pre-trained model* yang terkenal dengan kemampuannya yang baik dapat melakukan tugas klasifikasi gambar seperti Inception, ResNet, dan juga VGG. Ketiga *pre-trained model* tersebut pernah mengikuti kompetesi yang diadakan oleh ImageNet yaitu *Imagenet Large Scale Visual Recognation Challenge* (ILSVRC) dimana kompetesi tersebut mengharuskan partisipan untuk membangun sebuah model yang dapat melatih 1,2 juta data gambar yang dapat mengklasifikasikan gambar hingga 1000 label. Inception adalah salah satu model yang dibangun oleh para peneliti dari Google dan pernah memenangi kompetesi tersebut pada tahun 2014 (Szegedy et al., 2016) yang disusul oleh VGG yang dibangun oleh tim peneliti asal Universitas Oxford sebagai posisi kedua (Simonyan & Zisserman, 2015). Selanjutnya ResNet yang dibangun oleh para peneliti *Microsoft Research Asia* yang juga pernah mengikuti kompetesi tersebut dan berhasi memenangkannya pada tahun 2015 (He et al., 2016a).

Berbagai penelitian yang memanfaatkan ilmu *machine learning* dan algoritma CNN ini sudah terbilang cukup banyak dalam membangun model yang dapat mengidentifikasi berbagai jenis luka. Beberapa dari penelitian tersebut adalah klasifikasi ulkus diabetikum kaki dengan algoritma CNN (Goyal et al., 2018), klasifikasi ulkus diabetikum kaki dengan membandingkan berbagai *pre-trained model* (Alzubaidi et al., 2020), klasifikasi ulkus (Liu et al., 2020), dan klasifikasi luka bakar (Despo, 2015). Namun, dari berbagai penelitian yang ditemukan, belum ada penelitian yang membahas atau membangun model identifikasi atau deteksi jenis luka ringan seperti luka sayat, luka tusuk, luka robek, dan sebagainya yang termasuk dalam jenis luka akut.

Permasalahan tersebut pun mendorong penelitian ini untuk menemukan sebuah solusi dengan memanfaatkan teknologi AI dan metode *transfer learning* CNN. Rencana penelitian yang sudah dirancang adalah dengan membangun model dengan membandingkan hasil dari berbagai *pre-trained model* dan menemukan yang terbaik untuk diimplementasikan pada sistem identifikasi luka luar pada kulit manusia serta menunjukkan saran pertolongan pertama pada setiap luka yang terdeteksi sehingga dapat meminimalkan tenaga dan biaya sebelum menemui ahli medis di rumah sakit atau fasilitas medis lainnya.

1.2 Rumusan Masalah

Bedasarkan latar belakang di atas, permasalahan di penelitian ini adalah:

1. Bagaimana cara mengimplementasikan metode *transfer learning* pada CNN dalam membuat model identifikasi luka luar manusia?

2. Bagaimana tingkat akurasi dari masing-masing model yang dihasilkan oleh berbagai metode *transfer learning* yang telah dilakukan?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah:

1. Mengetahui cara mengimplementasikan metode *transfer learning* pada CNN dalam membuat model identifikasi luka luar manusia
2. Mengetahui tingkat akurasi dari masing-masing model yang dihasilkan dan menemukan model dengan hasil yang terbaik

1.4 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Memberikan informasi kepada para pembaca mengenai solusi atau alternatif lain akan penyampaian informasi mengenai jenis luka hanya dengan menggunakan gambar.
2. Membantu masyarakat dengan model yang dibangun untuk mengetahui jenis luka yang terjadi sehingga mengurangi atau bahkan menyelesaikan permasalahan yang ada terhadap luka dan kesehatan

II. TINJAUAN PUSTAKA

2.1 Luka

Luka merupakan suatu kerusakan jaringan kulit normal yang terbentuk oleh berbagai faktor penyebab kerusakan (G. Wang et al., 2023). Hal ini dapat diartikan bahwa luka dapat terjadi apabila terdapat tindakan eksternal yang baik sengaja maupun tidak sehingga dapat memengaruhi jaringan kulit dan terjadilah kerusakan.

Menurut ilmu kesehatan, terdapat berbagai jenis kategori luka yang dikelompokkan berdasarkan berbagai indikator, di antara lain: kondisi fisik luka (tampilan), lama waktu penyembuhan dan penyebab, benda tajam dan tampilan, kedalaman luka, dan sebagainya. Namun, pada penelitian ini, luka yang akan digunakan sebagai sampel data adalah kategori luka berdasarkan kondisi fisik/tampilan dan lama waktu penyembuhan dan penyebab. Berikut adalah beberapa kategori luka, serta jenis-jenis luka yang termasuk di dalamnya:

Berdasarkan Kondisi Fisik (Tampilan):

Kondisi fisik atau tampilan luka ini membagi jenis luka menjadi dua, yaitu luka terbuka dan luka tertutup.

1. Luka Terbuka

   Luka terbuka adalah kategori luka yang memiliki tampilan terbuka atau tepatnya ketika terjadi kerusakan terbuka pada kulit atau jaringan tubuh. Berdasarkan buku "Tintinally's Emergency Medicine: A Comprehensive Study Guide" oleh (Tintinalli et al., 2016), jenis-jenis luka yang termasuk dalam kategori luka terbuka adalah sebagai berikut:

   a. Luka Lecet (*Abrasion Wound*)

      Luka lecet atau biasa hanya disebut dengan 'lecet' adalah kerusakan pada permukaan kulit secara abrasi. Luka ini disebabkan oleh pertemuan antara kulit dan benda lain dengan permukaan yang kasar (Mao et al., 2017). Luka lecet ini terjadi ketika permukaan kasar tersebut bergesekan dengan kulit di luar batas kekuatan jaringan kulit dalam menahannya. Hal ini menyebabkan terputusnya kontinuitas jaringan pada kulit namun masih tergolong sebagai luka dengan penyembuhan yang sederhana karena hanya mengenai bagian teratas kulit atau epidermis dengan pendarahan yang sedikit (.

      Terdapat berbagai bentuk luka lecet, seperti abrasi linear/goresan yang paling sederhana karena hanya sedikit merusak lapisan epidermis kulit (biasanya hanya satu garis) dan biasanya

disebabkan oleh cakaran kuku. Lalu, goresan/abrasi kuas yaitu bentuk abrasi linear yang berlipat

ganda yang biasanya ditemukan pada kasus kecelakaan karena terjadinya gesekan antar kulit dan permukaan aspal (lalu lintas jalan). Dan yang terakhir adalah lecet berpola yang disebabkan oleh tekanan benda dengan permukaan kasar sehingga menghasilkan pola luka yang dapat mencirikan objek penyebab luka tersebut. Lecet berpola ini biasa ditemukan pada kasus akibat pukulan dan tabrakan (Shrestha et al., 2022) yang dapat dilihat pada Gambar 1.

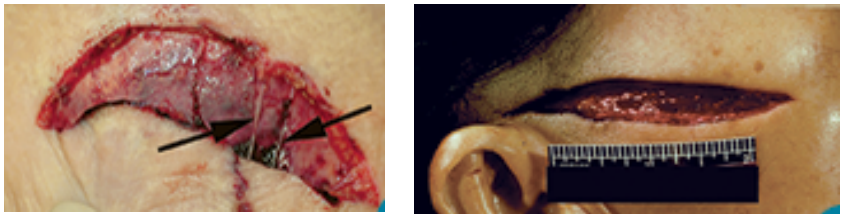
Gambar 1. Abrasi Linear pada Pergelangan Tangan Kanan

b. Luka Sayat (*Incised Wound*)

Luka sayat merupakan jenis luka terbuka yang terjadi akibat trauma benda tajam. Luka ini memiliki tampilan yang lebih bersih tanpa jembatan jaringan dengan tepi luka yang terlihat tajam atau lancip dan teratur sehingga lebih mudah untuk dilakukan penanganan berupa jahitan. Luka sayat sendiri dicirikan sebagai luka panjang dan sempit pada permukaan lukanya terutama pada kulit (Rozzi, 2014).

c. Luka Robek

Berbeda dengan luka sayat yang hanya disebabkan oleh benda tajam, luka robek dapat disebabkan oleh benda tajam yang dapat memotong kulit dan jaringan di bawahnya atau benda tumpul yang keras dan kasar sehingga menyebabkan terpisahnya elemen jaringan kulit yang tidak terjadi sepenuhnya (tidak benar-benar lepas dari bagian tubuh). Selain itu, apabila luka sayat terlihat lebih rapi, luka robek terlihat lebih tak beraturan. Perbedaan luka robek dan luka sayat dapat dilihat pada Gambar 2.



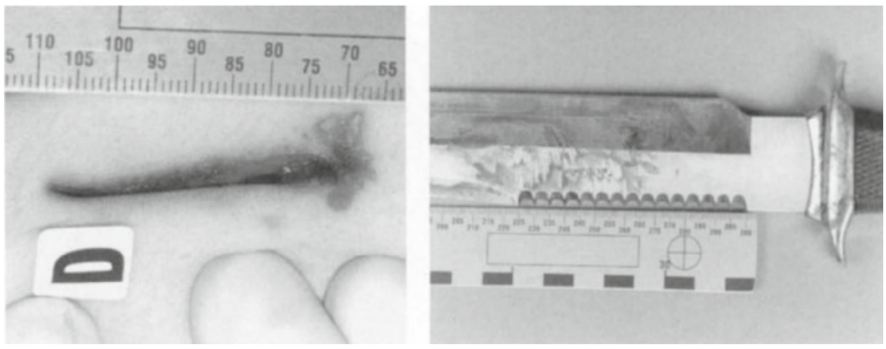
A B

Gambar 2. Luka Robek dan Luka Sayat (Rozzi, 2014): A. Luka Robek, B. Luka Sayat

Pada Gambar 2, terdapat bagian yang ditunjuk oleh panah, bagian itu adalah jembatan jaringan yang dimiliki oleh luka robek sehingga memudahkan pengamat dalam membedakan luka robek dengan luka sayat (Rozzi, 2014).

d. Luka Tusuk

Luka tusuk adalah luka yang terjadi akibat tekanan benda tajam yang menembus jaringan kulit bahkan organ yang lebih dalam (Tintinalli et al., 2016). Tampilan luka tusuk ini mirip dengan luka sayat namun lebih kecil atau tidak selebar luka sayat namun lebih dalam dari luka sayat (Davison, 2004). Luka tusuk ini dapat menyebabkan pendarahan yang hebat tergantung dengan lokasi dan kedalaman luka yang terjadi. Salah satu luka tusuk yang disebabkan oleh pisau dapat dilihat pada Gambar 5.



A B

Gambar 3. Luka Tusuk dan Alat Tusuk (Davison, 2004): A. Luka Tusuk, B. Pisau yang Digunakan untuk Menusuk

Gambar 3 (A) menunjukkan luka tusuk di dada yang menunjukan abrasi pada salah satu ujungnya, sesuai dengan ujung pisau dekat pelindung pegangan pisau pada Gambar 3 (B) yaitu pisau yang digunakan untuk melukai, terdapat gerigi dekat pelindung pegangan pisau yang menandakan bahwa kedalam luka tusuk hampir sama dengan panjang pisau yang digunakan.

e. Luka Bakar

Luka bakar adalah luka yang terjadi karena berbagai sumber, seperti panas (suhu panas, nyala api, kontak dengan permukaan panas), listrik, kimia (asam, bensin, pembersih rumah tangga, produk kebun), dan radiasi. Cedera luka bakar dapat berkisar dari ringan hingga parah (Manning, 2018). Berdasarkan informasi yang diberikan oleh *UpToDate* yang ditulis oleh (Rice dan Orgill, 2023) , luka bakar terbagi menjadi tiga jenis utama berdasarkan tingkat kedalamannya.

(1) *Superficial-thickness Burn* yaitu luka bakar paling ringan yang hanya mengenai lapisan epidermis. Tanda dari luka bakar ini adalah kemerahan pada kulit yang terasa nyeri dan panas, serta kulit yang memucat jika diberi tekanan, namun tidak sampai melepuh. Penyembuhan luka bakar jenis *superficial* ini biasanya hanya memerlukan waktu enam hari tanpa meninggalkan bekas luka dan biasanya terjadi karena sengatan matahari.

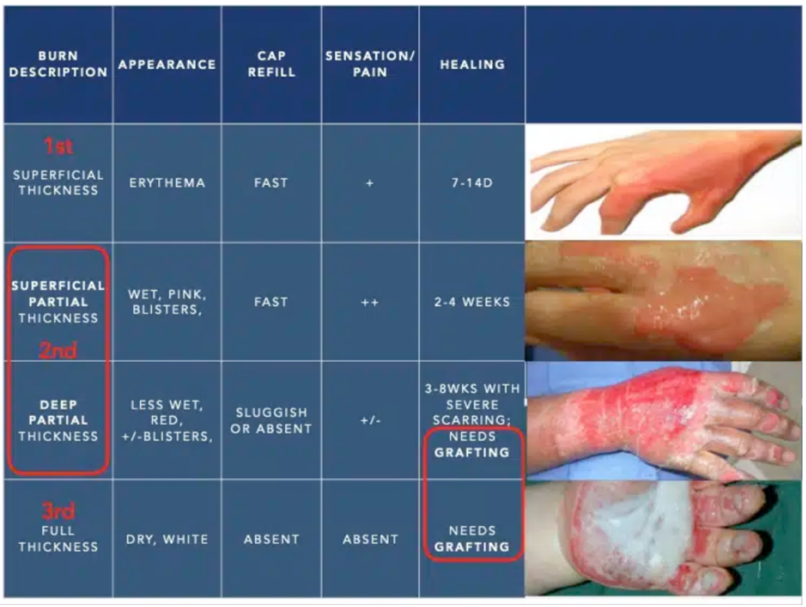
(2) *Partial-thickness Burn* ini adalah jenis luka bakar yang lebih dalam dari pada jenis *superficial*. Jenis luka ini mengenai lapisan epidermis dan beberapa lapisan dermis (lebih dalam lagi). Jenis luka bakar ini terbagi menjadi dua, yaitu:

(a) *Superficial-partical-thickness Burn* yaitu luka bakar yang awalnya terlihat seperti luka bakar tingkat epidermis (*superficial-thickness burn*), namun kulit akan mulai melepuh dalam waktu 7 jam. Biasanya, luka bakar jenis ini memerlukan waktu 7 hingga 21 hari dalam penyembuhannya tanpa gangguan fungsional tertentu.

(b) *Deep-partial-thickness Burn* yaitu luka bakar yang sudah mengenai lapisan dermis yang lebih dalam hingga merusakk folikel rambut dan jaringan kelenjar. Luka bakar ini akan terasa nyeri apabila ditekan dan tidak akan memucat. Biasanya hampir selalu melepuh, basah atau kering seperti lilin, dan memiliki bitnik-bintik dengan warna putih kekuningan dan merah. Tanpa metode pencangkokan dan tidak terinfeksi, luka bakar jenis ini dapat sembuh dalam 2 hingga 9 minggu. Apabila terkena sendi, disfungsi sendi dapat terjadi dan dapat disembuhkan dengan terapi tertentu.

Apabila tidak sembuh dalam 2 minggu, luka jenis *deep-partial* ini sudah termasuk dalam jenis luka bakar *full-thickness.*

(c) *Full-thickness Burn* ini adalah luka bakar yang sudah meluas dan merusak seluruh lapisan dermis dan biasanya merusak lapisan subkutan bawahnya. Biasanya terdapat *eschar* atau dermis yang mati dan terdenaturasi, dan apabila melingkar, dapat memabahayakan kelangsungan anggota atau batang tubuh. Tampilan dari luka bakar ini adalah bervariasi dari putih seperti lilin, abu-abu kasar, hingga paling parah hangus dan hitam. Kulit yang terkena luka bakar ini tidak elastis dan kering. Jenis luka ini tidak dapat sembuh secara spontan dan apabila tidak dioperasi, kontraktur luka dengan epitelialisasi pada sekitar luka akan terjadi. Perbedaan pada beberapa tingkatan luka bakar dapat dilihat pada Gambar 4.



Gambar 4. Perbedaan pada Beberapa Tingkatan Luka Bakar

2. Luka Tertutup

Luka tertutup adalah jenis luka yang terjadi di bawah kulit tanpa merusak permukaan kulit secara langsung, namun tidak ada celah atau retakan pada kulit yang menyebabkan jaringan yang rusak tidak terlihat secara langsung (terbuka). Luka tertutup sendiri membagi kategori luka menjadi dua yaitu yang dapat dilihat dan tidak dapat dilihat oleh mata. Pada penelitian ini, akan di bahas luka tertutup yang dapat dilihat oleh mata. Memar merupakan salah satu dari jenis luka tertutup yang tidak dapat dilihat secara langsung oleh mata.

a. Memar

Memar adalah kondisi dimana darah akibat kerusakan pembuluh darah menumpuk di bawah kulit, atau massa darah di jaringan akibat trauma seperti benturan atau tekanan, dan faktor lainnya yang dapat menyebabkan pecahnya pembuluh darah. Memar ditandai dengan perubahan warna pada permukaan kulit, biasanya berwarna kuning kemerahan, biru, hingga hitam (WebMD Editorial Contributors, 2021). Namun ketika pembuluh darah pecah akibat tusukan jarum, memar dapat terlihat dengan bintik-bintik berwarna merah ke merah tau serta kebiruan (Yu et al., 2017). Tampilan luka memar dapat dilihat pada Gambar 5.

Gambar 5. Luka Memar (Forensicmed.co.uk, 2020)

Berdasarkan Lama Waktu Penyembuhan dan Penyebab

Kategori luka berdasarkan lama waktu penyembuhan dan penyebab terbagi menjad dua, yaitu luka akut dan luka kronis.

1. Luka Kronis

Luka Kronis adalah luka yang memang memakan waktu yang lama dalam penyembuhannya. Pada penyembuhan normal, setelah terjadinya luka, akan terjadinya proses regenerasi pelindung kulit sebagai tahap penyembuhan, namun pada luka kronis, proses transisi luka yang berkembang menjadi regenerasi kulit tersebut memakan waktu tiga bulan hingga lebih. Luka kronis ini ditandai dengan fase peradangan yang berkepanjangan dan berkelanjutan yang mencegah sel dermal dan epidermal kulit untuk bereaksi pada sinyal kimia. Berbagai luka yang termasuk dalam luka kronis adalah sebagai berikut (Goldberg & Diegelmann, 2020):

a. Ulkus Vaskular

b. Ulkus Diabetik
c. Ulkus Tekan

2. Luka Akut

Berbeda dengan luka kronis yang terjadi seiring berjalannya waktu dan rentang waktu penyembuhan yang lama, luka akut merupakan luka yang terjadi secara tiba-tiba, dan terjadi akibat berbagai faktor trauma seperti benturan, api, atau bahkan pembedahan (Zhu et al., 2022). Lebih tepatnya, berbagai jenis luka akut tersebut adalah luka trauma, luka bakar, dan luka bekas operasi (Li et al., 2007).

Kebanyakan luka yang telah dijelaskan sebelumnya memiliki kesamaan pada penyebab dan hasil perubahan pada kulit. Jenis-jenis luka tersebut adalah luka yang termasuk dalam kategori luka berdasarkan tampilan yaitu luka terbuka dan luka tertutup. Kedua jenis luka ini sama-sama disebabkan oleh trauma (faktor eksternal) dan perubahan pada kulit yang dapat terlihat jelas baik itu luka terbuka yang mulai terjadi dari permukaan kulit, hingga luka tertutup yaitu memar yang terjadi di bawah permukaan kulit dengan perubahan warna permukaan kulit yang dapat terlihat jelas. Luka yang termasuk pada jenis luka terbuka adalah luka lecet, luka sayat, luka robek, luka tusuk, dan luka bakar dengan 3 kondisi serta luka yang termasuk pada jenis luka tertutup yaitu memar.

Ketujuh label luka yang disebutkan tersebut, merupakan luka yang terjadi secara tiba-tiba akibat trauma atau bukan terjadi seiring berjalannya waktu, terutama tidak didasari dengan berbagai penyakit sebelumnya. Hal ini mengartikan bahwa ketujuh label luka tersebut masuk dalam kategori luka berdasarkan penyebab dan waktu penyembuhannya, tepatnya luka akut.

Jenis luka ini adalah salah satu dari banyak jenis luka yang sangat sering terjadi di dunia dengan jumlah kasus lebih dari 305 juta per tahunnya. Jumlah ini sembilan kali lebih banyak dari jumlah orang yang menderita kanker di seluruh dunia (Defi et al., 2022). Dengan dasar-dasar tersebut, penelitian ini akan mengambil batasan luka akut sebagai luka yang akan dijadikan sebagai sampel data.

2.2 Pertolongan Pertama pada Luka

Menurut buku “Panduan Bantu Hidup Dasar & Pertolongan Pertama” yang ditulis oleh (Nasution, 2020), pertolongan pertama adalah perawatan yang diberikan pada orang yang tiba-tiba terluka atau sakit. Perawatan ini adalah salah satu cara untuk menangani luka atau sakit agar tidak terjadi hal-hal yang

tidak diinginkan seperti infeksi dan lainnya. Namun, pertolongan pertama tidak menggantikan perawatan medis yang tepat dan seharusnya dilakukan. Melainkan, hanya terdiri dari beberapa bantuan sementara hingga perawatan medis yang kompeten, jika perlu, diperoleh, atau sampai kesempatan pemulihan tanpa perawatan medis dapat dipastikan. Sebelum dan sesudah melakukan perawatan luka, selalu cuci tangan perawat, dan disarankan menggunakan sarung tangan karet bersih (apabila ada) dan selalu membersihkan alat yang bersentuhan dengan luka. Selanjutnya, di bawah ini adalah beberapa pertolongan pertama yang dapat dilakukan kepada beberapa klasifikasi luka yang menjadi batasan penelitian ini:

a. Luka Lecet (*Abrasion Wound*)

Biasanya luka lecet ini terjadi secara sederhana dan seringkali berukuran kecil sehingga tidak memerlukan banyak intervensi medis. Pertolongan pertama yang dapat dilakukan pada saat luka lecet terjadi adalah membersihkan luka dengan air mengalir atau larutan garam. Selanjutnya dapat dilakukan perawatan luka secara basah yaitu dengan menggunakan plester hidrokoloid yang elastis. Perban ini dapat membantu menciptakan lingkungan luka yang tertutup dan lembab sehingga luka tetap bersih, tercegah dari infeksi, dan sembuh lebih cepat.

Namun pada beberapa kondisi terdapat berbagai rekomendasi tindakan tambahan seperti ketika area permukaan terjadinya luka cukup luas dan dapat menghasilkan pembentukan jaringan parut. Hal ini terutama terlihat pada pasien yang rentan terhadap hiperplasia keloid sehingga sebagai rekomendasi, terapi intralesi steroid kortikosteroid dapat dipertimbangkan untuk mencegah pembentukan keloid pada pasien ini.

Setelah luka lecet dibersihkan dan dibalut dengan plester/perban, kondisi luka dapat dilakukan pengecekan berkala dan apabila terjadi infeksi, tindakan debridemen atau pengangkatan jaringan kulit mati dapat dilakukan. Tindakan ini juga dapat dilakukan untuk menghilangkan benda asing pada luka sehingga infeksi pun tidak berlanjut, dan proses penyembuhan dapat berlangsung lebih cepat.

Abrasi wajah dianggap lebih serius karena memiliki risiko sikatrikasi yang lebih tinggi dan harus dibersihkan, dibersihkan, dan dibalut setiap hari. *Dressing* atau perban mungkin memerlukan perekat kulit seperti kombinasi permen karet, styrax, alkohol, dan metil salisilat atau tingtur benzoin.

b. Luka Sayat (*Incised Wound*) dan Luka Robek

Luka sayat dan luka robek memiliki tampilan fisik luka yang cukup mirip sehingga memiliki pertolongan pertama yang sama untuk mengatasi ketika kedua luka ini pertama terjadi. Berdasarkan Buku "Panduan Bantuan Hidup Dasar dan Pertolongan Pertama pada Luka" oleh (Nasution, 2020), terdapat enam langkah perawatan pertama luka sayat dan laserasi (robek), yaitu:

1. Hentikan pendarahan

   Luka terbuka sayat dan robek ini sangat memungkinkan pendarahan yang lumayan parah meskipun pada awalnya terlihat kecil. Maka dari itu, diharuskan menghentikan pendarahan dengan memberikan tekanan selama 5 hingga 30 menit menggunakan kain bersih atau gulungan kasa. Apabila tidak ada, dapat dilakukan dengan tangan secara langsung, namun harus dipastikan bahwa tangan telah dicuci (dalam keadaan sangat bersih)

2. Nilai Kerusakan

   Ketika melakukan perawatan atau pertolongan pertama luka, menilai tingkat kerusakan luka dibutuhkan untuk memutuskan urgensi kebutuhan perawatan medis profesional untuk menangani luka tersebut. Ada dua kondisi luka yaitu yang berpotensi mengancam jiwa dan yang berpotensi menimbulkan kerisukan permanen. Kondisi luka yang berpotensi mengancam jiwa adalah sebagai berikut:

   (a) Pendarahan tidak berhenti bahkan setelah 30 menit memberi tekanan pada luka.
   (b) Bila jari tangan dan kaki menjadi dingin dan berubah warna.
   (c) Terdapat luka terlalu dalam pada rongga dada atau perut.
   (d) Luka di leher yang melibatkan jalan napas.

   Selanjutnya, kondisi luka yang berpotensi menimbulkan kerusakan permanen adalah sebagai berikut:

   (a) Fraktur terbuka yaitu bagian tulang yang patah dan menembus dari dalam hingga ke permukaan kulit, kondisi ini membuat luka sangat berisiko terkena infeksi.
   (b) Luka dengan kemungkinan kerusakan saraf yaitu ketika area distal ke luka (sisi jauh dari jantung) mati rasa.
   (c) Kerusakan tendon yaitu ketika luka menyebabkan tidak bergeraknya anggota gerak yang lebih rendah dari luka.

3. Bersihkan Luka

Terdapat tiga proses dalam tahapan pembersihan luka, yaitu:

(a) Mengangkat benda asing pada luka, direkomendasikan menggunakan pinset bila ada, namun apabila benda tersebut cukup besar, jangan dicabut sendiri dan minta pertolongan medis untuk melakukannya

(b) Gunakan sabun dan air untuk membersihkan permukaan luka dan kapas untuk area yang sulit dijangkau

(c) Alirkan cairan pada luka untuk membersihkannya dengan menggunakan air minum atau larutan garam (cairan infus NaCl 0,9%)

4. Tentukan Pengobatan

Pada tahap ini harus menilai luka apakah perlu menjahit luka yang terjadi, luka yang kecil, tidak dalam, tidak di atas sendi, atau kurang dari 6 mm akan sembuh dengan baik selama luka dijaga tetap bersih dan tertutup.

5. Menutup atau Menjahit Luka

Ada beberapa opsi untuk menutup luka, di antara lain adalah melakukan penjahitan, staples, lem atau selotip, atau rambut dan tali (untuk luka di kepala).

6. Perhatikan Tanda Infeksi

Tahapan akhir pada perawatan luka adalah dengan memperhatikan apakah terjadi infeksi atau tidak. Berbagai tanda infeksi kulit yang dapat diperhatikan adalah sebagai berikut:

(a) Peningkatan kehangatan atau kemerahan pada sekitar luka

(b) Bintik merah dan bengkak di kulit

(c) Garis-garis merah mengarah ke jantung

(d) Nanah

(e) Demam

c. Luka Tusuk

Ketika luka tusuk terjadi, pastikan untuk tidak mengambil atau mencabut benda yang menyebabkan luka apabila masih tertancap pada luka karena dapat menyebabkan pendarahan yang lebih serius. Pertolongan pertama pada luka tusuk tergantung pada kondisi dan/atau area terjadinya luka. Beberapa kondisi tersebut adalah (Deeny, 1994):

1. Apabila luka terjadi di area dada, yang mungkin menyebabkan kesulitan bernapas, cobalah meletakkan tangan langsung pada luka. Lalu perban luka dengan tekanan ringan untuk membuat segel

kedap udara pada luka. Pasien dengan luka di dada ini akan lebih nyaman dalam keadaan duduk untuk memudahkan bernapas.

2. Apabila luka menembus perut, kemungkinan terjadi pendarahan internal menjadi lebih tinggi sehingga tujuan pertolongan pertama pada kondisi ini adalah mencoba meminimalkan syok. Pada kondisi ini, baringkan pasien mendatar, tekuk, dan dukung lutut jika memungkinkan. Menaikkan dan menopang lutut dapat meredakan ketegangan pada luka. Selanjutnya, perban luka namun jangan kencangkan terlalu erat.
3. Apabila pasien kehilangan kesadaran, tempatkan pasien pada posisi pemulihan dan terus periksa denyut nadi dan pernapasan sesering mungkin. Korban dengan luka tusuk tembus baik dada atau perut mungkin memerlukan resusitasi yang dilakukan oleh bantuan medis profesional.
4. Apabila luka tusuk terjadi di kepala, segera langsung bawa pasien ke fasilitas kesehatan terdekat (lebih baik rumah sakit) untuk bantuan medis profesional.

d. Luka Bakar

Pada luka bakar, terdapat tiga kondisi utama yang membedakan pertolongan pertama, yaitu pada luka bakar besar yang memerlukan bantuan medis segera, luka bakar besar (*major*) dan luka bakar kecil (*minor*) (Mayo Clinic, 2022).

Kondisi luka bakar yang memerlukan perawatan medis secepat mungkin adalah sebagai berikut:

1. Luka bakar dalam yang meliputi seluruh lapisan kulit.
2. Menyebabkan kulit kering dan bersisik/kasar.
3. Tampak hangus atau bercak putih, coklat, atau hitam.
4. Berdiameter lebih dari 3 inci (sekitar 8 cm).
5. Menutupi tangan, kaki, wajah, selangkangan, pantat, atau persendian utama, atau melingkari lengan dan tungkai.
6. Disertai dengan menghirup asap (ketika terjadi kebakaran).
7. Mulai terjadi pembengkakan dengan sangat cepat.

Untuk luka besar (*major*), segera panggil pertolongan medis, dan selama menunggu, lakukan berbagai hal di bawah ini:

1. Lindungi pasien luka bakar dari bahaya lainnya
2. Pastikan pasien tetap bernapas
3. Lepaskan perhiasan, ikat pinggang, dan benda ketat lainnya dari korban

4. Tutupi area luka bakar secara longgar dengan kain kasa atau kain bersih
5. Angkat area yang terkena luka bakar lebih tinggi dari posisi jantung (jika mungkin)
6. Awasi pasien apakah mengalami syok atau tidak

Untuk luka kecil (*minor*), lakukan beberapa pertolongan pertama di bawah ini:

1. Dinginkan area yang terluka dengan mengaliri luka dengan air bersih selama 10 menit
2. Lepaskan perhiasan atau benda ketat lainnya dari area yang terkena luka bakar
3. Jangan memecahkan kulit yang melepuh (berisi cairan) untuk menghindari infeksi. Namun, apabila pecah, bersihkan area tersebut dengan air mengalir dan oleskan salep antibiotic
4. Oleskan losion setelah luka bakar dingin. Losion bisa dari lidah buaya atau mentega kakao.
5. Perban luka bakar dengan perban yang bersih, namun jangan terlalu ketat.
6. Jika perlu, konsumsi pereda nyeri tanpa resep seperti ibuprofen.

e. Memar

Beberapa pertolongan pertama atau perawatan pada memar adalah sebagai berikut (Mayo Clinic, 2022a):

1. Turunkan bagian tubuh yang memar di bawah jantung (jika mungkin)
2. Mengkompres memar dengan kantong es selama kurang lebih 20 menit. Ulangi tahapan ini setelah satu atau dua hari hingga bengkak dan rasa sakit berkurang.
3. Bila terjadi pembengkakan, tutup memar dengan perban elastis namun jangan terlalu kencang.
4. Pertimbangkan mengonsumsi pereda nyeri tanpa resep jika diperlukan
5. Istirahatkan dan angkat area yang memar untuk mencegah pembengkakan dan menghilangkan rasa sakit

2.3 *Artificial Inteetrlligence* (AI)

Artificial Intelligence (AI) atau kecerdasan buatan ditemukan dengan latar belakang kemampuan manusia yang dapat memahami berbagai hal. Namun hingga saat ini, masih belum ada satu mesin pun yang berhasil menyaingi atau

bahkan menyamai kecerdasan manusia secara keseluruhan. Banyak ilmuan yang telah melakukan riset guna mempelajari kecerdasan manusia. Hingga akhirnya, terbitlah sebuah ilmu yang mempelajari kecerdasan manusia yang disebut dengan AI. Dengan bantuan teknologi yang sudah sangat pesat ini, AI menghadirkan berbagai piranti cerdas yang membantu banyak kegiatan manusia yang mana masih dipelajari hingga sekarang dan bahkan dikembangkan lebih maju lagi (Cholissodin & Soebroto, 2021).

Maka dari itu, AI adalah ilmu atau teori dan pengembangan pada sistem komputer sehingga dapat menyelesaikan berbagai tugas yang biasanya memerlukan kecerdasan manusia, termasuk persepsi atau identifikasi visual, pengenalan suara, dan pembuatan keputusan. AI sendiri memiliki berbagai tipe, dimana salah satunya adalah machine learning (ML) atau pembelajaran mesin.

2.4 *Machine Learning* (ML)

Machine Learning (ML) atau pembelajar mesin adalah salah satu dari banyak tipe kecerdasan buatan yang banyak digunakan untuk pengenalan pola. ML ditujukan untuk mensimulasikan pengalaman mental manusia dan pola praktiknya, seperti kemampuan untuk belajar, menilai, dan bereaksi terhadap situasi (Aggarwal et al., 2022).

Ilmu ini didasari dengan tingkah laku manusia yang biasa memprediksi berbagai macam hal berdasarkan sedikit hingga banyaknya pengalaman yang dialami, seperti memprediksi rasa semangka yang manis dengan ciri-ciri buah semangka dengan kulit berwarna hijau, akar keriting, dan suara yang meredam. Ciri-ciri tersebut didapatkan dari pengalaman-pengalaman manusia yang biasa mendapatkan buah yang manis dengan ciri-ciri di atas. Sama dengan pengalaman tersebut, bahwa pengalaman belajar manusia dapat mengarah ke nilai akademik yang bagus. Dengan begitu, prediksi manusia dapat dipercaya karena pembelajaran dari pengalaman yang dilalui dan keputusan-keputusan yang diambil berdasarkan pengalaman.

Ketika manusia dapat belajar dari pengalaman, komputer pun dapat melakukan hal yang sama. Namun, tidak seperti kode komputer biasa yang dibuat oleh programmer yang didesain untuk memberikan keluaran sesuai dengan masukan yang diberikan, ML menggunakan data sebagai “pengalaman” untuk menghasilkan kode statistik yang mana dengan mempelajari proses atau pola dari pemberian masukan sebelumnya hingga akhirnya memberikan keluaran yang benar.

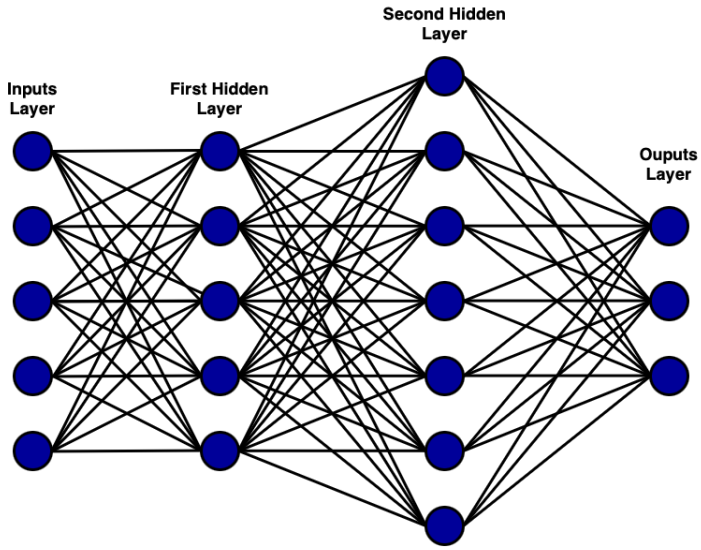
Pada sistem komputer, pengalaman tersebut berbentuk data, dan tugas utama dari ML adalah mengembangkan algoritma pembelajaran tersebut hingga menghasilkan model dari data tersebut (*Zhi-Hua Zhou - Machine Learning-*

Springer Singapore (2021), n.d.) Data yang diperlukan oleh model ML adalah data yang valid sebagai bahan pembelajaran (ketika proses *training*) sebelum akhirnya digunakan oleh model ML ketika proses *testing* agar menghasilkan hasil yang optimal (Cholissodin & Soebroto, 2021). Terlebih lagi, dengan data yang valid, serta kemampuan ML yang dapat memberikan hasil dan prediksi yang optimal, permasalahan dengan dimensi tinggi pun dapat terselesaikan.

2.5 *Deep Learning*

Machine Learning sangat membantu dalam pemecahan masalah dengan mengolah data, guna memperoleh pengetahuan akan pembelajaran yang telah dilakukan. Namun dengan metode ML yang tradisional ini, pengetahuan yang telah didapat tidak dapat diperbaharui apabila tidak dilakukan pembelajaran ulang dengan modifikasi atau penambahan data sebelumnya. Maka dari itu, muncullah metode ML yang lebih modern dimana pengetahuan yang telah didapat melalui suatu pembelajaran, dapat diperbarui dengan menggunakan pengetahuan sebelumnya dan menambah atau mengubah data yang digunakan. Salah satu dari metode ML yang modern tersebut adalah *Deep Learning* (DL). Dengan *deep learning*, pengetahuan dapat selalu diperbarui dengan lebih cepat. Pembelajaran yang lebih cepat ini dikarenakan oleh keterlibatan data yang dapat ditambah secara berkala dengan ukuran yang lebih kecil sesuai dengan kemampuan komputer yang digunakan. Hasil pembelajaran tersebut pun bisa lebih akurat karena kemampuannya yang dapat mengekstraksi fitur secara mandiri dan menggunakan data yang lebih banyak (*Big Data*).

Deep Learning terdiri dari algoritma pemodelan dengan abstraksi tingkat tinggi pada pada data yang menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mandalam. Model yang dibangun menggunakan metode *deep learning* ini, dibangun berdasaran jaringan saraf tiruan atau *Neural Network* dimana jika suatu jaringan memiliki lebih dari 3 lapisan (*layer*), maka jaringan tersebut disebut dengan *Deep Network* (Cholissodin & Soebroto, 2021). Ilustrasi lapisan pada *Deep Network* dapat dilihat pada Gambar 6.



Gambar 6. Ilustrasi Arsitektur pada *Deep Network*

2.6 *Convolutional Neural Network*

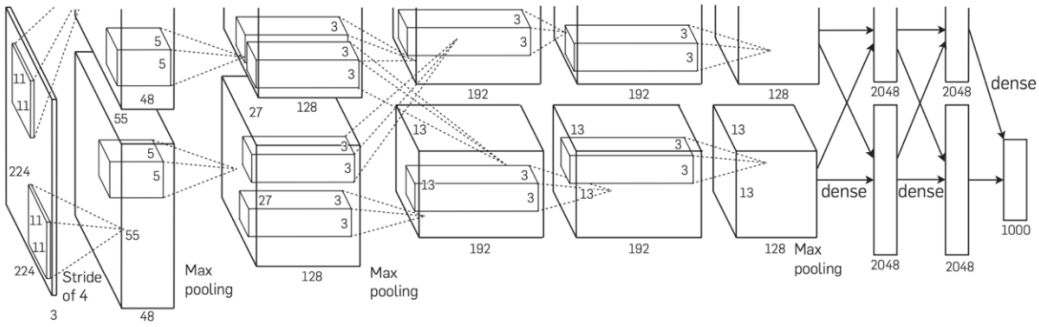
Convolutional Neural Network (CNN) adalah salah satu algoritma paling popular dalam *deep learning*. Sebutan "convolutional" pada CNN ini diambil dari operasi linear matematika antar matriks yang disebut dengan konvolusi atau "convolution". Oleh karena itu, CNN pun diindikasi pasti menggunakan, paling sedikit, satu konvolusi pada lapisannya (Lecun et al., 2015).

CNN adalah algoritma yang digunakan untuk mengklasifikasi data yang berlabel. Dimana terdapat dapat yang dilatih dan variabel yang dijadikan sebagai target yang berarti, CNN membantu dalam mengelompokkan sebuah data pada data yang telah ada (Mashita, 2020). CNN mengambil masukan berupa citra dan kemudian menerapkan operasi konvolusi pada masukan tersebut dengan menggunakan fitur-fitur khusus. Dalam proses ini, CNN menghasilkan fitur-fitur khusus pada citra yang diproses. Selanjutnya, CNN melakukan proses *pooling* untuk mengurangi dimensi data dan meningkatkan efisiensi komputasi, sebelum melakukan klasifikasi atau regresi dengan menggunakan lapisan-lapisan terakhir dari jaringan.. Lebih jelasnya, pada setiap lapisan (*layer*), terdapat ekstraksi fitur yang berbeda fokus. Dengan contoh, lapisan pertama akan berfokus pada identifikasi tepian, lapisan kedua berfokus pada identifikasi bentuk, dan sebagainya (Cholissodin & Soebroto, 2021). CNN memiliki tujuan akhir yaitu mendeteksi label dari data masukan. Dalam prosesnya, label dikenali sesuai dengan fitur data yang telah ada. Semakin dekat kemiripan fitur pada setiap lapisan dengan fitur salah satu kategori, maka semakin besar kemungkinan hasil deteksi akan

mengelompokkan data masukan ke dalam kategori yang paling mirip pada setiap fitur ekstraksi pada lapisan-lapisan CNN.

Terdapat sebuah penelitian yang membahas arsitektur CNN dengan judul "ImageNet Classification with Deep Convolutional Neural Networks" yang ditulis oleh Alex Krizhevsky, Ilya Sutskever, dan Geoffrey Hinton pada tahun 2017. Jurnal ini membahas tentang arsitektur CNN yang digunakan dalam kompetisi *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) tahun 2012, di mana CNN berhasil mencapai tingkat akurasi pengenalan citra yang lebih tinggi dibandingkan dengan metode-metode sebelumnya (Krizhevsky et al., 2017).

Arsitektur CNN yang digunakan dalam jurnal tersebut terdiri dari 5 lapisan konvolusi dan *pooling*, diikuti oleh 3 lapisan terhubung penuh (*fully connected*). Lapisan pertama dari jaringan tersebut adalah lapisan konvolusi yang terdiri dari 96 *filter* dengan ukuran 11x11. Lapisan kedua dan ketiga juga merupakan lapisan konvolusi dengan masing-masing 256 dan 384 filter dengan ukuran 5x5. Lapisan keempat dan kelima juga merupakan lapisan konvolusi dengan masing-masing 384 dan 256 filter dengan ukuran 3x3. Setiap lapisan konvolusi diikuti oleh lapisan max-pooling berukuran 3x3. Setelah melalui lapisan konvolusi dan pooling, data di-*flatkan* dan dihubungkan ke lapisan-lapisan terhubung penuh untuk dilakukan klasifikasi atau regresi. Salah satu arsitektur CNN dapat dilihat pada Gambar 7.



Gambar 7. Arsitektur CNN pada Kompetisi ImageNet *Large-Scale Visual Recognition Challenge* (Krizhevsky et al., 2017)

Jurnal tersebut pun membuktikan bahwa penggunaan algoritma CNN sangat membantu dalam mengatasi masalah pengenalan citra yang kompleks, terutama dengan dataset yang besar seperti ImageNet. Sejak itu, banyak variasi arsitektur CNN yang dikembangkan sehingga dapat diimplementasikan pada berbagai kasus pengenalan citra lainnya, seperti RestNet, Inception, dan VGGNet yang disebut dengan metode *transfer learning*.

2.7 *Transfer Learning*

Transfer learning pada CNN adalah teknik yang memanfaatkan model CNN yang sebelumnya telah dilatih menggunakan dataset yang lebih besar, untuk mengekstraksi fitur pada gambar/citra pada dataset yang lebih kecil atau dataset dalam kasus yang berbeda (Tammina, 2019). Proses ini dilakukan dengan mengambil model yang telah ada dan menambahkan beberapa lapisan *fully-connected* baru untuk melakukan klasifikasi pada dataset, baik dengan kasus yang sama dengan dataset yang baru, maupun dengan kasus yang baru. Model yang telah ada ini disebut dengan *pre-trained model* yang artinya adalah model yang telah dilatih.

Pada penelitian yang dilakukan oleh Simonyan dan Zisserman tahun 2015, penggunaan *transfer learning* dengan menggunakan arsitektur dalam (*Deep* CNN), seperti VGGNet, pada *dataset ImageNet* berhasil meningkatkan akurasi bahkan dengan *dataset* baru yang lebih kecil. Hasil penelitian menunjukkan bahwa *transfer learning* dengan menggunakan model CNN yang telah dilatih sebelumnya mampu meningkatkan akurasi klasifikasi pada dataset baru secara signifikan. Hal ini dikarenakan model CNN yang sudah dilatih sebelumnya telah belajar untuk mengekstraksi fitur-fitur yang sangat berguna dari gambar pada dataset besar, yang dapat diterapkan pada dataset baru dengan cara melakukan *fine-tuning* pada beberapa lapisan terakhir untuk menyesuaikan dengan kasus baru, seperti jumlah kelas baru, atau meminimalkan *loss* pada *dataset* baru, dan sebagainya (Simonyan & Zisserman, 2015).

1. **Inception-v3**

   Inception-v3 adalah salah satu *pre-trained model* yang sangat populer untuk melakukan klasifikasi beberapa kelas gambar. Model ini merupakan arsitektur Convolutional Neural Network (CNN) yang diciptakan oleh Google pada tahun 2015 sebagai pengembangan dari versi sebelumnya. Dengan kompleksitas arsitektur yang dimiliki oleh Inception, pengubahan arsitektur tersebut akan menjadi sulit. Jika dilakukan secara naif, kemungkinan kehilangan sebagian besar hasil komputasi akan terjadi. Selain itu, penelitian sebelumnya pada arsitektur GoogleNet, tidak dijelaskan faktor apa saja yang memengaruhi keputusan desain arsitektur GoogleNet (nama lain Inception-v1) (Szegedy et al., 2016). Maka dari itu, dikembangkan lah Inception-v3 oleh Szegdy et al untuk meningkatkan jaringan konvolusi namun dengan cara yang lebih efisien. Berikut adalah perbandingan *top*-5 *error* dari tiga versi model inception:

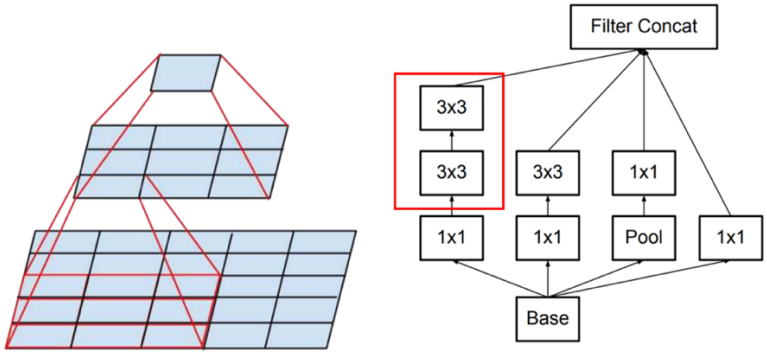
Tabel 1. *Top-5 Error* dari Tiga Versi Model Inception Sebelumnya

No	Model	Top-5 Error
1	GoogLeNet (Inception-v1)	6,67%
2	Inception-v2	4,9%
3	Inception-v3	3,58%

Inception-v3 terkenal karena menggunakan teknologi "inception modules"yang memproses gambar secara efisien dengan menggunakan filter dari berbagai ukuran. Inception-v3 memiliki sejumlah besar layer yang dalam (*deep*), dan telah dilatih pada dataset ImageNet yang terdiri dari lebih dari 1,2 juta gambar dengan 1.000 kelas. Inception-v3 terkenal karena kemampuannya menghasilkan akurasi yang tinggi dalam tugas klasifikasi gambar. Model ini berhasil memenangkan kompetisi ImageNet tahun 2015 dan berhasil mengalahkan model-model lainnya yang ada pada waktu itu.

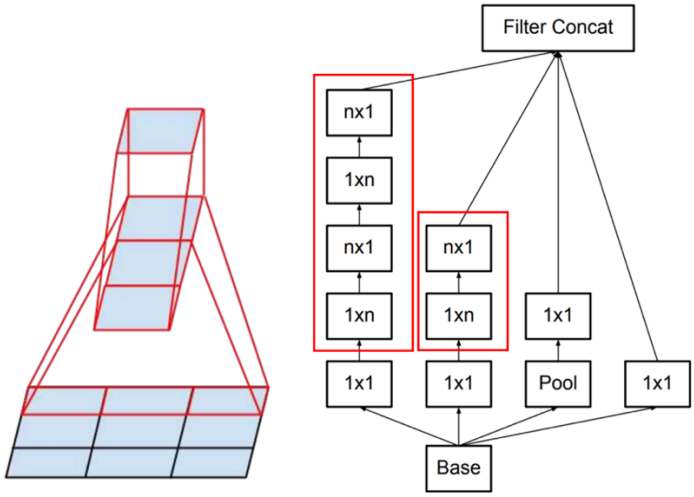
Inception *Module A*

Pada modul A, dilakukan faktorisasi dari yang awalnya hanya satu lapisan (*layer*) dengan *filter* 5x5 yang menghasilkan parameter sejumlah 25, menjadi 2 *layer* dengan *filter* 3x3 yang menghasilkan parameter yang lebih sedikit yaitu 18. Ilustrasi modul A pada Inception-v3 ini dapat dilihat pada Gambar 8.



Gambar 8. Inception-v3 *Module* A (Szegedy et al., 2016)

Inception *Module B*

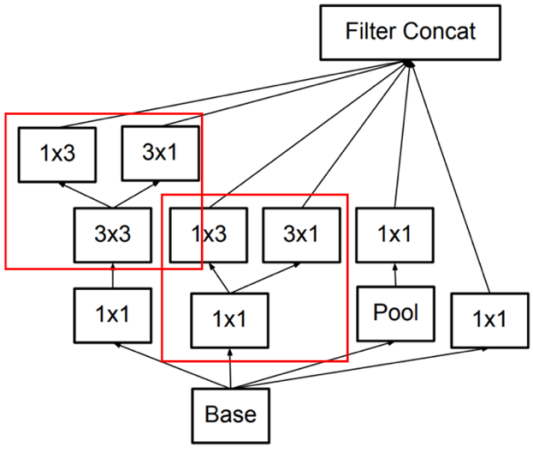


Gambar 9. Inception-v3 *Module* B (Szegedy et al., 2016)

Pada modul B, satu *layer* dengan *filter* nxn disederhanakan menjadi dua *layer* dengan filter 1xn dan nx1 sehingga parameter yang dihasilkan lebih sedikit. Pada inception-v3, nilai n tersebut adalah 7 untuk *grid* 17x17. Ilustrasi modul B pada Inception-v3 ini dapat dilihat pada Gambar 9.

Inception *Module C*

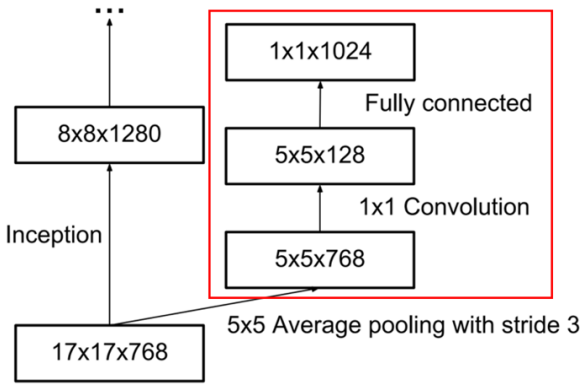
Inception Module C adalah hasil dari penggabungan atau implementasi teknik optimasi pada modul A dan B. Ilustrasi modul C pada Inception-v3 ini dapat dilihat pada Gambar 10.



Gambar 10. Inception-v3 *Module* C (Szegedy et al., 2016)

Auxiliary Classifier

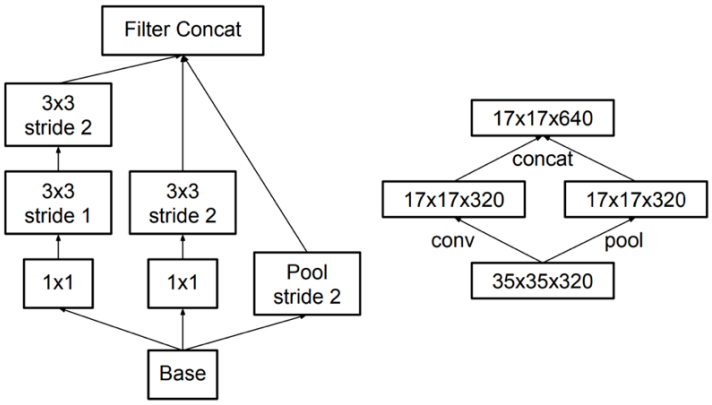
Auxiliary classifier pada inception-v3 ini berperan dalam regularisasi model sehingga membantu mengurangi *overfitting* dan meningkatkan generalisasi. Ilustrasi *auxiliary classifier* pada Inception-v3 ini dapat dilihat pada Gambar 11.



Gambar 11. *Auxiliary Classifier* pada Inception-v3 (Szegedy et al., 2016)

Gride Size Reduction

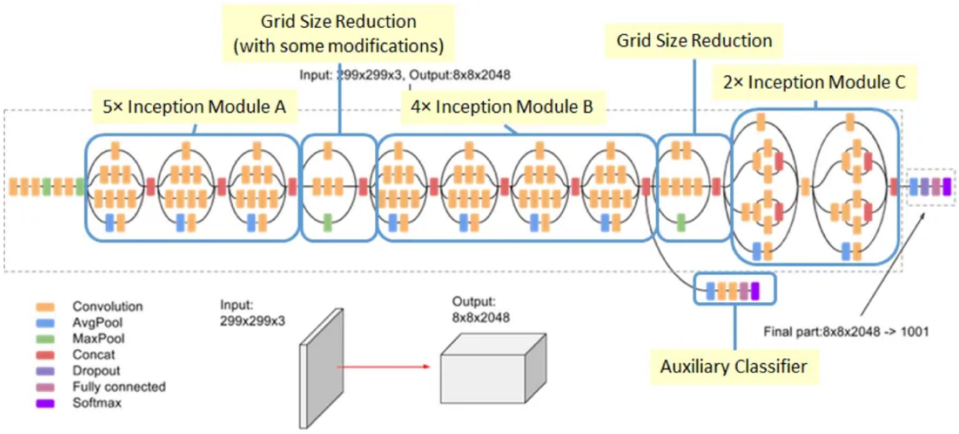
Proses *grid size reduction* biasanya melibatkan operasi konvolusi dengan kernel besar, seperti kernel 3x3 dengan langkah (*stride*) 2 atau *pooling* maksimum dengan ukuran kernel 3x3 dan langkah 2. Kedua operasi ini membantu dalam mengurangi dimensi spasial dari peta fitur, sehingga menghasilkan representasi yang lebih kompak. Ilustrasi *gride size reduction* pada Inception-v3 ini dapat dilihat pada Gambar 12.



Gambar 12. *Grid Size Reduction* pada Inception-v3 (Szegedy et al., 2016)

Arsitektur Keseluruhan Inception-v3

Dengan menggabungkan berbagai insepsi dan modul di atas, pada Gambar 14 menunjukkan arsitektur keseluruhan dari *pre-trained model* Inception-v3.



Gambar 13. Arsitektur *Pre-trained Model* Inception-v3 (Tsang, 2018)

2. VGG-16

Selain Inception-v3, VGG-16 adalah *pre-trained model* yang juga memiliki kepopuleran yang tinggi untuk digunakan dalam bidang *computer vision* dan pengolahan citra. VGG-16 dikembangkan oleh tim peneliti asal Unversitas Oxford, Inggris pada tahun 2014 (Simonyan & Zisserman, 2015) dari versi-versi sebelumnya. Berikut adalah hasil top-5 error dari tiga versi berbeda VGG:

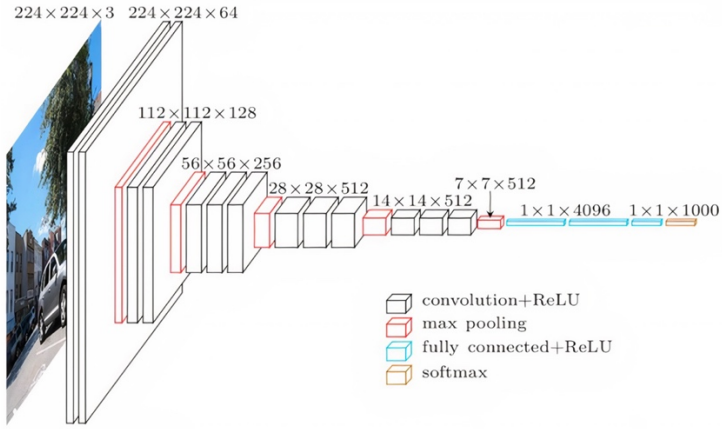
Tabel 2. *Top-5 Error* dari Tiga Versi Model VGG Lainnya

No	Model	Top-5 Error
1	VGG-13	9,6%
2	VGG-16	7,5%
3	VGG-19	7,5%

Berdasarkan hasil pada Tabel 2, terdapat persamaan hasil oleh VGG-16 dan VGG-19. Namun, pada penelitian ini, penulis memustukan untuk menggunakan VGG-16 karena kompleksitas arsitektur yang lebih sedikit sehingga lebih sedikit memakan sumber daya komputasi. VGG-16 adalah hasil yang dibuat oleh Simonyan dan lainnya dalam mengikuti *challenge* atau kompetisi ImageNet *Large Scale Visual Recognition Challenge* (ILSVRC) pada tahun 2014. Kompetisi tersebut bertujuan untuk mengembangkan model yang dapat mendeteksi dan mengklasifikasikan objek berdasarkan dataset ImageNet. VGG-16 merupakan model terbaik kedua yang berhasil menyelesaikan *challenge* tersebut.

Setelah itu, arsitektur VGG-16 menjadi terkenal dan umum digunakan oleh praktisi dan peneliti dalam bidang pengolahan citra dan visi komputer untuk melakukan klasifikasi gambar pada dataset yang besar seperti ImageNet. Hal ini karena arsitektur VGG-16 telah terbukti

efektif dan mampu mencapai tingkat akurasi yang sangat tinggi dalam tugas klasifikasi gambar. Di bawah ini adalah arsitektur dari VGG-16:



Gambar 14. Arsitektur *Pre-trained Model* VGG-16 (Great Learning, 2021)

Berikut adalah beberapa informasi mengenai arsitektur VGG16 (Great Learning, 2021):

1. Arsitektur pada VGG-16 terdiri dari 21 *layer* yang dengan perinciannya adalah 13 *layer* konvolusi, 5 *layer max pooling*, dan 3 *dense layer.* Namun, dari total lapisan tersebut, hanya 16 *layer* yang memiliki bobot sehingga penamaannya dari *pre-trained model* ini adalah VGG-16.
2. Tensor masukan pada VGG-16 berukuran 224, 224 dengan 3 saluran RGB.
3. Hal yang unik dari VGG-16 adalah penggunaannya akan lapisan (layer) konvolusi filter 3x3 dengan langkah (stride) 1, dan selalu menggunakan lapisan pooling dan padding yang sama dari filter 2x2 dengan langkah 2, alih-alih menggunakan hyper-parameter yang berjumlah besar.
4. Pada arsitektur VGG-16, lapisan-lapisan konvolusi dan max pooling-nya diatur dengan konsisten secara keseluruhan.
5. Lapisan konvolusi pertama pada arsitektur tersebut memiliki 64 jumlah filter, konvolusi kedua memiliki 128 filter, konvolusi ketiga memiliki 256 filter, dan konvolusi kelima yaitu yang terakhir memiliki 512 filter.
6. Setelah seluruh lapisan konvolusi dan pooling, terdapat tiga lapisan fully-connected yang mana dua lapisan pertamanya memiliki 4096 saluran dan yang ketiga memiliki 1000 saluran sebagai lapisan dengan tujuan klasifikasi 1000 arah pada kompetesi ILSVRC ketika pre-trained model ini diperlombakan.
7. Dan lapisan terakhir pada VGG-16 adalah lapisan softmax.

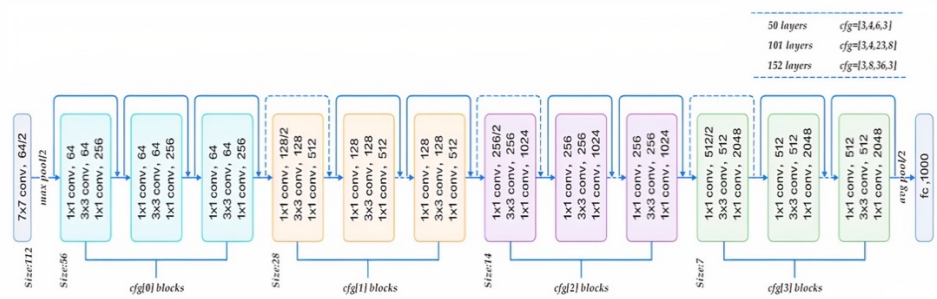
3. ResNet-50

Selain Inception-v3 dan VGG-16, ResNet juga cukup populer untuk melakukan klasifikasi beberapa kelas gambar. Dibuat oleh *Microsoft Research* pada tahun 2015, model ResNet menggunakan teknologi "Residual Network" untuk mengatasi masalah "vanishing gradient" yang terjadi ketika algoritma pembelajaran mendalam sulit di-optimalkan pada jaringan yang sangat dalam. ResNet memiliki berbagai versi dalam pengembangannya. Berikut adalah perbandingan dari hasil pelatihan antara tiga versi berbeda dari model *ResNet*:

Tabel 3. *Top-5 Error* dari Tiga Versi Model ResNet Sebelumnya

No	Model	Top-5 Error
1	ResNet-34 B (*version1*)	5,71%
2	ResNet-34 C (*version* 2)	5,6%
3	ResNet-50	5,25%

Berdasarkan hasil di atas, penulis pun memutuskan menggunakan model ResNet dengan 50 *layer*. Arsitektur dari ResNet-50 dapat dilihat pada Gambar 15. ResNet terdiri dari banyak *layer* yang sangat dalam dan memiliki sejumlah besar parameter dan dikembangkan lagi hingga memiliki 50 lapisan (*layer*) pada arsitekturnya sehingga disebut dengan ResNet-50. Sama seperti Inception-v3 dan VGG-16, ResNet-50 ini juga dilatih lagi dengan data ImageNet. Hasilnya, ResNet-50 mampu mencapai akurasi yang sangat tinggi dalam tugas klasifikasi gambar pada *dataset* besar seperti ImageNet, hingga pengenalan wajah dengan tingkat akurasi yang tinggi (He et al., 2016b).



Gambar 15. Arsitektur *Pre-trained Model* ResNet-50 (Rastogi, 2022)

ResNet memiliki berbagai variasi berdasarkan jumlah lapisan pada arsitekturnya. Dapat dilihat di atas, setiap blok konvolusi memiliki jumlah yang berbeda tergantung dengan jumlah lapisan yang digunakan. Pada ResNet-50 yang memiliki 50 lapisan, sesuai dengan informasi di kanan atas, pada blok konvolusi pertama, terdapat 3 lapisan konvolusi. Lalu pada blok kedua, terdapat 4 lapisan konvolusi,

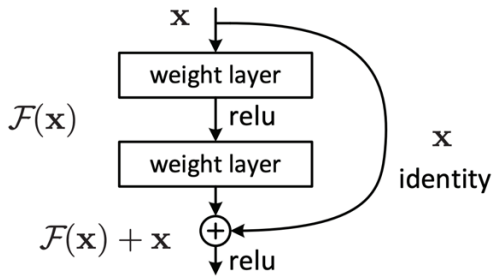
dan seterusnya. Informasi ini didapat berdasarkan penelitian yang dilakukan oleh pencetus ResNet, yang dapat dilihat pada Gambar 16.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				

Gambar 16. Arsitektur Berbagai Variasi ResNet (He et al., 2016a)

Kembali pada Gambar 15, setelah setiap blok konvolusi terdapat proses melangkahi lapisan di antaranya (ditandai dengan panah bergaris titik-titik). Proses ini adalah konsep utama dari ResNet yang disebut dengan "identity shortcut connections" yang menggunakan teknik yang disebut dengan blok residual yang dapat dilihat pada Gambar 17.

Secara sederhana, pencetus ResNet percaya bahwa pemetaan secara resudial lebih mudah diaplikasikan daripada pemetaan aktual sehingga proses resudial ini digunakan pada seluruh lapisan pada ResNet. Selain itu, pencetus ResNet juga percaya bahwa banyaknya lapisan yang ditumpuk tidak akan mengurangi kinerja model.



Gambar 17. Blok Resudial (He et al., 2016a)

2.8 Metode *Hash* pada Pengecakan Duplikasi Data

Metode hash merupakan sebuah teknik pengolahan data yang digunakan untuk menghasilkan nilai hash atau *checksum* yang unik dari suatu data, seperti *file* atau pesan. Metode hash banyak digunakan dalam berbagai aplikasi, seperti verifikasi integritas data, autentikasi, enkripsi, dan pengecekan duplikasi data, termasuk pada dataset gambar. Pada penelitian ini, metode hash digunakan untuk melakukan pengecekan duplikasi gambar dengan

mengidentifikasi gambar yang serupa atau identik secara efisien tanpa harus membandingkan setiap piksel gambar. Salah satu penelitian yang membahas tenatang penggunaan metode hash pada dataset gambar adalah "Efficient Incremental Near Duplicate Detection Based on Locality Sensitive Hashing" oleh Fisichella et al. Penelitian tersebut mengusulkan penggunaan teknik hash untuk mengidentifikasi gambar duplikat (Fisichella et al., 2010).

Pada penelitian ini, metode hash diaplikasikan dengan menggunakan salah satu algoritma yang populer yaitu MD5. Nilai hash yang dihasilkan menggunakan algoritma *hashing* kriptografi MD5 memiliki panjang 128-bit. Berdasarkan penelitian "The Authenticity of Image Using Hash MD5 and Steganography Least Significant Bit" oleh Khairina et al., penggunaan algoritma MD5 ini membantu menghasilkan nilai hash yang unik untuk setiap gambar yang akan divalidasi (Khairina et al., 2018).

2.9 Penanganan *Imbalanced Data*

Imbalanced data atau data yang tidak seimbang adalah kondisi dimana terjadi perbedaan jumlah yang cukup besar antara kelas minoritas dan mayoritas. Ketika mengembangkan sebuah model yang efektif, dengan menggunakan algoritma *machine learning*, akan sangat dilakukan apabila tanpa mempertimbangkan prapemrosesan data (*data pre-processing*) untuk menyeimbangkan data yang tidak seimbang. Dengan keterbatasan data yang dimiliki, data yang tidak seimbang (*imbalanced data*) pada dataset yang akan diuji akan sangat berpengaruh pada hasil akurasi proses identifikasi atau klasifikasi menggunakan model *machine learning* (*classifier*). Untuk mengatasi permasalahan tersebut, terdapat berbagai teknik *undersampling* atau pengurangan dan *oversampling* atau penambahan data pada *dataset* yang ada (Shelke et al., 2017). *Undersampling* dilakukan untuk mengurangi jumlah data pada kelas mayoritas, dan *oversampling* dilakukan untuk menambahkan jumlah data pada kelas minoritas.

Salah satu metode *undersampling* yang dilakukan adalah dengan memanfaatkan teknik *clustering* pada tahapan *pre-processing data*, dengan menggunakan strategi penentuan pusat kluster untuk mewakili kelas mayoritas, dan strategi lainnya yaitu menggunakan tetangga terdekat dari pusat kluster (Lin et al., 2017). Selain memanfaatkan teknik *clustering,* proses *undersampling* juga dapat dilakukan dengan mengurangi data pada kelas mayoritas secara acak.

Pada kelas yang memiliki data yang sedikit, dapat dilakukan proses *oversampling* dengan berbagai metode. Salah satu yang paling populer adalah metode SMOTE (*The Synthetic Minority Oversampling Technique*). SMOTE kerap

dianggap sebagai standar *framework* untuk mengatasi *imbalanced dataset*. Hal ini dikarenakan kesederhanaan pada desain prosedur dan kemampuannya dalam menyelesaikan berbagai *imbalanced dataset* pada kasus yang berbeda. SMOTE juga memiliki berbagai pendekatan untuk mengatasi permasalahan *imbalanced data* dan sangat berkontribusi pada berbagai pembelajaran seperti *supervised learning*, *semi-supervised learning*, klasifikasi multikelas, dan sebagainya. Metode SMOTE menerapkan proses pembuatan sampel sintetik baru, alih-alih hanya replikasi sederhana dari sampel kelas minoritas. Sampel-sampel minoritas tersebut digabungkan dan dipelajari hingga muncullah sampel baru untuk menambah data pada kelas minoritas (Fernández et al., 2018).

SMOTE terbukti menjadi teknik yang efektif untuk menciptakan sampel tambahan dari kelas minoritas dan menyamakan ukuran *dataset* dengan kelas mayoritas, guna mengatasi ketidakseimbangan kelas ini. Metode ini dapat meningkatkan kinerja klasifikasi pada kelas dengan jumlah data yang sedikit dengan memperbesar jumlah *dataset* sehigga dapat mengurangi dampak negatif dari sampel kecil tersebut sepertinya kurangnya kemampuan model dalam mengidentifikasi (akurasi yang rendah) atau *overfitting* dimana model menjadi terlalu bias pada label tertentu. SMOTE pun terbukti dapat meningkatkan akurasi klasifikasi tidak hanya pada data kecil atau kelas minoritas, tetapi juga pada data berukuran besar atau kelas mayoritas (Elreedy & Atiya, 2019).

2.10 Metrik Perhitungan Performa Deteksi Citra

Model yang dihasilkan dengan proses *machine learning* pasti sangat membantu dalam mendeteksi dan mengklasifikasikan berbagai objek terutama berbasis citra. Namun, keberhasilan sebuah model yang dibuat memilik parameter atau metrik yang mengukur tingkat keberhasilan tersebut. Dengan metode CNN, berbagai metrik disediakan untuk mengukur hasil atau tingkat keberhasilan sebuah model dalam mendeteksi. *Confusion matrix* dan *accuracy* adalah variabel yang menentukan tingkat keberhasilan dan akurasi model dalam mendeteksi dan mengklasifikasikan data masukan.

Confusion matrix adalah matriks yang mempresentasikan hasil klasifikasi dari sebuah model dan dibuat dalam bentuk tabel. Tabel tersebut terdiri dari berbagai jumlah baris dan kolom sesuai dengan jumlah kelas yang menunjukkan nilai *False Positives* yaitu nilai akan data negatif yang terdeteksi sebagai data positif, *False Negaitves* yaitu data positif yang terdeteksi sebagai data negatif, *True Positive* yaitu data positif yang terdeteksi benar, dan *True Negative* yaitu data negatif yang terdeteksi benar (Paraijun et al., 2022).

Pada penelitian ini, metrik yang digunakan dalam pengukuran hasil ini adalah *Precision, Recall, F1-score,* dan akurasi (*accuracy*). Berikut adalah persamaan dari akurasi:

$$\boldsymbol{accuracy} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + TN + FP + FN}} \quad (1)$$

Selanjutnya adalah nilai *Precision*, yang mana adalah perbandingan dari nilai *True Positives* (TP) dan banyaknya data yang diprediksi positif, yang mana persamaannya adalah sebagai berikut:

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} \quad (2)$$

Sedangkan *Recall* adalah hasil perbandingan dari nilai *True Positives* (TP) dan banyaknya data yang sebenarnya adalah positif. Persamaan *Recall* dapat dilihat di bawah ini:

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} \quad (3)$$

Dengan kedua nilai *Precision* dan *Recall*, akan dipatkan *F1-score* yang mana persamaannya dapat dilihat sebagai berikut:

$$\boldsymbol{F1\ score} = \mathbf{2\ x}\ \frac{\boldsymbol{Precision} \times \boldsymbol{Recall}}{\boldsymbol{Precision + Recall}} \quad (4)$$

2.11 *Web Service*

Web service adalah layanan yang dapat menghubungkan berbagai aplikasi atau computer untuk dapat berkomunikasi melalui jaringan internet. *Web service* didasarkan pada konsep arsitektur berorientasi layanan atau *Service Oriented Architecture* (SOA) dan *Resource Oriented Architecture* (ROA). SOA merupakan pengembangan komputasi terintegrasi yang memungkinkan komponen berbagai perangkat lunak, mencakupi aplikasi, objek, dan proses dari sistem yang berbeda untuk dapat digunakan sebagai layanan (*service*) sedangkan ROA memungkinkan berbagai informasi pada sistem berbeda menjadi sumber yang dapat digunakan (Wagh & Thool, 2012). Dengan *web service*, aplikasi atau komputer tersebut dapat mengirim permintaan dan menerima respon melalui berbagai protokol standar seperti SOAP, WSDL, dan UDDI (Curbera et al., 2002) serta protokol lainnya seperti Restful API.

REST API adalah salah satu jenis dari *web service* yang menggunakan protokol HTTP untuk berkomunikasi antara aplikasi yang berbeda. Dalam arsitektur RESTful, server menyediakan sumber daya dalam bentuk URI

(Uniform Resource Identifier) dan client dapat mengakses sumber daya tersebut melalui permintaan HTTP seperti *GET*, *POST*, *PUT*, dan *DELETE*.

2.12 Penelitian Terdahulu

Tabel 4. Daftar Penelitian Terdahulu

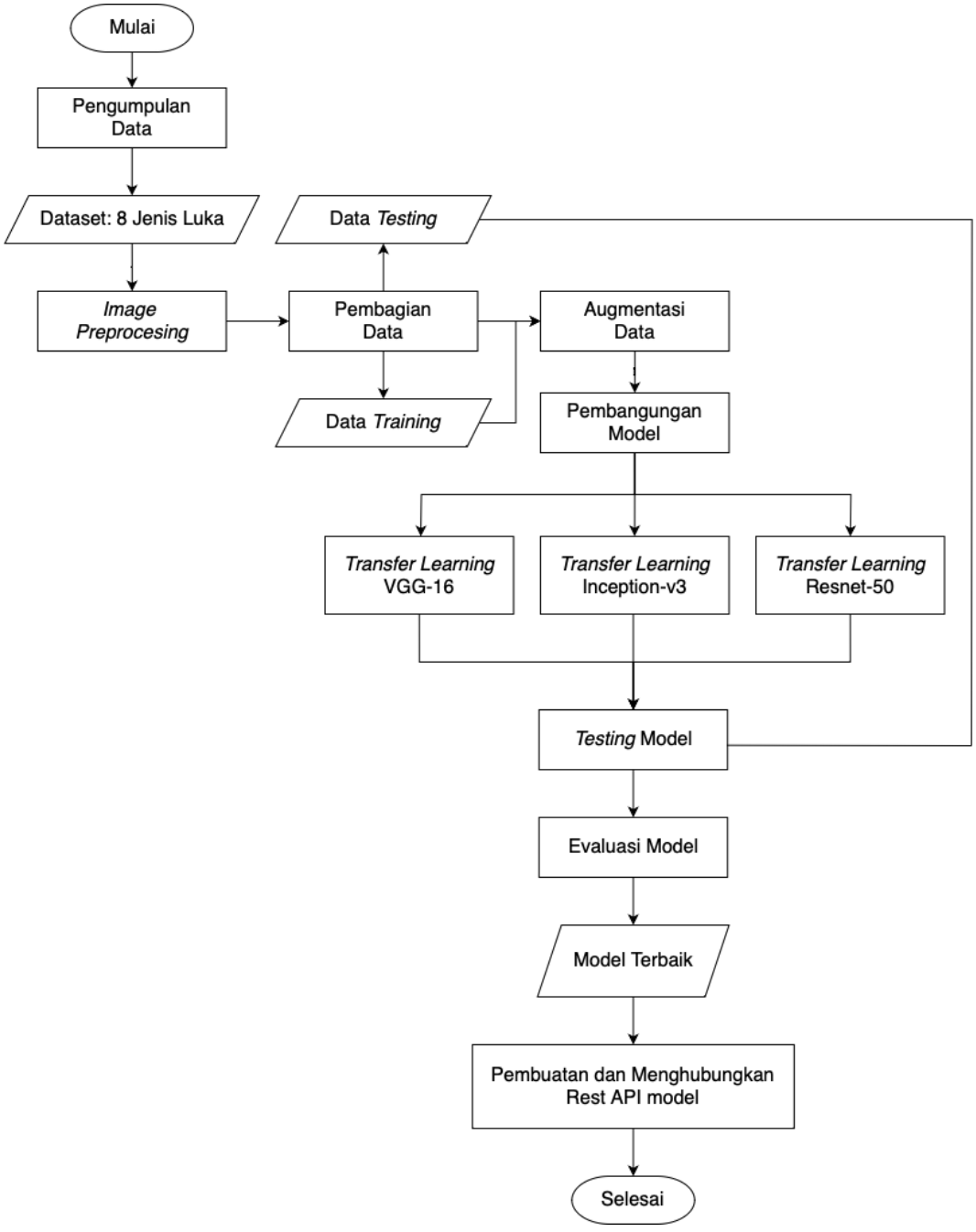
No	Penulis, Tahun	Judul Penelitian	Metode	Hasil
1	(Alzubaidi et al., 2020)	DFU_QUTNet: *Diabetic Foot Ulcer Classification Using Novel Deep Convolutional Neural Network*	*Convolutional Neural Network* (CNN) yaitu membandingkan berbagai *pre-trained model* seperti AlexNet, VGG-16, GoogleNet, DFUNet, dan model sendiri yaitu DFU_QUTNet dengan ekstraksi fitur SVM	Berdasarkan penelitian yang dilakukan, hasil terbaik didapat oleh DFU_QUTNet dengan nilai *F1-Score* 94,5%
2	(Despo, 2015)	*BURNED: Towards Efficient and Accurate Burn Prognosis Using Deep Learning*	*Convolutional Neural Network* (CNN) dengan memodifikasi *pre-trained model* lainnya yaitu AlexNet, VGG, GoogleNet, FCN-8, RNN, CRF	Dengan menggabungkan berbagai metode, percobaan untuk membedakan luka jenis *burn* dan *no burn* dengan nilai terbaik akurasi pixel 85% dan IoU 67%, yaitu dan *multi-burn*), percobaan *multi-burn* dengan nilai terbaik akurasi pixel 60% da IoU 37%.
3	(Shenoy et al., 2019)	*Deepwound: Automated Postoperative Wound Assesment and Surgical Site Surveillance through*	*Convolutional Neural Network* (CNN) dan metode *transfer learning* menggunakan *pre-trained model* V66-16	Hasil yang didapat adalah rata-rata keseluruhan nilai akurasi 85,11%, sensitifitas 86,88%, spesifikasi 69%, dan *F1-score*

No	Penulis, Tahun	Judul Penelitian	Metode	Hasil
		Convolutional Neural Network (CNN)	dan menghasilkan model yang disebut dengan WoundNet	76,11%.
4	(Liu et al., 2020)	*Classification of Ulcer Images Using Convolutional Neural Networks* (CNN)	*Convolutional Neural Network* (CNN) dan metode *transfer learning* menggunakan *pre-trained model* V66-19	Hasil yang didapat adalah nilai akurasi, presisi, dan *recall* dengan masing-masing bernilai sekitar 82%, 85%, dan 75%.
5	(Goyal et al., 2018)	DFUNet: *Convolutional Neural Networks for Diabetic Foot Ulcer Classification*	DFUNet	Akurasi yang didapat setelah pengujian adalah 92,5%.
6	(C. Wang et al., 2015)	*A Unified Framework for Automatic Wound Segmentation and Analysis with Deep Convolutional Neural Networks*	*Convolutional Neural Network* (CNN)	Akurasi tertinggi yang didapat dari percobaan ini mencapai 95,6%.

III. METODOLOGI PENELITIAN

3.1. Tahapan Penelitian

Di bawah ini adalah diagram alir dari tahapan-tahapan yang akan dilaksanakan pada penelitian ini:



Gambar 18. Diagram Alir Penelitian

Pengumpulan Data

Pada penelitian ini, dataset gambar luka dikumpulkan dengan metode *web crawling* atau mengunduh dari internet secara otomatis menggunakan ektensi Google Chrome Bernama imagedownloader serta mengambil data dari

kaggle. Dataset gambar tersebut dari 8 label luka yaitu luka lecet, luka sayat, luka robek, luka tusuk, memar, dan luka bakar dengan 3 jenis*: superficial-thickness, partial-thickness,* dan *full-thicknes*. Dataset yang dikumpulkan ini telah divalidasi oleh Fakultas Kedokteran Universitas Padjajaran pada tahun 2022. Berikut adalah sampel dari dataset 8 jenis luka tersebut:

Gambar 19. *Dataset* yang Dikumpulkan

Image Preprocessing

Pada tahap ini, *dataset* yang ditemukan, dilakukan pengolahan agar data yang digunakan dalam pengembangan model *machine learning* menjadi lebih teratur dan dapat meningkatkan kualitas hasil dari model yang akan dibangun.

Data Duplication. Dengan metode *crawling*, proses pengumpulan data ini menghasilkan data yang belum tentu terdiri dari data yang bersih. Pada tahap ini, dilakukan pengecekan duplikasi data menggunakan teknik *hashing* pada *dataset* yang telah dikumpulkan. Pada setiap label, setiap gambar yang termasuk dalam label tersebut akan diberikan representasi numerik yang unik. Setelah itu, dilakukan pengecekan duplikat data dengan membandingkan representasi unik satu gambar dengan yang lainnya. Apabila ditemukan gambar yang terduplikat, maka hanya satu gambar yang dipertahankan dalam *dataset* untuk nantinya dilakukan proses lanjutan.

Undersampling/Oversampling Data. Metode *crawling* melalui internet ini biasanya menghasilkan data yang belum merata. Apabila pembersihan data

duplikat pada *dataset* menghasilkan jumlah gambar per label yang tidak merata, maka dilakukan pengurangan atau penambahan *dataset*, tergantung jumlah gambar pada setiap label yang ditemukan. Jika jumlah data per label yang ditemukan tergolong sedikit, maka diharapkan dilakukan penambahan data pada setiap labelnya, begitupun sebaliknya. Dengan metode SMOTE, setelah dilakukan pengurangan atau penambahan data sintetis, setiap label data akan menghasilkan jumlah gambar yang sama.

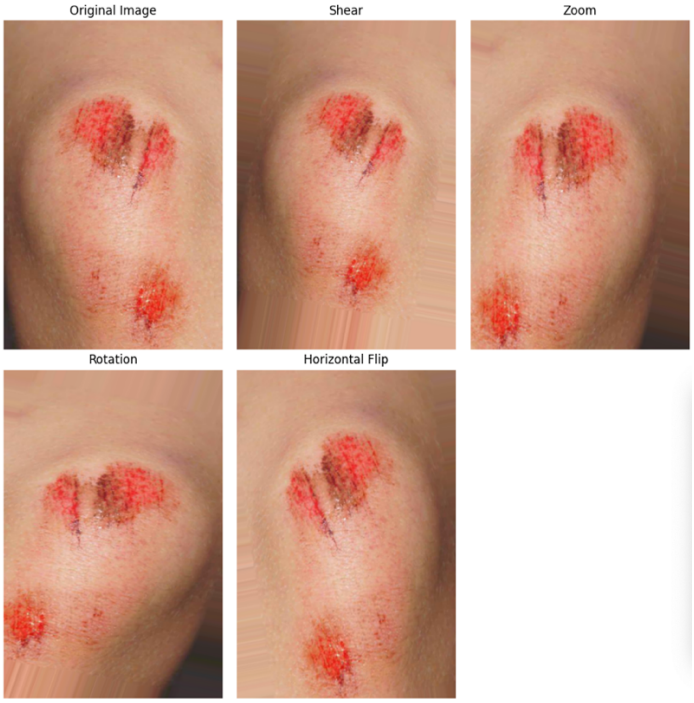
Pembagian dan Augmentasi Data

Sebelum dilakukan pembangunan model *machine learning*, *dataset* yang sebelumnya telah diproses terlebih dahulu, akan dibagi menjadi *data training* dan *data validation*. Setelah dilakukan pembagian data, akan dilakukan augmentasi pada *data training* yang bertujuan untuk memodifikasi dan memanipulasi data sehingga menghasilkan data yang lebih banyak. Data yang dimanipulasi melalui proses augmentasi ini dapat menghasilkan beberapa data dengan berbagai bentuk dan arah yang berbeda. Teknik augmentasi ini terdiri dari *shear, zoom, rotation, fill, flip,* dan lainnya. Di bawah ini adalah deskripsi setiap parameter yang digunakan:

Tabel 5. Deskripsi Parameter pada Proses Augmentasi yang Digunakan

No	Teknik Augmentasi	Deskripsi
1	*Rescale*	Digunakan untuk mengubah nilai intensitas piksel dalam gambar. Dalam hal ini, setiap nilai piksel akan dibagi oleh 255. Hal ini biasanya dilakukan untuk mengubah nilai-nilai piksel menjadi kisaran antara 0 dan 1, yang merupakan skala yang lebih sesuai untuk model jaringan saraf tiruan.
2	*Shear Range*	Digunakan untuk mengontrol sejauh mana gambar dapat digeser (diburamkan) secara horizontal. Nilai 0,4 menunjukkan bahwa gambar dapat digeser sebanyak 0.4 kali lebar gambar.
3	*Zoom Range*	Digunakan untuk mengontrol sejauh mana gambar dapat diperbesar atau diperkecil. Nilai 0.4 menunjukkan bahwa diperbesar atau diperkecil hingga 40% dari ukuran aslinya.
4	*Rotation*	Digunakan untuk melakukan perputaran pada gambar. Nilai 40 berarti gambar dapat diputar hingga 40 derajat dalam arah positif dan negatif.
5	*Horizontal Flip*	Membalikkan citra secara acak dan horizontal.
6	*Fill Mode*	Digunakan untuk mengontrol cara mengisi piksel yang kosong setelah dilakukan transformasi seperti *shear* atau pencerminan. 'nearest' mengartikan bahwa piksel kosong akan diisi dengan piksel terdekat yang ada di gambar.

Di bawah ini adalah contoh gambar asli dan hasil augmentasinya:



Gambar 20. Hasil Augmentasi Gambar

Setelah melakukan proses augmentasi, dilakukanlah proses evaluasi yang membandingkan gambar pada data hasil augmentasi dengan gambar asli/original sebelum dilakukan augmentasi. Proses evaluasi ini menggunakan metode *Peak Signal to Noise Ratio* (PSNR). PSNR adalah metrik yang mengukur tingkat distorsi atau kerusakan yang terjadi dalam proses kompresi atau pengolahan gambar. Metrik ini mengukur perbandingan antara "peak signal" (amplitudo maksimum sinyal asli) dengan "noise" (kesalahan atau distorsi) yang dihasilkan oleh proses. PSNR dinyatakan dalam desibel (dB) dan semakin tinggi nilai PSNR, semakin baik kualitas gambar. PSNR adalah metrik yang umum digunakan untuk mengukur kualitas citra dalam konteks kompresi gambar atau video. Nilai PSNR yang lebih tinggi pasti akan menghasilkan akurasi yang lebih baik (Salomon, 2011).

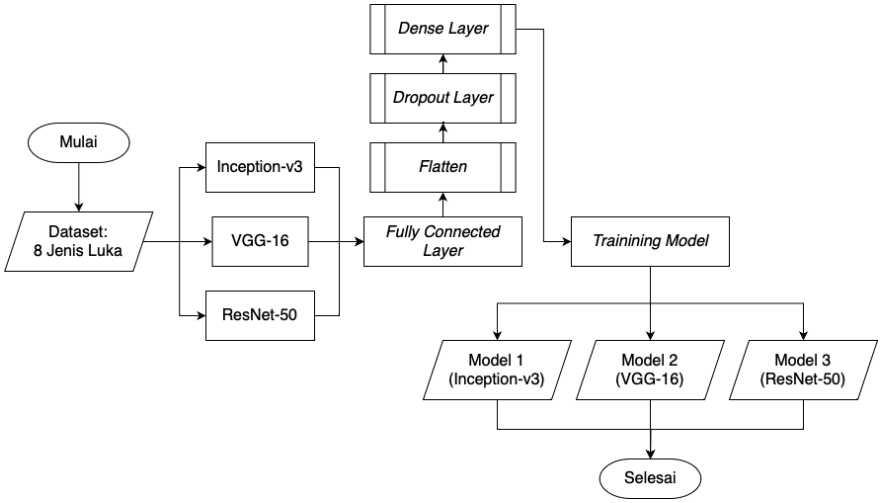
Proses yang dilakukan adalah menghitung rata-rata keseluruhan nilai PSNR yang mana selanjutnya rata-rata tersebut dijadikan sebagai ambang batas. Setelahnya, proses perhitungan PSNR gambar dari data yang telah diaugmentasi ini dilakukan kembali dengan penambahan proses penyalinan gambar. Penyalinan gambar ini dimana gambar dengan nilai PSNR yang lebih

besar dari ambang batas yang telah ditentukan, akan disalin ke direkotori *final* (akhir) sebagai data *final* untuk dilakukan proses pelatihan.

3.2. Pembangunan Model

Setelah dilakukan pemrosesan data, dibangunlah model *machine learning* untuk identifikasi 8 label luka dengan *dataset* yang didapat. Model yang dibangun akan melakukan pelatihan pada *dataset* yang tersedia, dimana *data training* yang telah dibagi sebelumnya lah yang akan digunakan. Proses pembangunan model yang akan dilakukan pada penelitian ini adalah menggunakan metode *transfer learning* dengan mengimpor *pre-trained model* - yang telah dilatih sebelumnya. Pada penelitian ini, terdapat tiga *pre-trained model* yang digunakan, yaitu *Inception-v3, VGG16,* dan *ResNet50*. Nantinya, hasil ketiga model yang menggunakan *pre-trained model* tersebut akan dibandingkan berdasarkan hasil akurasi dan *confusion matrix.*

Untuk setiap *pre-trained model*, akan dilakukan penambahan beberapa *layer* dan parameter pelatihan yang sama persis sehingga proses pembandingan akan sama. Dalam penelitian ini, digunakan algoritma optimasi yaitu *Adam* (*Adaptive Moment Estimation*) *Optimizer*. Algoritma optimasi Adam ini merupakan salah satu algoritma optimasi yang banyak digunakan dalam *deep learning*. Beberapa penelitian mengenai *deep learning* telah menunjukkan bahwa dengan menggunakan *Adam* menghasilkan akurasi terbaik dibandingkan dengan *optimizer* lainnya (Bera & Shrivastava, 2020; Irfan et al., 2022; Wikarta et al., 2020). *Adam optimizer* pertama kali diperkenalkan pada tahun 2015. Pada penelitian tersebut, dapat disimpulkan beberapa keunggulan dari algoritma ini, yaitu efisien secara komputasi, persyaratan memori kecil, dan beberapa kelebihan lainnya (Kingma & Ba, 2015).



Gambar 21. Diagram Alir Pembangunan Model

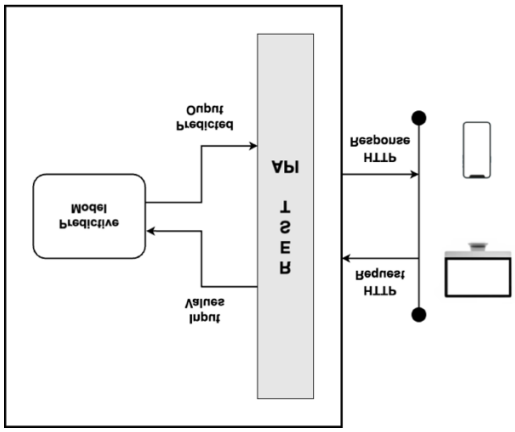
Penelitian ini menggunakan salah satu jenis *multi-class classification loss function* yaitu *categorical cross entropy*. *Categorical cross entropy* merupakan *loss function cross entropy* yang digunakan untuk permasalahan klasifikasi multi- kelas. Dalam prosesnya, *cross entropy* akan meminimalkan nilai *loss* (nilai *cross entropy* yang sempurna adalah 0) dengan menghitung skor perbedaan antara distribusi probabilitas aktual dan prediksi untuk semua kelas (Géron, 2017). Berdasarkan hal tersebut, maka digunakan *categorical cross entropy* sebagai *loss function* karena pada penelitian ini klasifikasi yang digunakan lebih dari dua kelas. Diagram alur dari pembangunan model menggunakan ketiga *pre-trained model* yang telah disebutkan, dapat dilihat pada Gambar 21.

3.3. Evaluasi Model

Pada tahap ini, ketiga model yang telah dibangun akan dilakukan proses evaluasi guna mengambil informasi mengenai *pre-trained model* dengan akurasi terbaik. Proses ini akan dilakukan dengan membandingkan ketiga model yang ada berdasarkan akurasi dari masing-masing model.

Tiga jenis model dari ketiga *pre-trained model* yang diimpor akan dilakukan proses pengujian performa di tahap ini. Pengujian ini dilakukan dengan menggunakan *data validation* dari dataset yang telah didapatkan. Pengujian ini bertujuan untuk mendapatkan informasi performa seperti akurasi dan *confusion matrix*, di antaranya *precision, recall*, dan *f1-score*. Hasil yang didapat dari setiap model tersebut akan menjadi matriks pembanding untuk mendapatkan model dengan *pre-trained model* terbaik.

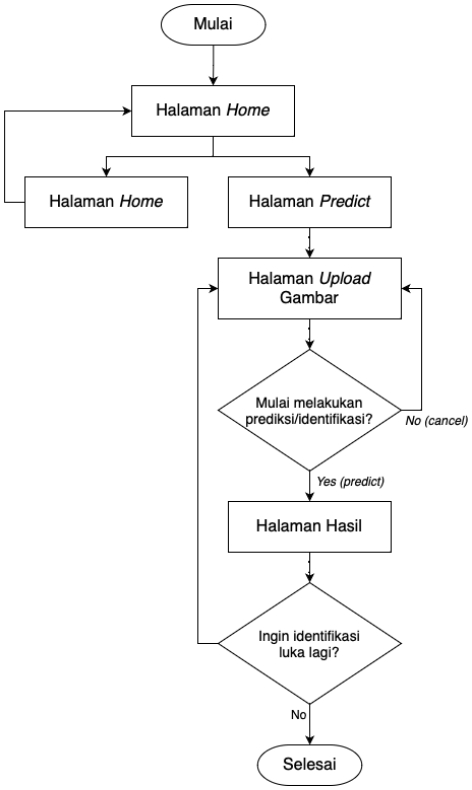
3.4. Menghubungkan Model ke API



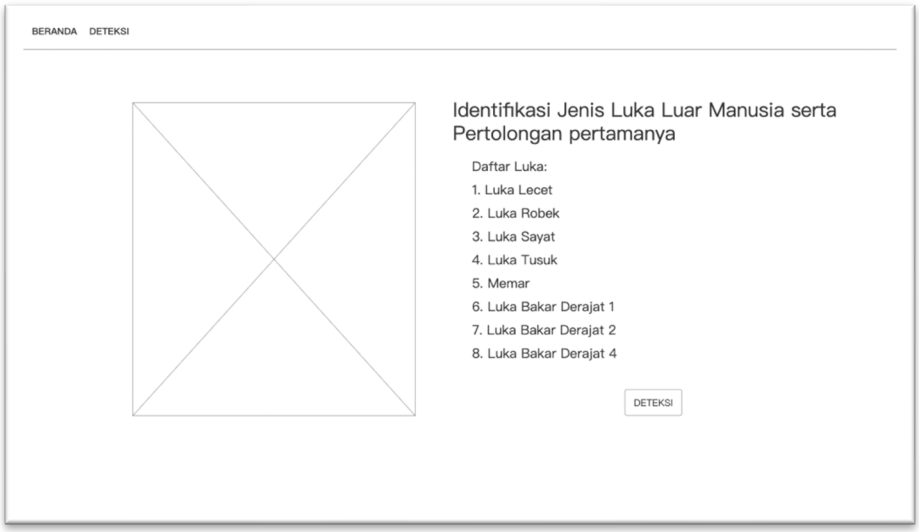
Gambar 22. *Architecture of Implementing ML Model as Service* (Muthu, 2020)

Pada tahap ini, model terbaik, sebagai hasil dari pembandingan ketiga model yang dibangun, akan dihubungkan dengan *web service* dan juga API untuk mempermudah pengguna dalam menjalankan proses deteksi luka luar. Ilustrasi arsitektur dari hubungan *web service* dan API yang akan dibangun dapat dilihat pada Gambar 22.

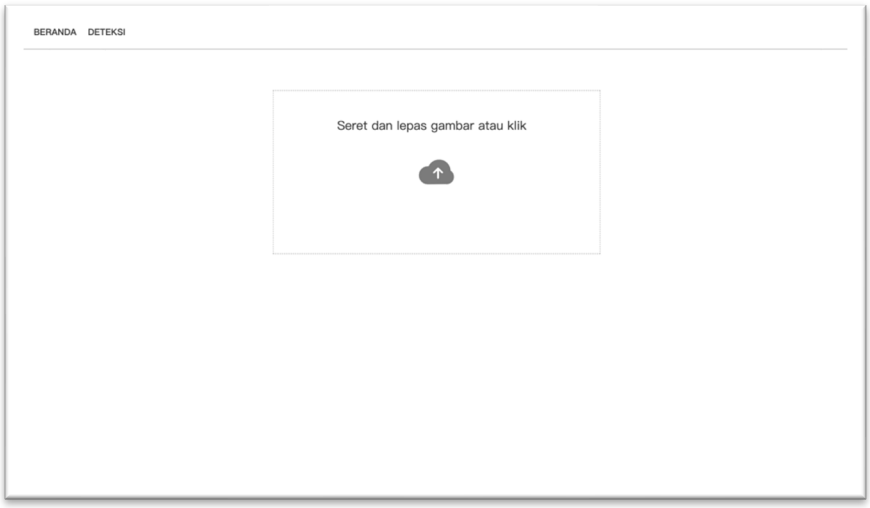
Tampilan *web* yang akan dibangun, direncanakan untuk dapat digunakan oleh pengguna yang mencoba mengidentifikasi 8 jenis luka luar. Tahapan penggunaan web dapat dilihat pada Gambar 23. Selanjutnya adalah *wireframe* atau rancangan halaman pada web yang akan dibangun dapat dilihat pada Gambar 24 hingga Gambar 27.



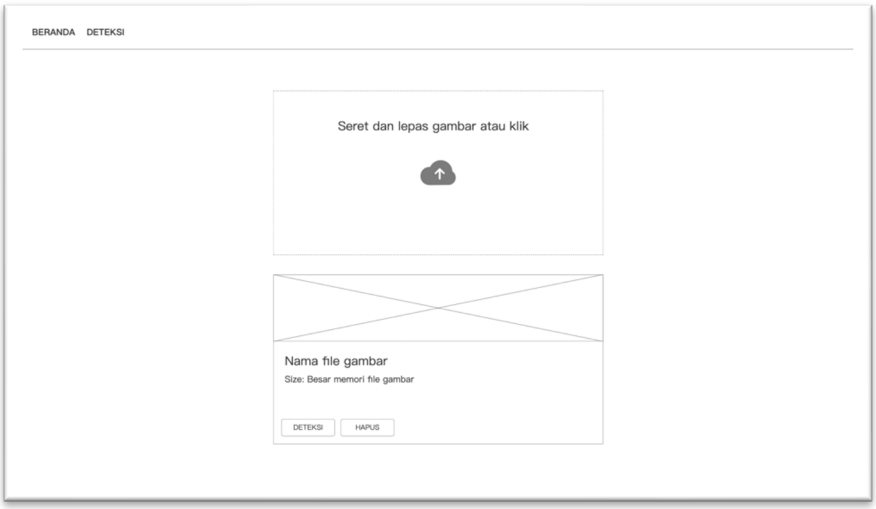
Gambar 23. Digram Alir *Web* Identifikasi Luka



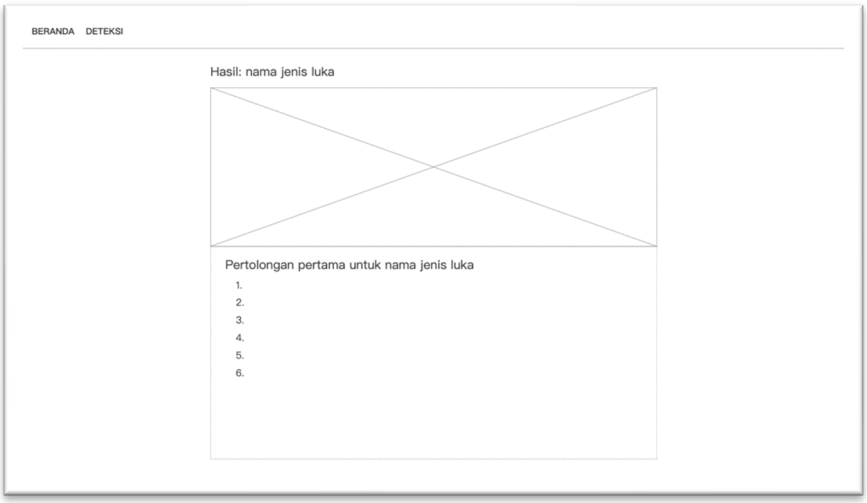
Gambar 24. Halaman *Home*



Gambar 25. Halaman *Upload* Gambar



Gambar 26. Halaman Mulai *Predict*



Gambar 27. Halaman Hasil

3.5. Bahan dan Alat Penelitian

Bahan yang digunakan pada penelitian ini adalah *dataset* dengan 8 jenis luka luar manusia yang didapat dengan metode *crawling web. Crawling dataset* tersebut menggunakan ekstensi aplikasi peranti *Google Chrome*. *Dataset* yang didapat telah divalidasi oleh Fakultas Kedokteran Universitas Padjajaran pada tahun 2022.

Alat yang dibutuhkan pada penelitian ini adalah sebagai berikut:

1. Perangkat Keras (*Hardware*)

Tabel 6. Spesifikasi Perangkat Keras

Spesifikasi	Keterangan
Processor	Chip Apple M1: CPU 8-core, GPU 7-core, Neural Engine 16-core
RAM	8 GB
Storage	256 GB
Display	-
Operating System	macOS Ventura Version 13.1

2. Perangkat Lunak (*Software*)

Tabel 7. Spesifikasi Perangkat Lunak

Spesifikasi	Keterangan
Python Version	Python 3.9.16
RAM	13 GB
Storage	108 GB

3.6. Waktu Penelitian

Berikut adalah jadwal penelitian yang akan dilakukan:

Tabel 8. Jadwal Penelitian

Kegiatan	Bulan								
	Mar	April	Mei	Juni	Juli	Agu	Sept	Okt	Nov
Literature review									
Penulisan skripsi									
Pengumpulan data									
Pre-processing data									

Perancangan model									
Pengujian model									
Pengujian klasifikasi gambar									
Implementasi model ke GUI									

IV. HASIL DAN PEMBAHASAN

4.1. Pengumpulan *Dataset*

Pada tahap awal penelitian ini, *dataset* gambar luka dikumpulkan dengan metode *web crawling* atau mengunduh dari internet secara otomatis menggunakan ektensi *Google Chrome* bernama *imagedownloader*. *Dataset* gambar tersebut dari 8 label luka yaitu luka lecet, luka sayat, luka robek, luka tusuk, memar, dan luka bakar dengan 3 jenis: *superficial-thickness, partial-thickness, dan full-thicknes*. *Dataset* yang dikumpulkan ini telah divalidasi oleh Fakultas Kedokteran Universitas Padjajaran pada tahun 2022.

Proses *web crawling* ini dilakukan secara otomatis dengan mencari label luka tertentu pada kategori *images*/gambar di *Google*. Dengan ekstensi *imagedownloader* ini, dapat dilakukan proses pemilahan pada gambar yang ingin disimpan sehingga penulis pun dapat langsung menyimpan gambar tertentu (sesuai kriteria) pada *folder* yang ditentukan. Hasil dari pengumpulan *dataset* pun dapat dilihat di bawah ini:

Tabel 9. *Keyword* yang Digunakan saat *Web Crawling*

Label	Jumlah Gambar
Luka lecet	133
Luka sayat	68
Luka robek	82
Luka tusuk	23
Luka memar	170
Superficial dermal burn	17
Partial thickness burn	23
Full thickness burn	23
Total	**539**

Berikut adalah beberapa gambar yang didapat:

Gambar 28. Data Gambar Hasil *Crawling Web*

4.2. *Image Preprocessing*

Pada proses ini, *dataset* yang telah ditemukan akan dilakukan pembersihan data sebelum diolah lebih lanjut. Sebelumnya, data yang telah dikumpulkan diunggah di *google drive* milik penulis sehingga ketika ingin diolah dengan *google colab, google colab* harus dihubungkan dengan *google drive* asal data (milik penulis). Berikut merupakan kode program untuk menyambungkan *google drive* dan *colab* serta mengekstraksi data yang telah diambil dalam tipe zip:

```
from google.colab import drive
drive.mount('/content/drive/')

!unzip '/content/drive/MyDrive/Dataset/Dataset.zip'
```

Gambar 29. Kode Program Pengambilan dan Ekstrak Data

Data Duplication

Setelah *dataset* telah terambil (terunduh), *dataset* tersebut akan masuk ke dalam *folder* bernama 'Dataset' yang berisikan total 539 data citra. Sebelum diolah lebih lanjut, *dataset* tersebut akan dilakukan pengecekan data yang duplikat. Selanjutnya, data yang terdeteksi memiliki duplikat akan ditampilkan. Data duplikat tersebut pun akan dipindahkan dari direktori awal ke direktori baru yang mana pada penelitian ini adalah 'DuplicateFiles'. Pada proses ini, terdapat dua fungsi yang akan diperlukan yaitu fungsi *find_duplicates* untuk melakukan pengecekan duplikasi data pada *dataset* dan fungsi *move_dups_to_new_dir* untuk memindahkan data duplikat yang terdeteksi dari direktori awal ke direktori baru.

```python
def find_duplicates(col_name):
    os.chdir(col_name)
    os.getcwd()
    files_list = os.listdir('.')
    print("Number of files in the current directory:", len(files_list))

    duplicates = []
    hash_keys = dict()
    for index, filename in enumerate(os.listdir('.')):
        if os.path.isfile(filename):
            with open(filename, 'rb') as f:
                filehash = hashlib.md5(f.read()).hexdigest()
            if filehash not in hash_keys:
                hash_keys[filehash] = index
            else:
                duplicates.append((index, hash_keys[filehash]))
    if not duplicates:
        print('Result of Duplicates: 0')
    else:
        duplicates.sort(key=lambda x: x[1])
        print('Result of Duplicates:', duplicates)
        return duplicates, files_list

# Define a function to move files to a new directory
def move_dups_to_new_dir(source_dir, destination_dir, dup_indexes):
    for index in dup_indexes:
        source_file = files_list[index]
        destination_file = os.path.join(destination_dir, os.path.basename(source_file))
        shutil.move(source_file, destination_file)
```

Gambar 30. Kode Program Fungsi Temukan dan Pindahkan Data Duplikat

Fungsi *find_duplicates* memiliki parameter *col_name* sebagai direktori kerja dalam pemeriksaan data duplikat. Pengecekan duplikat data ini menggunakan perbandingan nilai *hash* pada setiap *file* dalam *dataset*. Selanjutnya fungsi *move_dups_to_new_dir* yang menggunakan parameter *source_dir* sebagai direktori awal (asal), *destination_dir* sebagai direktori tujuan dipindahkannya data duplikat, dan *dup_indexes* sebagai indeks dari data duplikat yang ditemukan. Berikut adalah kode program untuk memanggil kedua fungsi tersebut:

```
path = '/content/Dataset'
duplicate_dir = '/content/DuplicateFiles'

# Loop through all directories in the path
for col_name in os.listdir(path):
    if os.path.isdir(os.path.join(path, col_name)):
        col_path = os.path.join(path, col_name)
        print(f"Checking duplicates in {col_path}")

        result = find_duplicates(col_path)
        if result is not None:
            duplicates, files_list = result
            os.makedirs(duplicate_dir, exist_ok=True) # Create the duplicate directory if it doesn't exist
            # Showing result of duplicates
            for file_indexes in duplicates[:30]:
                try:
                    file_index_0, file_index_1 = file_indexes
                    if file_index_0 < len(files_list) and file_index_1 < len(files_list):
                        plt.subplot(121), plt.imshow(imread(files_list[file_index_1]))
                        plt.title('index {}'.format(file_index_1)), plt.xticks([]), plt.yticks([])
                        plt.subplot(122), plt.imshow(imread(files_list[file_index_0]))
                        plt.title('index {} is the duplicate'.format(file_index_0)), plt.xticks([]), plt.yticks([])
                        plt.show()
                except OSError as e:
                    continue
            # Move duplicate files to the new directory
            move_dups_to_new_dir(col_path, duplicate_dir, [index[0] for index in duplicates])
```

Gambar 31. Kode Program Temukan dan Pindahkan Data Duplikat

Berikut adalah contoh hasil dari kode program di atas:

Gambar 32. Hasil Pengecekan dan Pemindahan Data Duplikat

Setelah dilakukan pengecekan dan penghapusan data yang terduplikat, total *dataset* berkurang 21 gambar yang terdeteksi sebagai gambar duplikat. Berikut adalah hasil *dataset* yang baru:

Tabel 10. Jumlah Dataset tanpa Duplikasi Data

Label	Jumlah Gambar
Luka lecet	133
Luka sayat	65
Luka robek	81

Luka tusuk	23
Luka memar	154
Superficial dermal burn	17
Partial thickness burn	23
Full thickness burn	22
Total	**518**

Oversampling Data

Pada tahap ini, seluruh data pada setiap label akan disamaratakan sehingga seluruh label memiliki jumlah data yang seimbang. Pada penelitian ini, metode *oversampling* atau menambahkan data akan digunakan. Kode program dari proses *oversampling* data tersebut dapat dilihat pada Gambar 34.

```
path = '/content/Dataset'
Categories_1= os.listdir(path)
flat_data_arr_1=[]
target_arr_1=[]
for i in Categories_1:
  print(f'loading... category : {i}')
  path1 = os.path.join(path,i)
  for img in os.listdir(path1):
    if img != '.DS_Store':
      img_array=imread(os.path.join(path1,img))
      img_resized=resize(img_array,(150,150,3))
      flat_data_arr_1.append(img_resized.flatten())
      target_arr_1.append(Categories_1.index(i))
  print(f'loaded category:{i} successfully')
flat_data_1=np.array(flat_data_arr_1)
target=np.array(target_arr_1)
df_1=pd.DataFrame(flat_data_1)
df_1['Target']=target
x_1=df_1.iloc[:,:-1]
y_1=df_1.iloc[:,-1]
y_1 = LabelEncoder().fit_transform(y_1)
#oversampling
oversample_1 = SMOTE()
x_1, y_1 = oversample_1.fit_resample(x_1, y_1)
```

Gambar 33. Kode Program *Oversampling* Data

Proses *oversampling* tersebut digunakan untuk mendistribusikan penyebaran data sehingga label dengan jumlah data yang sedikit bisa memiliki data yang lebih banyak dengan proses manipulasi data yaitu membuat sintetis data baru hingga seluruh label memiliki jumlah data yang sama banyak dengan label dengan jumlah data paling banyak dimana pada kasus ini adalah luka memar dengan jumlah 154 data. Setelah proses *oversampling*, seluruh data yang masih dalam tipe *dataframe* pun akan dikembalikan menjadi tipe data gambar kembali. Berikut adalah hasil dari distribusi data setelah dilakukan *oversampling*:

Tabel 11. Jumlah *Dataset* setelah *Oversampling*

Label	Jumlah Gambar
Luka lecet	154
Luka sayat	154
Luka robek	154
Luka tusuk	154
Luka memar	154
Superficial dermal burn	154
Partial thickness burn	154
Full thickness burn	154
Total	**1.232**

4.3. Pembagian dan Augementasi Data

Pembagian Data

```
# Real dataset directory
dataset_dir = '/content/new_dataset_after_resampling'
output_dir = '/content/Dataset_split'

# Split dataset into train dan validation
splitfolders.ratio(
    dataset_dir,
    output=output_dir,
    seed=42,
    ratio=(0.8, 0.2),
    group_prefix=None)
train_dir = '/content/Dataset_split/train'
val_dir = '/content/Dataset_split/val'

# Directory where augmented images will be saved to
augmented_train_dir = '/content/Dataset_split/train_augmented'
os.makedirs(augmented_train_dir, exist_ok=True)

# Subdir for classes in augmented_train_dir
class_names = os.listdir(dataset_dir)
for class_name in class_names:
    class_augmented_dir = os.path.join(augmented_train_dir, class_name)
    os.makedirs(class_augmented_dir, exist_ok=True)

# Function for calling the classes
def get_class_name_from_image(image_path):
    # Ambil nama direktori terakhir dari path gambar sebagai nama kelas
    class_name = os.path.basename(os.path.dirname(image_path))
    return class_name
```

Gambar 34. Kode Program Pembuatan dan Pembagian *Folder* pada Proses Augmentasi

Setelah melakukan *image preprocessing* dengan menghapus duplikat data dan proses *oversampling* untuk memperbanyak serta meratakan sebaran data dengan metode SMOTE. *Dataset* tersebut pun disimpan pada *folder new_dataset_after_resampling*. Sebelum ditentukan jumlah pembagian *dataset* pada data *training* dan *validation*, terlebih dahulu dibuat *folder* baru untuk menyimpan dataset yang akan dibagi nantinya serta rasio pembagian data *train*

dan *valid.* Rasio pembagian data *train* dan *valid* pada penelitian ini adalah data *training* sebanyak 80% dan data *valid* sebanyak 20%. Dengan fungsi *splitfolders.ratio,* data yang dibagi otomatis menjadi dua *folder* dimana urutan definisi rasio sesuai dengan data *train* dan data *val.* *Folder* atau direktori penyimpanan data hasil augmentasi pun juga dibuat dengan nama 'train_augmented' karena proses augmentasi hanya dilakukan pada data *training*. Selanjutnya, pada *folder* tersebut, juga dibuat sub-direktori untuk tiap kelas/label pada penelitian ini. Setelah itu, diinisialisasi fungsi 'get_class_name_from_image' yang akan dipanggil saat proses augmentasi. Fungsi ini berfungsi untuk mengambil kelas gambar (data) dari nama terakhir data yang menunjukkan kelas dari gambar tersebut.

Augmentasi Data

Dataset yang telah dibagi, kemudian diproses dengan melakukan augmentasi. Proses augmentasi ini bertujuan untuk memperbanyak data citra dengan membuat gambar baru dari gambar sebelumnya dengan berbagai sudut dan posisi yang berbeda. Pada penelitian ini dalam proses augmentasi digunakan fungsi *ImageDataGenerator* dengan menerapkan beberapa teknik augmentasi, yaitu *rescale*, *shear*, *zoom*, *rotation*, *horizontal flip*, dan *fill mode*.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    rotation_range=20,
    horizontal_flip=True,
    fill_mode='nearest',
)
val_datagen = ImageDataGenerator(
    rescale=1./255,
)
```

Gambar 35. Kode Program Fungsi *ImageDataGenerator*

Di atas adalah kode program pada pembuatan fungsi *ImageDataGenerator* yang akan dipanggil pada proses augmentasi yang dapat dilihat pada Gambar 36.

```
# Augmentaion process and saving to 'train_augmented' process
BATCH_SIZE = 16
TARGET_SIZE = 150
AUGMENTATION_FACTOR = 8

for root, _, files in os.walk(train_dir):
    for file in files:
        image_path = os.path.join(root, file)
        img = tf.keras.preprocessing.image.load_img(
            image_path,
            target_size=(TARGET_SIZE, TARGET_SIZE))
        x = tf.keras.preprocessing.image.img_to_array(img)
        x = x.reshape((1,) + x.shape)
        i = 0
        class_name = get_class_name_from_image(image_path)
        class_augmented_dir = os.path.join(augmented_train_dir, class_name)
        os.makedirs(class_augmented_dir, exist_ok=True)
        for batch in train_datagen.flow(x, batch_size=1):
            augmented_image = tf.keras.preprocessing.image.array_to_img(batch[0])
            augmented_image.save(os.path.join(class_augmented_dir, f'augmented_{i}_{file}'))
            i += 1
            if i >= AUGMENTATION_FACTOR:
                break
```

Gambar 36. Kode Program Proses Augmentasi

Kode di atas dalah kode untuk melakukan proses augmentasi dengan data *training.* Seluruh data yang akan diaugmentasi akan dilakukan proses *resize* menjadi 150x150 sesuai dengan 'TARGET_SIZE' yang telah diinisialisasi. Selanjutnya, setiap gambar pada *dataset* akan dilakukan augmentasi sebanyak 8 kali sesuai dengan inisialisasi 'AUGMENTATION_FACTOR'. Selanjutnya, seluruh data yang telah diaugmentasi akan disimpan ke dalam *folder* atau direktori 'augmented_train_dir'. Setelah dilakukan augmentasi, jumlah data *train* dan data validasi yang dihasilkan pun menjadi sebagai berikut:

Tabel 12. Jumlah Data setelah Augmentasi

Label	**Jumlah Gambar**	
	Data *Training*	**Data Validasi**
Luka lecet	984	31
Luka sayat	984	31
Luka robek	984	31
Luka tusuk	984	31
Luka memar	984	31
Superficial dermal burn	984	31
Partial thickness burn	984	31
Full thickness burn	984	31
Total	**7.872**	**248**

Berikut adalah contoh hasil augmentasi pada tiap kelas yang dilakukan:

Gambar 37. Hasil Augmentasi

Setelah menghasilkan data *training* yang sangat banyak sebagai hasil augmentasi, dilakukan proses evaluasi penghitungan nilai *Peak Signal to Noise Ratio* (PSNR). Proses ini akan menghitung nilai PSNR dari data *training* hasil augmentasi pada direktori 'train_augmented' dengan cara membandingkannya dengan data *training* awal (sebelum augmentasi) sesuai dengan kelasnya. Seluruh data *training* hasil augmentasi akan dibandingkan dengan data awal/asli yang dipilih sesuai indeksnya untuk dapat dihitung nilai PSNR-nya. Data asli diambil dari direktori 'train' dan data hasil augmentasi diambil dari direktori 'train_augmented'. Setelah itu dihitunglah rata-rata dari nilai PSNR keseluruhan.

```
# Variables inisialitation to store total PSNR and count
original_images_dir = "/content/Dataset_split/train"
augmented_images_dir = "/content/Dataset_split/train_augmented"

# Variables inisialitation to store total PSNR and count
total_psnr = 0
count = 0

# Looping each subdir classes
class_names = os.listdir(augmented_images_dir)  # Use train_augmented dir
for class_name in class_names:
    class_original_dir = os.path.join(original_images_dir, class_name)
    class_augmented_dir = os.path.join(augmented_images_dir, class_name)

    original_images = os.listdir(class_original_dir)
    augmented_images = os.listdir(class_augmented_dir)

    # Looping imgs in class subdir
    for augmented_image_name in augmented_images:
        augmented_image_path = os.path.join(class_augmented_dir, augmented_image_name)
        augmented_image = io.imread(augmented_image_path)
        augmented_image = color.rgb2gray(augmented_image)

        for original_image_name in original_images:
            original_image_path = os.path.join(class_original_dir, original_image_name)
            original_image = io.imread(original_image_path)
            original_image = color.rgb2gray(original_image)

            psnr_score = psnr(original_image, augmented_image)

            total_psnr += psnr_score
            count += 1

# Calculate the overall average PSNR for all classes
overall_average_psnr = total_psnr / count
print(f"Overall Average PSNR Score: {overall_average_psnr}")
```

Gambar 38. Kode Perhitungan Rata-Rata PSNR

```
# Copying resoluted img to new dir
# Variables inisialitation to store total PSNR and count
original_images_dir = "/content/Dataset_split/train"
augmented_images_dir = "/content/Dataset_split/train_augmented"

# output dir to save imgs with PSNR > 11.992841544610027
output_dir = "/content/resoluted_dataset"
os.makedirs(output_dir, exist_ok=True)

# Looping each subdir classes
class_names = os.listdir(augmented_images_dir)  # Use train_augmented dir
for class_name in class_names:
    class_original_dir = os.path.join(original_images_dir, class_name)
    class_augmented_dir = os.path.join(augmented_images_dir, class_name)

    # Variables inisialitation for current class
    total_psnr = 0
    count = 0

    original_images = os.listdir(class_original_dir)
    augmented_images = os.listdir(class_augmented_dir)

    # Looping imgs in class subdir
    for augmented_image_name in augmented_images:
        augmented_image_path = os.path.join(class_augmented_dir, augmented_image_name)
        augmented_image = io.imread(augmented_image_path)
        augmented_image = color.rgb2gray(augmented_image)

        for original_image_name in original_images:
            original_image_path = os.path.join(class_original_dir, original_image_name)
            original_image = io.imread(original_image_path)
            original_image = color.rgb2gray(original_image)

            psnr_score = psnr(original_image, augmented_image)

            if psnr_score > 11.992841544610027:
                target_dir = os.path.join(output_dir, class_name)
                os.makedirs(target_dir, exist_ok=True)
                shutil.copy(augmented_image_path, target_dir)
                break  # Move to the next augmented image
```

Gambar 39. Kode Pembersihan Data Hasil Augmentasi dengan nilai di Bawah Ambang Batas (Rata-rata)

Setelah didapatkan nilai rata-ratanya dengan kode pada Gambar 38, nilai tersebut akan dijadikan ambang batas untuk data hasil augmentasi yang akan digunakan untuk proses training. Nilai rata-rata yang didapatkan adalah 11.992841544610027. Selanjutnya, dilakukanlah proses perhitungan nilai PSNR kembali dimana dengan modifikasi kode yang mengatur data hasil augmentasi dengan nilai PSNR > 11.992841544610027 akan disalin ke direktori baru yaitu 'resoluted_dataset' yang dapat dilihat pada Gambar 39. Berikut adalah jumlah data *training* baru dalam direktori 'resoluted_dataset' dan data validasi yang akan digunakan pada proses *training*:

Tabel 13. Jumlah Data *Final*

Label	**Jumlah Gambar**	
	Data *Training*	**Data Validasi**
Luka lecet	968	31
Luka sayat	968	31
Luka robek	948	31

Luka tusuk	944	31
Luka memar	911	31
Superficial dermal burn	973	31
Partial thickness burn	962	31
Full thickness burn	977	31
Total	**7.651**	**248**

Selanjutnya adalah pemanggilan fungsi 'flow_from_directory' untuk mengolah data *training* dan data validas *final* (akhir) yang dapat dilihat pada Gambar 40. Pada Gambar 40 tersebut, terdapat proses inisialisasi fungsi *flow_from_directory* untuk data *train* dan data *validation.* 'Target_generator' akan memanggil data *train final* yaitu pada direktori 'resoluted_dataset sedangkan 'val_generator' akan memanggil data pada direktori 'val_dir' yang mana selanjutkan dilakukan pengaturan data sesuai dengan *target size, batch,* mode kelas, dan pengaturan acak ketika *generator* tersebut digunakan nantinya.

```
train_final = '/content/resoluted_dataset'
# Generator for augmented dataset
train_generator = train_datagen.flow_from_directory(
    train_final,
    target_size=(TARGET_SIZE, TARGET_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
)

val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(TARGET_SIZE, TARGET_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,
)
```

Gambar 40. Inisialisasi Fungsi 'flow_from_directory'

4.4. Pembangunan Model

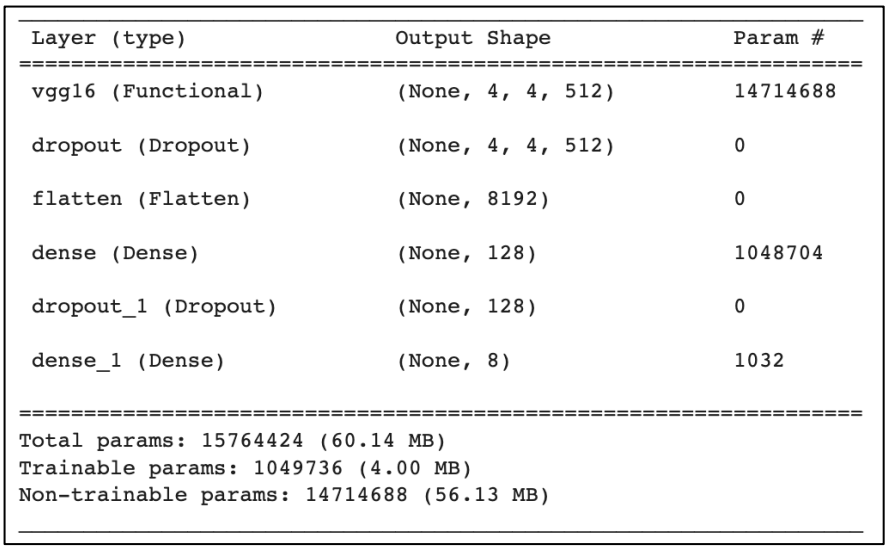
Setelah melalui beberapa tahapan, maka selanjutnya data yang telah diproses akan dilatih. Pada penelitian ini, proses pelatihan data dilakukan dengan tiga model berbeda yang mengambil pengetahuan dari model yang telah ada yaitu VGG-16, Inception-v3, dan ResNet50. Dalam proses pembangunan modelnya, ketiga *pre-trained model* tidak akan dilatih kembali untuk mempertahankan bobot atau *weight* pada tiap *pre-trained model* pada pelatihan sebelumnya. Namun, setelah meng-*input layer pre-trained model,* akan ditambah beberapa *layer* tambahan guna menyesuaikan pengetahuan pada *pre-trained model* dengan tugas baru (identifikasi luka), yang mana pada penelitian ini adalah *dropout, flatten,* dan *fully-connected layer* (*dense*).

Pada penelitian ini, input gambar yang dimasukan adalah 150x150x3 dan lapisan *output* pada tiap *pre-trained model* yang asli tidak dimasukkan. Selanjutnya, terdapat lapisan *dropout,* yaitu teknik regularisasi yang akan menghilangkan *neuron* di dalam jaringan. Pada penelitian ini, digunakan lapisan *dropout* sebesar 0,4 sebagai probabilitas pada setiap *neuron* yang diharapkan dapat mencegah atau mengurangi *overfitting* sekaligus mempercepat pembelajaran.

Setelah lapisan *dropout*, terdapat tahap *flatten* atau merubah keluaran data pada *pre-trained model* yang digunakan, seperti VGG-16 yang sebelumnya berupa tensor empat dimensi menjadi vektor satu dimensi. Selanjutnya adalah proses *dense*, yaitu menambah *fully-connected layer* dengan 128 unit dan fungsi aktivasi ReLU pada *hidden layer* ini. Tahap berikutnya adalah proses *dropout* kembali namun dengan tingkat yang berbeda dengan sebelumnya menjadi 0,25. Selanjutnya, *layer* akhir pun ditambahkan dengan nilai keluaran berjumlah 8 unit dimana sesuai dengan jumlah kelas atau label pada penelitian ini yaitu: 'luka_lecet', 'luka_sayat', 'luka_robek', 'luka_tusuk', 'luka_memar', 'superficial_dermal_burn', 'partial_thickness_burn', dan 'full_thickness_burn'. Dikarenakan terdapat lebih dari dua kelas (multi-kelas), maka fungsi aktivasi yang digunakan pada *output layer* ini adalah *softmax* dimana aktivasi *softmax* akan mengembalikan probabilitas dari masing-masing kelas dan kelas target akan memiliki nilai probabilitas lebih tinggi dibandingkan dengan kelas yang lain.

VGG-16

Berikut adalah *model summary* dari *model* yang menggunakan *pre-trained model* VGG-16:



```
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688

dropout (Dropout)            (None, 4, 4, 512)         0

flatten (Flatten)            (None, 8192)              0

dense (Dense)                (None, 128)               1048704

dropout_1 (Dropout)          (None, 128)               0

dense_1 (Dense)              (None, 8)                 1032

=================================================================
Total params: 15764424 (60.14 MB)
Trainable params: 1049736 (4.00 MB)
Non-trainable params: 14714688 (56.13 MB)
```

Gambar 41. *Model Summary* (VGG-16)

Hasil *summary* pada model yang dibangun ini terdiri dari setiap *layer* pada model, *output shape*, dan jumlah parameter yang digunakan untuk proses pelatihan pada tiap *layer*-nya. Berikut adalah deskripsi dari hasil *summary* model dengan VGG-16:

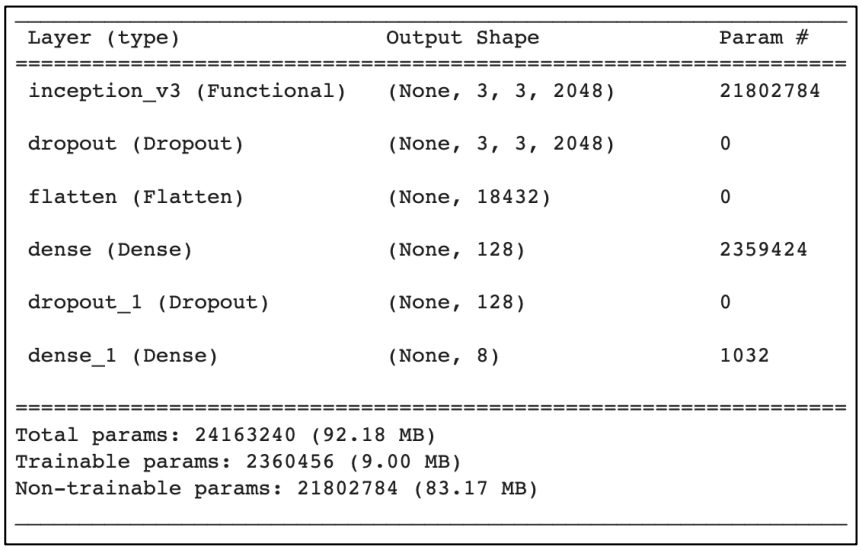
Tabel 14. Deskripsi *Model Summary* (VGG-16)

Layer **(*type*)**	**Deskripsi**	
	Output Shape	**Parameter**
Vgg16	(None, 4, 4, 512) menandakan struktur data yang dihasilkan adalah 4 dimensi tensor. 1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 4, 4, 512: Menunjukkan dimensi spasial dari *feature map* yang dihasilkan yaitu panjang/lebar 4x4 sebanyak 512 fitur/*filter*	Nilai 14.714.688 ini menandakan jumlah bobot (*weights*) dan bias pada data pelatihan sebelumnya yang dapat digunakan (yang mana pada pelatihan *layer vgg16* ini dilatih dengan data *Imagenet*)
Dropout	Karena tidak ada bobot pelatihan yang terlibat pada proses *dropout* maka bentuk keluaran *layer* ini sama dengan sebelumnya yaitu (None, 4, 4, 512)	Nilai 0 menandakan tidak ada bobot/parameter pelatihan yang digunakan (tidak dibutuhkan)
Flatten	1. *None*: Menujukkan tidak	Nilai 0 menandakan tidak

	adanya jumlah *batch default* 2. 8192: Menunjukkan bahwa dimensi spasial hasil dari *layer* sebelumnya diratakan menjadi 8192 dengan perhitungan 4x4x512 = 8192	ada bobot/parameter pelatihan yang digunakan (tidak dibutuhkan)
Dense	1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 128: Menunjukkan jumlah neuron tujuan sesuai dengan parameter *layer* ini yaitu 128	1.048.704 merupakan bobot yang dapat digunakan, didapat dari jumlah neuron sebelumnya + 1 bias lalu dikalikan dengan jumlah neuron tujuan. 8192 + 1 = 8193 8193 x 128 = 1.048.704
Dropout_1	Karena tidak ada bobot pelatihan yang terlibat pada proses *dropout* maka bentuk keluaran *layer* ini sama dengan sebelumnya yaitu (None, 128)	Nilai 0 menandakan tidak ada bobot/parameter pelatihan yang digunakan (tidak dibutuhkan)
Dense_1	1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 128: Menunjukkan jumlah neuron tujuan sesuai dengan parameter *layer* ini yaitu 8, yaitu jumlah kelas prediksi pada penelitian ini	1.032 merupakan bobot yang dapat digunakan, didapat dari jumlah neuron sebelumnya + 1 bias lalu dikalikan dengan jumlah neuron tujuan. 128 + 1 = 129 129 x 8 = 1.032

Inception-V3

Berikut adalah *model summary* dari *model* yang menggunakan *pre-trained model* Inception-v3:



Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 3, 3, 2048)	21802784
dropout (Dropout)	(None, 3, 3, 2048)	0
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 128)	2359424
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 8)	1032

Total params: 24163240 (92.18 MB)
Trainable params: 2360456 (9.00 MB)
Non-trainable params: 21802784 (83.17 MB)

Gambar 42. *Model Summary* (Inception-v3)

Hasil *summary* pada model yang dibangun ini terdiri dari setiap *layer* pada model, *output shape*, dan jumlah parameter yang digunakan untuk proses pelatihan pada tiap *layer*-nya. Berikut adalah deskripsi dari hasil *summary* model dengan Inception-v3:

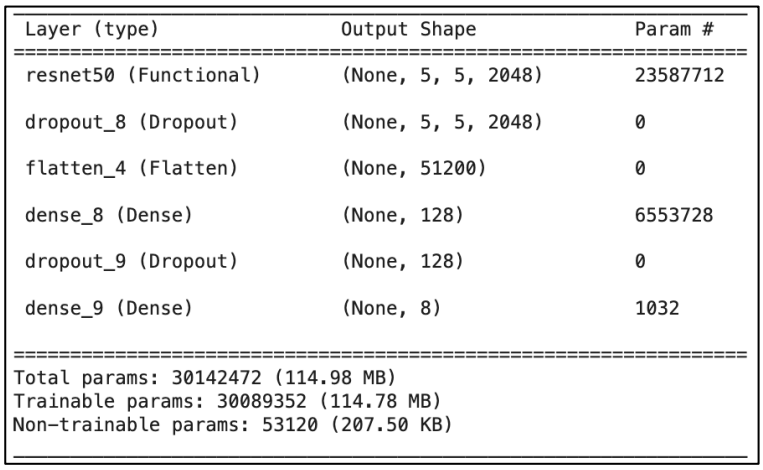
Tabel 15. Deskripsi Model *Summary* (Inception-v3)

Layer **(*type*)**	**Deskripsi**	
	Output Shape	**Parameter**
Inception_v3	(None, 3, 3, 2048) menandakan struktur data yang dihasilkan adalah 4 dimensi tensor. 1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 3, 3, 2048: Menunjukkan dimensi spasial dari *feature map* yang dihasilkan yaitu panjang/lebar 3x3 sebanyak 2048 fitur/*filter*	Nilai 21.802.784 ini menandakan jumlah bobot (*weights*) dan bias pada data pelatihan sebelumnya yang dapat digunakan (yang mana pada pelatihan *layer inception_v3* ini dilatih dengan data *Imagenet*)
Dropout	Karena tidak ada bobot pelatihan yang terlibat pada proses *dropout* maka bentuk keluaran *layer* ini sama dengan sebelumnya yaitu (None, 3, 3, 2048)	Nilai 0 menandakan tidak ada bobot/parameter pelatihan yang digunakan (tidak dibutuhkan)

Flatten	1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 18.432: Menunjukkan bahwa dimensi spasial hasil dari *layer* sebelumnya diratakan menjadi 18.432 dengan perhitungan 3x3x2048 = 18.432	Nilai 0 menandakan tidak ada bobot/parameter pelatihan yang digunakan (tidak dibutuhkan)
Dense	1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 128: Menunjukkan jumlah neuron tujuan sesuai dengan parameter *layer* ini yaitu 128	2.359.424 merupakan bobot yang dapat digunakan, didapat dari jumlah neuron sebelumnya + 1 bias lalu dikalikan dengan jumlah neuron tujuan. 18.432 + 1 = 18.433 18.433 x 128 = 2.359.424
Dropout_1	Karena tidak ada bobot pelatihan yang terlibat pada proses *dropout* maka bentuk keluaran *layer* ini sama dengan sebelumnya yaitu (None, 128)	Nilai 0 menandakan tidak ada bobot/parameter pelatihan yang digunakan (tidak dibutuhkan)
Dense_1	1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 128: Menunjukkan jumlah neuron tujuan sesuai dengan parameter *layer* ini yaitu 8, yaitu jumlah kelas prediksi pada penelitian ini	1.032 merupakan bobot yang dapat digunakan, didapat dari jumlah neuron sebelumnya + 1 bias lalu dikalikan dengan jumlah neuron tujuan. 128 + 1 = 129 129 x 8 = 1.032

ResNet-50

Berikut adalah *model summary* dari *model* yang menggunakan *pre-trained model* ResNet-50:



Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 5, 5, 2048)	23587712
dropout_8 (Dropout)	(None, 5, 5, 2048)	0
flatten_4 (Flatten)	(None, 51200)	0
dense_8 (Dense)	(None, 128)	6553728
dropout_9 (Dropout)	(None, 128)	0
dense_9 (Dense)	(None, 8)	1032

Total params: 30142472 (114.98 MB)
Trainable params: 30089352 (114.78 MB)
Non-trainable params: 53120 (207.50 KB)

Gambar 43. *Model Summary* (ResNet-50)

Hasil *summary* pada model yang dibangun ini terdiri dari setiap *layer* pada model, *output shape,* dan jumlah parameter yang digunakan untuk proses pelatihan pada tiap *layer*-nya. Berikut adalah deskripsi dari hasil *summary* model dengan ResNet-50:

Tabel 16. Deskripsi Model *Summary* (ResNet-50)

Layer **(*type*)**	**Deskripsi**	
	Output Shape	**Parameter**
Resnet50	(None, 5, 5, 2048) menandakan struktur data yang dihasilkan adalah 4 dimensi tensor. 3. *None*: Menujukkan tidak adanya jumlah *batch default* 4. 5, 5, 2048: Menunjukkan dimensi spasial dari *feature map* yang dihasilkan yaitu panjang/lebar 5x5 sebanyak 2048 fitur/*filter*	Nilai 23.587.712 ini menandakan jumlah bobot (*weights*) dan bias pada data pelatihan sebelumnya yang dapat digunakan (yang mana pada pelatihan *layer resnet50* ini dilatih dengan data *Imagenet*)
Dropout	Karena tidak ada bobot pelatihan yang terlibat pada proses *dropout* maka bentuk keluaran *layer* ini sama dengan sebelumnya yaitu (None, 5, 5, 2048)	Nilai 0 menandakan tidak ada bobot/parameter pelatihan yang digunakan (tidak dibutuhkan)
Flatten	1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 51.200: Menunjukkan bahwa	Nilai 0 menandakan tidak ada bobot/parameter pelatihan yang digunakan

	dimensi spasial hasil dari *layer* sebelumnya diratakan menjadi 51.200 dengan perhitungan 5x5x2048 = 51.200	(tidak dibutuhkan)
Dense	1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 128: Menunjukkan jumlah neuron tujuan sesuai dengan parameter *layer* ini yaitu 128	6.553.728 merupakan bobot yang dapat digunakan, didapat dari jumlah neuron sebelumnya + 1 bias lalu dikalikan dengan jumlah neuron tujuan. 51.200 + 1 = 51.201 51.201 x 128 = 6.553.728
Dropout_1	Karena tidak ada bobot pelatihan yang terlibat pada proses *dropout* maka bentuk keluaran *layer* ini sama dengan sebelumnya yaitu (None, 128)	Nilai 0 menandakan tidak ada bobot/parameter pelatihan yang digunakan (tidak dibutuhkan)
Dense_1	1. *None*: Menujukkan tidak adanya jumlah *batch default* 2. 128: Menunjukkan jumlah neuron tujuan sesuai dengan parameter *layer* ini yaitu 8, yaitu jumlah kelas prediksi pada penelitian ini	1.032 merupakan bobot yang dapat digunakan, didapat dari jumlah neuron sebelumnya + 1 bias lalu dikalikan dengan jumlah neuron tujuan. 128 + 1 = 129 129 x 8 = 1.032

Inisialisasi *Parameter Learning*

Tujuan dari perancangan serta melakukan pelatihan algoritma CNN ini adalah agar sistem dapat mengenali ciri dari setiap gambar dan kemudian menandai seluruh *neuron* serta menentukan dari setiap *neuron* tersebut yang mana yang akan diaktifkan ketika gambar diklasifikasi. Sebelum dilakukan proses training, data *training* yang telah melalui proses augmentasi gambar perlu dipanggil terlebih dahulu dan perlu dilakukan inisialisasi terhadap beberapa *parameter learning*. Beberapa *parameter learning* yang perlu diinisialisasi untuk proses training ada tiga, yaitu *learning rate*, *callbacks, batch size,* dan *epoch*.

```
lr_reduce = ReduceLROnPlateau(
              monitor='val_loss',
              factor=0.6,
              patience=6,
              verbose=1,
              mode='min',
              min_lr=5e-5
              )
callbacks = EarlyStopping(
              monitor='val_accuracy',
              mode='max',
              patience=10,
              restore_best_weights=True
              )
train_steps = train_generator.samples
val_steps = validation_generator.samples
epoch = 100
model.compile(optimizer=tf.optimizers.Adam(learning_rate=0.001),
                loss='categorical_crossentropy',
                metrics=['accuracy']
              )
history = model.fit(
            train_generator,
            steps_per_epoch= train_steps // BATCH_SIZE,
            epochs=epoch,
            validation_data=(validation_generator),
            validation_steps= val_steps // BATCH_SIZE,
            callbacks= [lr_reduce, callbacks],
            verbose=1,
                    )
```

Gambar 44. Inisialisasi Parameter Pembelajar dan Pemanggilan Fungsi Pelatihan

Pada penenlitian ini, nilai *learning rate* yang digunakan adalah 0,001 dengan *optimizer* Adam, fungsi *loss* 'categorical cross entropy' sebagai fungsi *loss* multikelas, serta metriks yang digunakan adalah nilai akurasi (*accuracy*). Pada proses pelatihan, digunakan data *training* dan data validasi yang telah diinisialisasi sebelumnya yaitu *train_generator* dan *validation_generator*. Dengan nilai *batch* 16, pada tiap *epoch*, pelatihan akan dibagi menjadi 16 *batch* dimana tiap *batch*-nya akan melatihan jumlah total data dibagi dengan 16 sehingga setiap *batch* akan melatih jumlah data yang setara, dan pelatihan ini akan dilakukan sebanyak 100 kali sesuai dengan jumlah *epoch* yang ditentukan yaitu 100. Guna memonitor proses pelatihan, penulis pun menambah dua fungsi *callbacks* yaitu *ReduceLROnPlateu* dan *EarlyStopping*.

ReduceLROnPlateau

Fungsi ini akan memantai atau memonitor sebuah kuantitas dan akan mengurangi atau memperkecil nilai *learning rate* jika *metric* atau nilai *loss* tidak mengalami perkembangan dalam sejumlah 'patience' dari *epoch*. Pada penelitian ini, nilai *learning rate* dimulai dengan 0,001 dan akan dikurangi dengan cara mengkalikan nilai *learning rate* sebelumnya dengan nilai *factor* yaitu 0,6 apabila

nilai ‘val_loss’ tidak berkurang selama 6 kali pelatihan (6 *epoch*) dan pengurangan nilai *learning rate* tidak akan dilakukan lagi apabila sudah mencapai nilai minimal *learning rate* yaitu 5e-0 atau 0,00005.

EarlyStopping

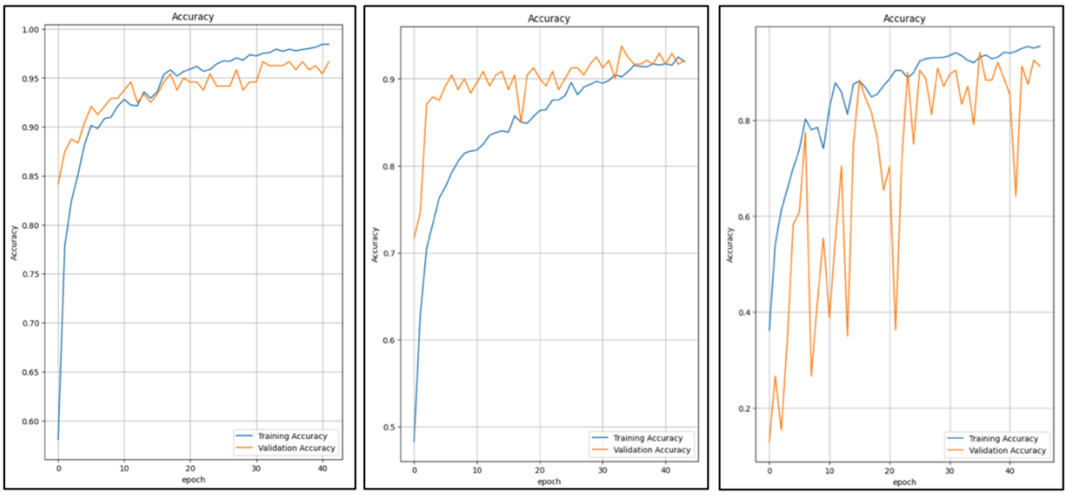
Fungsi ini digunakan untuk menghentikan pelatihan jika kriteria tertentu tidak terpenuhi dalam beberapa *epoch* berturut-turut. Ini berguna untuk mencegah *overfitting* dan menghemat waktu pelatihan. Pada penelitian ini, proses pelatihan akan dihentikan apabila nilai ‘val_accuracy’ tidak bertambah selama 10 kali pelatihan (10 *epoch*). Dan setelah pelatihan dihentikan, bobot atau *weight* terbaik selama pelatihan akan diambil sebagai bobot *final* yang akan digunakan untuk model yang dibangun.

4.5. Evaluasi Model

Evaluasi hasil pelatihan atau *training* pada ketiga model ini meliputi akurasi dan *loss* pada data *training* maupun validasi selama proses pelatihan, hasil *confusion matrix* dan *classification report* dan kesalahan prediksi selama *testing*. Yang mana selanjutnya, akan dilakukan perbandingan keseluruhan nilai *classification report* dari ketiga model untuk ditentukan model dengan hasil terbaik.

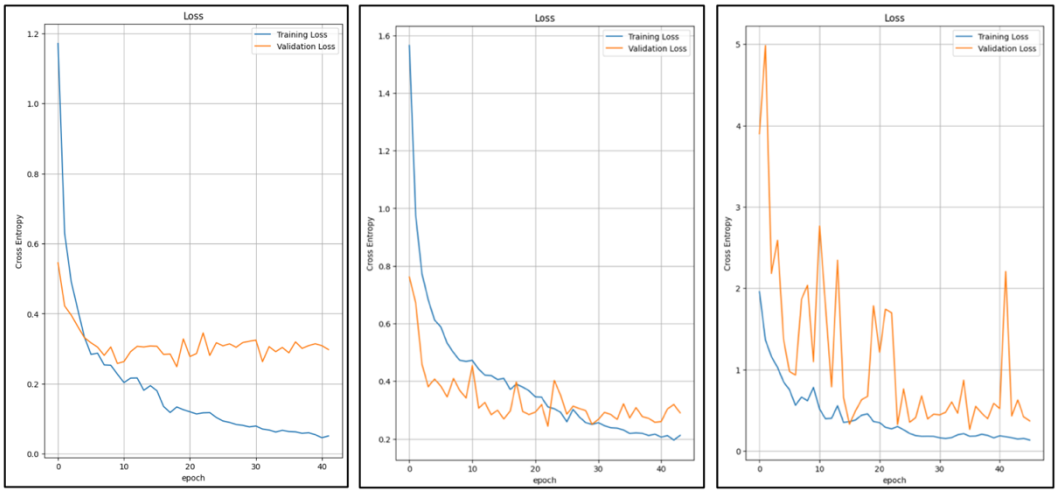
1. Akurasi dan *Loss* Pelatihan
   (a) Akurasi

   Pada data *training*, dapat terlihat bahwa dari tingkat kestabilan, grafik terbaik didapat oleh Inception-v3 dan VGG-16 memiliki kestabilan yang sama, baru setelahnya diikuti ResNet-50. Sedangkan pada data validasi, dapat terlihat grafik paling stabil diperoleh VGG-16 dan diikutin oleh Inception-v3 dan ResNet-50. Pada model dengan *transfer learning* ResNet-50, tingkat akurasi data validasinya terlihat sangat tidak stabil meskipun pada akhir pelatihan (*training*), akurasi validasi tidak berbeda jauh dengan kedua model lainnya. Perbandingan visualisasi akurasi pelatihan dapat dilihat pada Gambar 45.



Gambar 45. Perbandingan Visualisasi Hasil Pelatihan (Akurasi): A. VGG-16, B. Inception-v3, ResNet-50

(b) *Loss*



Gambar 46. Perbandingan Visualisasi Hasil Pelatihan (Loss): A. VGG-16, B. Inception-v3, ResNet-50

Pada data *training*, dapat terlihat bahwa VGG-16 dan Inception-v3 memiliki kestabilan yang sama, yang lalu diikuti oleh ResNet-50. Selanjutnya, pada data validasi, VGG-16 memiliki nilai *loss* yang lebih stabil dari pada Inception-v3 dan ResNet-50. Pada ResNet-50, dapat terlihat bahwa nilai *loss* terlihat sangat tidak stabil karena sering mengalami kenaikan dan penurunan dengan tiba-tiba dan tajam.

Berdasarkan hasil pelatihan ketiga model yang dibangun, berikut adalah hasil akhir pelatihan:

Dari table 17, model dengan jumlah *epoch* pelatihan terkecil adalah model dengan *transfer learning* VGG-16 yang mengartikan bahwa hanya diperlukan 42 pelatihan untuk mendapatkan hasil terbaik selama pelatihan. Selanjutnya, nilai

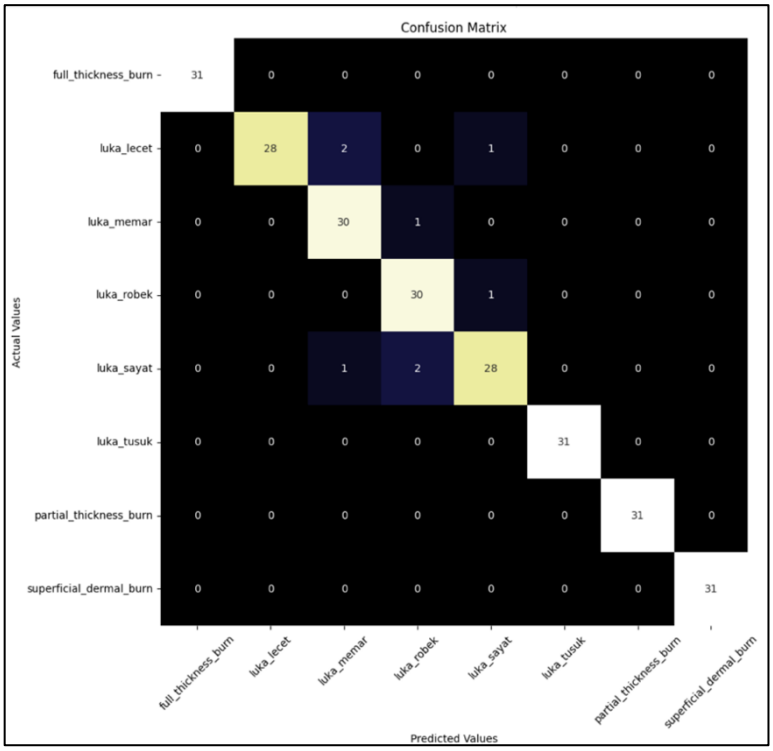
akurasi *training* maupun *validation* tertinggi didapat dari hasil pelatihan model dengan *transfer learning* VGG-16, dimana nilai akurasi *training* dan *validation*-nya masing-masing adalah 98% dan 97%. Meskipun nilai *validation loss* terendah didapat dari model dengan *transfer learning* Inception-v3 yaitu 24%, namun nilai *training loss* terendah diperoleh oleh model dengan *transfer learning* VGG-16 yaitu 4,6%.

Tabel 17. Hasil Akhir Pelatihan Ketiga Model

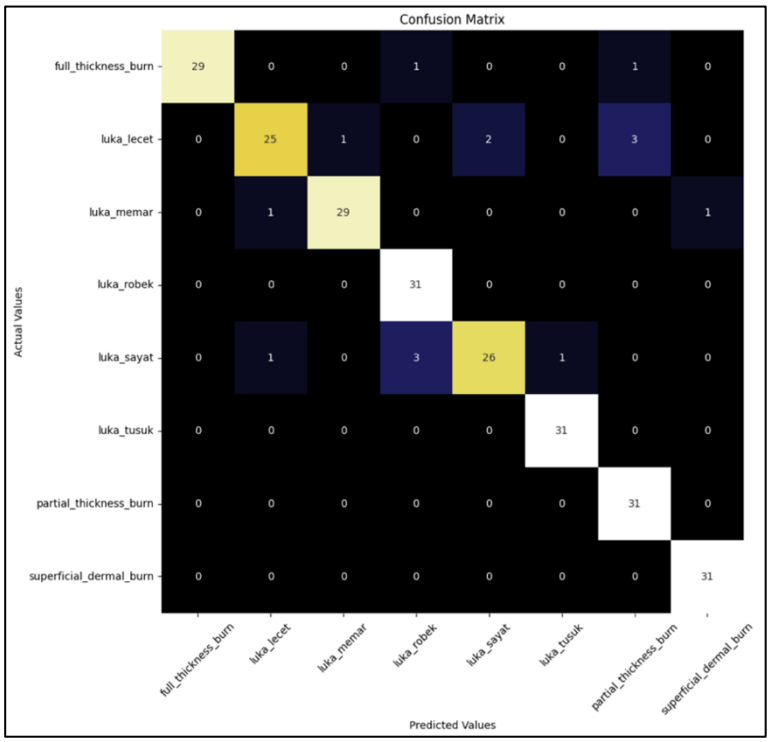
Pre-trained Model	**Jumlah *Epoch***	***Learning Rate* Akhir**	***Train***		**Validation**	
			Acc	***Loss***	***Acc***	***Loss***
VGG-16	42	1.2960×10^{-4}	98%	4,6%	97%	25%
Inception-v3	44	1.2960×10^{-4}	92,5%	20%	94%	24%
ResNet-50	46	2.1600×10^{-4}	95%	13%	94%	26%

2. *Confusion Matrix* dan *Classification Report*

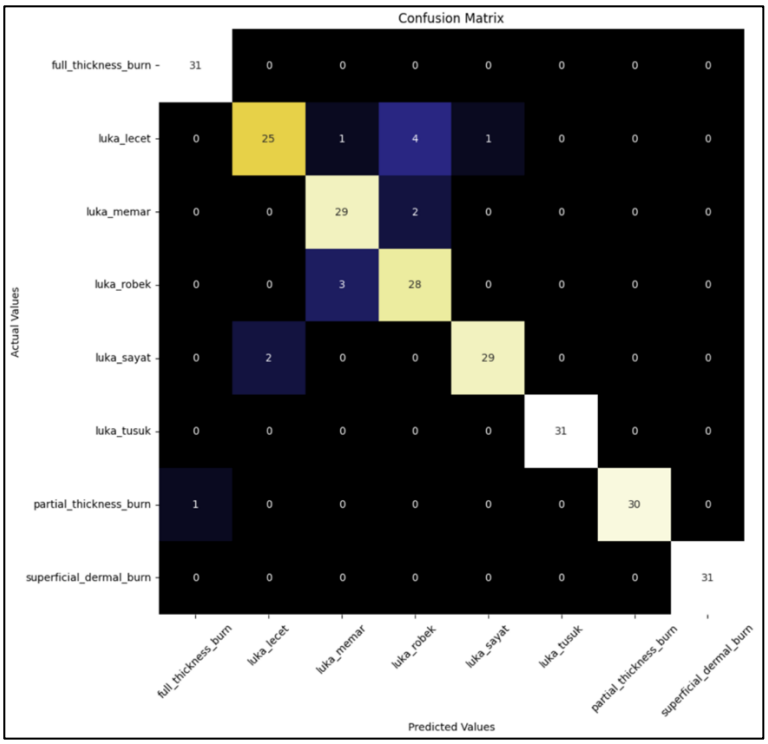
(a) *Confusion Matrix*



A



B



C

Gambar 47. Perbandingan *Heatmap Confusion Matrix*: A. VGG-16, B. Inception-v3, C. ResNet-50

Setiap model dilakukan pengujian dengan 248 data validasi dimana setiap kelasnya diuji oleh 31 data. Berdasarkan hasil *testing* pada *heatmap*

confusion matrix di atas, didapatkan lah nilai metriks pada setiap kelas dari ketiga model yang diuji:

Tabel 18. Perbandingan Nilai *True Positive* pada Ketiga Model

Model	**Banyak Data dengan Prediksi Benar (*True Positive*)**							
	Nama Kelas (Label)							
	Lc	**Sy**	**Rb**	**Ts**	**Mm**	**SB**	**PTB**	**FTB**
VGG-16	28	28	30	31	30	31	31	31
Inception-v3	25	26	31	31	29	31	31	29
ResNet-50	25	29	28	31	29	31	30	31

Keterangan nama kelas:

Lc = 'luka_lecet' Mm = 'luka_memar'

Sy = 'luka_sayat' SB = 'superficial_dermal_burn'

Rb = 'luka_robek' PTB = 'partial_thickness_burn'

Ts = 'luka_tusuk' FTB = 'full_thickness_burn'

Berdasarkan hasil pengujian tersebut, model dengan empat kelas yang memiliki nilai TP sempurna (31 data) adalah model dengan *transfer learning* VGG-16 dan Inception-v3 sedangkan pada ResNet-50, hanya tiga kelas yang memiliki nilai TP sempurna.

(b) *Classification Report*

Confusion Matrix
Classification Report

	precision	recall	f1-score	support
full_thickness_burn	1.00	1.00	1.00	31
luka_lecet	1.00	0.90	0.95	31
luka_memar	0.91	0.97	0.94	31
luka_robek	0.91	0.97	0.94	31
luka_sayat	0.93	0.90	0.92	31
luka_tusuk	1.00	1.00	1.00	31
partial_thickness_burn	1.00	1.00	1.00	31
superficial_dermal_burn	1.00	1.00	1.00	31
accuracy			0.97	248
macro avg	0.97	0.97	0.97	248
weighted avg	0.97	0.97	0.97	248

A

Confusion Matrix
Classification Report

	precision	recall	f1-score	support
full_thickness_burn	1.00	0.94	0.97	31
luka_lecet	0.93	0.81	0.86	31
luka_memar	0.97	0.94	0.95	31
luka_robek	0.89	1.00	0.94	31
luka_sayat	0.93	0.84	0.88	31
luka_tusuk	0.97	1.00	0.98	31
partial_thickness_burn	0.89	1.00	0.94	31
superficial_dermal_burn	0.97	1.00	0.98	31
accuracy			0.94	248
macro avg	0.94	0.94	0.94	248
weighted avg	0.94	0.94	0.94	248

B

Confusion Matrix
Classification Report

	precision	recall	f1-score	support
full_thickness_burn	0.97	1.00	0.98	31
luka_lecet	0.93	0.81	0.86	31
luka_memar	0.88	0.94	0.91	31
luka_robek	0.82	0.90	0.86	31
luka_sayat	0.97	0.94	0.95	31
luka_tusuk	1.00	1.00	1.00	31
partial_thickness_burn	1.00	0.97	0.98	31
superficial_dermal_burn	1.00	1.00	1.00	31
accuracy			0.94	248
macro avg	0.95	0.94	0.94	248
weighted avg	0.95	0.94	0.94	248

C

Gambar 48. Perbandingan *Classification Report* Ketiga Model: A. VGG-16, B. Inception-v3, C. ResNet-50

Berikut adalah ontoh perhitungan metriks pada kelas 'luka_sayat' model dengan *transfer learning* VGG-16 adalah sebagai berikut:

True Positive (TP) = 28, *False Positive* (FP) = 2, *False Negative* (FN) = 3

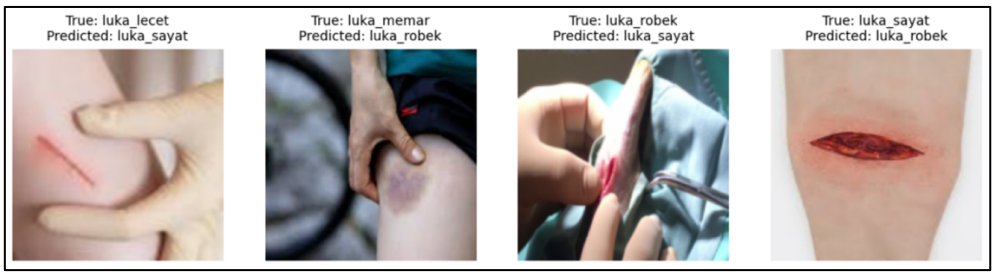
$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{28}{28 + 2} = 0{,}93$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{28}{28 + 3} = 0{,}90$$

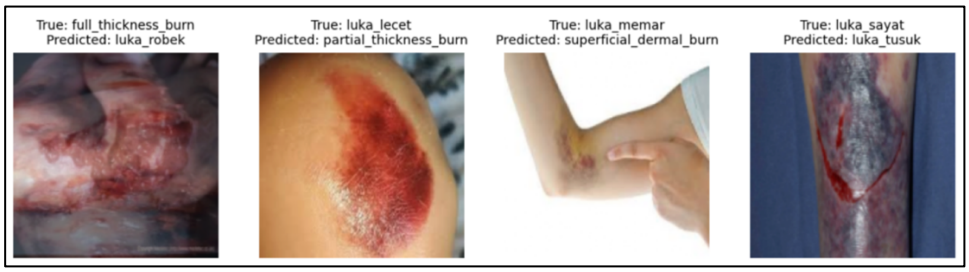
$$F1\ score = 2 \times \left(\frac{precision \times recall}{precision + recall}\right) = 2 \times \left(\frac{0{,}93 \times 0{,}9}{0{,}93 + 0{,}9}\right) = 0{,}92$$

Dari perhitungan di atas, dapat disimpulkan bahwa rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif adalah 0,93. Selanjutnya rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif adalah 0,90. Selain itu, dapat diketahui bahwa dari 31 data *test* kelas 'luka_sayat' terdapat 2 data yang diprediksi 'luka_sayat' namun kenyataannya data tersebut bukan termasuk ke dalam kelas 'luka_sayat' (*False Positive*).

(c) Kesalahan Prediksi selama *Testing*



A



B

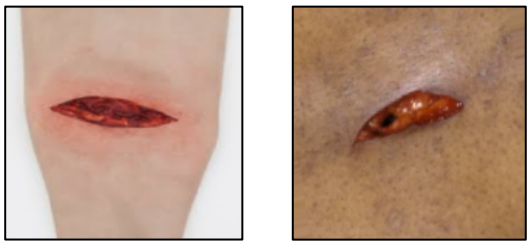
C

Gambar 49. Beberapa Kesalahan Prediksi pada Ketiga Model: A. VGG-16, B. Inception-v3, C. ResNet-50

Berdasarkan hasil *testing* yang ditunjukkan oleh *heatmap confusion matrix* dan *classification report*, pada model dengan *transfer learning* VGG-16 dan Inception-v3 hanya 4 kelas yang memiliki kesalahan prediksi sedangkan pada ResNet-50 terdapat 5 kelas yang memiliki kesalahan prediksi.

(1) Kesalahan Prediksi Luka Sayat sebagai Luka Robek, Tusuk, dan Lecet

Luka sayat dan luka robek memang memiliki tampilan yang paling mirip. Hal ini dikarenakan perbedaan tampilan antara luka sayat dan luka robek adalah besarnya pemisahan jaringan kulit dan tekstur kerusakan kulit dimana pada luka sayat,pemisahan jaringan kulit lebih kecil dan pinggiran kerusakan jaringan kulit lebih rapi dibandingkan luka robek. Contoh kemiripan pada luka sayat dan luka robek dapat dilihat pada Gambar 50.



A B

Gambar 50. Contoh Kemiripan pada Data Training Luka Sayat dan Luka Robek: A. Luka Sayat, B. Luka Robek

Selanjutnya, kesalahan prediksi gambar luka sayat sebagai luka tusuk kemungkinan terjadi karena pada data *training* luka tusuk terdapat gambar yang tidak terlalu jelas dimana pada gambar tersebut tidak terlalu terlihat bagian kerusakan kulit yang terjadi karena tusukan benda asing, sebaliknya malah terlihat garis panjang karena hasil augmentasi. Berikut adalah contoh gambar luka tusuk tersebut:

A B C

Gambar 51. Beberapa Gambar pada Data Training Luka Tusuk

Setelah itu, terdapat kesalahan pada gambar luka sayat yan terdeteksi sebagai luka lecet. Hal ini mungkin dikarenakan pada kemiripan pada luka lecet yang biasanya terdapat goresan tebal pada permukaan kulit yang rusak.

A B

Gambar 52. Contoh Kemiripan pada Data *Training* Luka Sayat dan Luka Lecet: A. Luka Sayat, B. Luka Lecet

(2) Kesalahan Prediksi Luka Lecet sebagai Luka Sayat dan *Partial Thickness Burn*

Kesalahan prediksi pada gambar luka lecet yang terdeteksi sebagai luka sayat ini telah dijelaskan pada bagain sebelumnya dimana memang terdapat beberapa kondisi luka lecet yang mana terjadi goresan garis pada tampilan kulit sehingga terlihat seperti luka sayat. Selain seperti luka sayat, gambar luka lecet juga ada yang terdeteksi sebagai *partial thickness burn.* Hal ini kemungkinan terjadi karena kemiripan pada gambar luka lecet tes tersebut yang mirip dengan beberapa tampilan pada gambar *partial thickness burn* dimana pada permukaan kulit yang rusak, biasanya terdapat gelembung/benjoal berisi air dengan

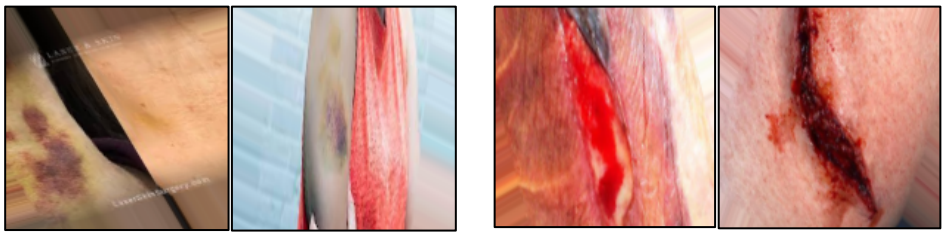
kemerahan pada kulit sekitarnya. Selain kondisi tersebut, pada *partial thickness burn*, terkadang hanya terdapat kemerahan pada kulit yang rusak namun dengan beberapa bagian kulit yang terkelupas namun tidak terlalu parah seperti *full thickness burn*. Pada beberapa data *training* di kelas *partial thickness burn,* fokus gambar hasil augmentasi yang terjadi adalah tampilan kemerahan pada kulit saja dan kurang menampilkan kondisi lain seperti gelembung berisi air yang telah disebutkan. Berikut contoh kemiripan tersebut:

A B

Gambar 53. Contoh Kemiripan pada Data *Training* Luka Lecet dan *Partial Thickness Burn*: A. Luka Lecet, B. *Partial Thickness Burn*

(3) Kesalahan Prediksi Luka Memar sebagi Luka Robek dan *Superficial Dermal Burn*

Kesalahan prediksi pada gambar luka memar yang terdeteksi sebagai luka robek ini adalah kesalahan yang mungkin adalah kesalahan yang fatal. Hal ini kemungkinan terjadi karena kesalahan atau ketidaksempurnaan pada proses *oversampling* atau augmentasi dimana terdapat beberapa gambar pada data *training* luka memar yang mengalami kerusakan/ketidaksempurnaan sehingga dideteksi sebagai luka robek. Berikut adalah contohnya:

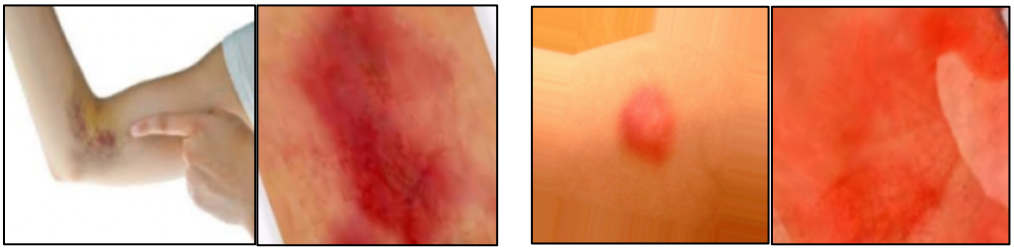


A B

Gambar 54. Contoh Kemiripan pada Data *Training* Luka Memar dan Luka Robek: A. Luka Memar, B. Luka Robek

Selain luka robek, terdapat kesalah prediksi luka memar sebagai *superficial thickness burn.* Hal ini kemungkinan terjadi karena kemiripan pada gambar luka memar tes tersebut yang mirip dengan beberapa

tampilan pada gambar *superficial dermal burn* dimana pada permukaan kulit yang rusak, terjadi perubahan warna kulit menjadi kemerahan saja, dimana pada beberapa kasus luka memar, pecahnya pembuluh darah di bawah jaringan teratas kulit ini menyebabkan kemerahan pada tampilan kulit atasnya. Berikut contoh kemiripan tersebut:



A B

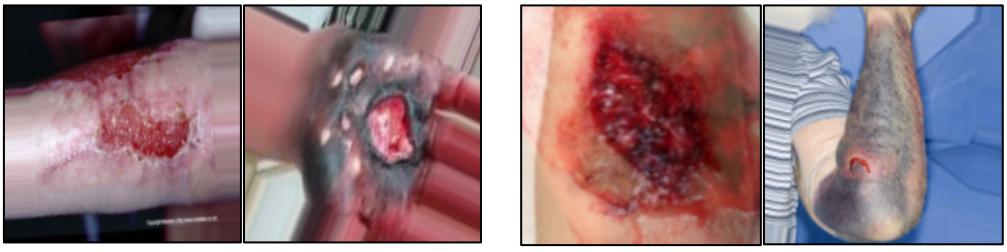
Gambar 55. Contoh Kemiripan pada Data *Training* Luka Memar dan *Superficial Dermal Burn*: A. Luka Memar, B. *Superficial Dermal Burn*

(4) Kesalahan Prediksi Luka Robek sebagai Luka Sayat dan Luka Memar

Kesalahan prediksi pada gambar luka robek yang terdeteksi sebagai luka sayat dan juga luka memar ini sudah dibahas pada penjelasan sebelumnya yaitu bagian (1) dan (3) yang menunjukkan beberapa kondisi kulit yang mirip.

(5) Kesalahan Prediksi *Full Thicknes Burn* sebagai Luka Robek

Kesalahan prediksi *full thickness burn* sebagai luka robek ini terjadi karena kondisi kulit pada luka *thickness burn* yang mana kulit yang rusak biasanya terkelupas lebih dalam sehingga tampak seperti pemisahan jaringan kulit akibat benda asing yang mana merupakan kondisi dari luka robek. Kemiripan dari kedua jenis luka ini dapat dilihat pada Gambar 57.



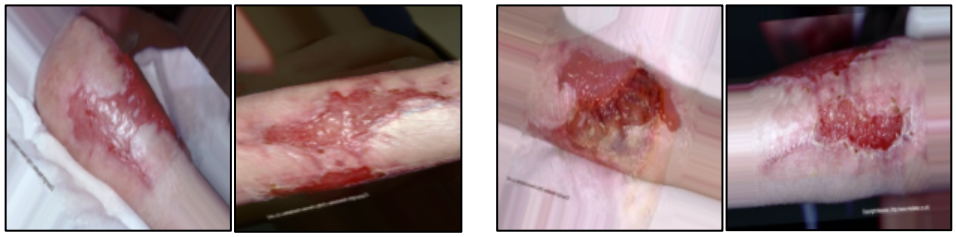
A B

Gambar 56. Kemiripan pada Data *Training Full Thickness Burn* dan Luka Robek: A. *Full Thickness Burn*, B. Luka Robek

(6) Kesalahan Prediksi *Partial Thickness Burn* sebagai *Full Thickness Burn*

Kesalahan prediksi *partial thickness burn* sebagai *full thickness*

burn ini terjadi karena kondisi kulit pada luka *thickness burn* yang mana kulit yang rusak biasanya terkelupas lebih dalam sehingga tampak seperti pemisahan jaringan kulit akibat benda asing yang mana merupakan kondisi dari luka robek. Kemiripan dari kedua jenis luka ini dapat dilihat pada Gambar 57.



A B

Gambar 57. Kemiripan pada Data Training Partial Thickness Burn dan Full Thickness Burn: A. Partial Thickness Burn, B. Full Thickness Burn

Dari kesalahan prediksi yang ditemukan, beberapa alasan yang memungkinkan adalah gambar pada dataset pelatihan (training) maupun testing yang kurang bersih (dalam artian terdapat interupsi elemen lain seperti teks atau resolusi gambar yang kurang baik). Selain itu, kemiripan kondisi kulit pada jenis-jenis luka yang digunakan juga berkemungkinan besar menjadi salah satu alasan mengapa nilai loss pada pelatihan sangat tinggi hingga terjadinya kesalahan prediksi selama proses testing. Meskipun telah dilakukan evaluasi data augmentasi dengan metriks Peak Signal to Noise Ratio (PNSR), masih terdapat data atau gambar yang tidak sepenuhnya bersih atau sesuai kriteria pada kelas asal data tersebut.

3. Perbandingan Hasil Ketiga *Pre-trained Model* berdasarkan Nilai *Confusion Matrix* dan Akurasi serta *Loss* pada Proses Pelatihan

Perbandingan dari ketiga hasil yang didapat dengan tiga *pre-trained model* berbeda yaitu VGG-16, Inception-V3, dan ResNet-50 dapat dilihat pada Tabel 17.

Tabel 19. Perbandingan Hasil Ketiga Model

Transfer Learning	***Val_loss***	***Val_acc***	***Classification Report (Confusion Matrix)***			
			Precision	***Recall***	***F1-score***	***Accuracy***
VGG-16	25%	97%	97%	97%	97%	97%
Inception-v3	24%	94%	94%	94%	94%	94%
ResNet-50	26%	94%	95%	94%	94%	94%

Berdasarakan hasil dari tabel perbandingan di atas, selama proses pelatihan didapatkan nilai *loss* validasi terendah dimiliki oleh Inception-v3 yang tidak memiliki perbedaan yang signifikan yaitu 1% dengan model lainnya sedangkan akurasi validasi tertinggi diperoleh VGG-16 dengan nilai 97% yang memiliki perbedaan 3% dengan model lainnya. Selain itu, selama pengujian yang ditunjukkan oleh *confusion matrix* dan *classification report,* ketiga model dengan *pre-trained model* berbeda dilakukan tes pada data yang sama dan mendapatkan nilai yang tertinggi yang dihasilkan oleh model dengan metode *transfer learning* VGG-16 dengan nilai *precision, recall, f1-score,* dan akurasi yang sama yaitu 97%. Oleh karena itu, model VGG-16 ini lah yang akan diimplementasikan pada GUI di penelitian ini.

4.6. Menghubungkan Model ke API

Pada tahap ini penulis membuat rancangan *request* dan *respone* yang akan menjelaskan aturan dalam proses permintaan sumber data dan hasil kembalian dari layanan. *Request* prediksi gambar dilakukan untuk meminta hasil identifikasi kelas terhadap gambar yang dikirimkan. Berikut merupakan rancangan *request* dan *response* dalam prediksi gambar:

Tabel 20. Rancangan *Request* dan *Response* Prediksi Gambar

Method	Post
URL Scheme	http
URL Host	127.0.0.1:5000
Endpoint	/predict
Headers	-
Body	image
Keterangan *Body*	Pada *body*, dilakukan penyisipan data dengan *key* bernama *image* saat *request* dilakukan. Parameter *file* ini memiliki *value* gambar berformat 'jpg', 'jpeg', 'png', atau 'jfif' dan gambar ini akan diproses oleh REST API untuk mendapatkan hasil dari pendeteksian gambar.
Format Response	JSON

Response	{
	'result': class_name
	}

Keterangan *Response*	*Class_name* pada *response* ini adalah kelas-kelas luka yang tersedia, yaitu: luka_memar, luka_sayat, luka_lecet, luka_robek, luka_tusuk, *superficial_dermal_burn*, *partial_thickness_burn*, dan '*full_thickness_burn*'

REST API dibangun dengan menggunakan *microframework* Python, yaitu Flask. Dalam *preprocessing* hingga menampilkan *output* kelas tujuan dari hasil deteksi, digunakan modul tensorflow. Untuk *preprocessing* gambar, digunakan

modul tensorflow.keras.preprocessing.image dan untuk penggunaan *trained model* dalam memprediksi gambar, digunakan fungsi model.predict() yang berasal dari modul tf.keras.Model. Kode program dan setiap fungsi yang digunakan pada program REST API dapat dilihat mulai dari Gambar 58.

Pada REST API ini akan digunakan model VGG-16 yang telah ditentukan sebelumnya dengan hasil terbaik dan berformat h5. Selain itu, didefinisikan variabel 'classes' yang akan digunakan untuk pelabelan setiap kelas. Proses ini dapat dilihat pada Gambar . Selanjutnya, dibuatlah fungsi 'predict_image' yang mana menerima sebuah parameter *input* sebuah gambar dan gambar tersebut diubah ukurannya menjadi 150x150 piksel dan setiap pikselnya diubah menjadi *array*. Selanjutnya, *array* dari gambar tersebut dilakukan pembentukan ulang (*reshape*) dengan nilai *batch* 1 dan jumlah *channel* yaitu 3. Setiap piksel tersebut bertipe data *float* sesuai dengan derajat komponen nilai RGB. Nilai gambar yang telah diolah tersebut pun akan diproses kembali dengan memanggil fungi *predict* dan mengembalikan *class* sebagai kelas tujuan dari gambar yang telah diprediksi.

```
model = tf.keras.models.load_model('model.h5')
classes = sorted([ "full_thickness_burn", "luka_memar", "luka_tusuk",
                   "luka_sayat", "luka_robek", "superficial_dermal_burn",
                   "partial_thickness_burn", "luka_lecet"])
```

Gambar 58. Pendefinisian Model dan Label Tiap Kelas

Pada fungsi 'predict', akan diterima *file* berupa *image* (gambar) dari *client* yang melakukan *request* HTTP ke *server* dengan *method* POST. Setelah gambar berhasil dibaca oleh *server*, fungsi 'predict_image' pun akan dipanggil untuk melakukan pengolahan gambar dan memprediksi gambar yang diterima yang akhirnya mengambalikan hasil atau *result* yaitu *class_name* dari gambar tersebut yang disimpan dalam format JSON.

```
def predict_image(img):
    img = load_img(io.BytesIO(img), target_size=(150, 150))
    img = img_to_array(img)
    img = img.reshape(1, 150, 150, 3)
    img = img.astype('float32')
    img = img / 255.0
    pred = model.predict(img)
    return classes[pred.argmax()]
```

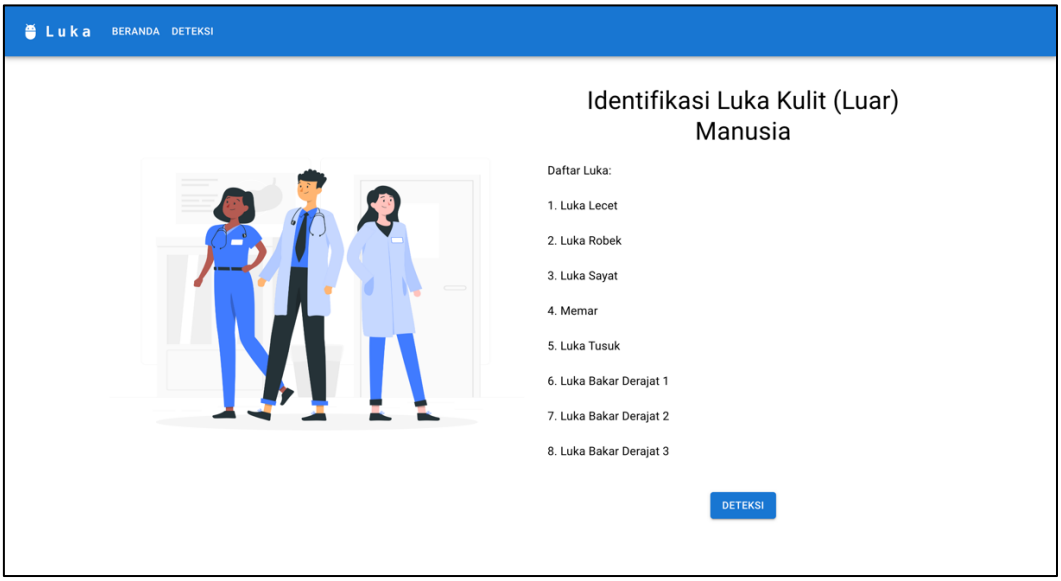
Gambar 59. Fungsi 'predict_image'

```
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        img = request.files['image'].read()
        class_name = predict_image(img)
        return jsonify({'result': class_name})
```

Gambar 60. Fungsi 'predict'

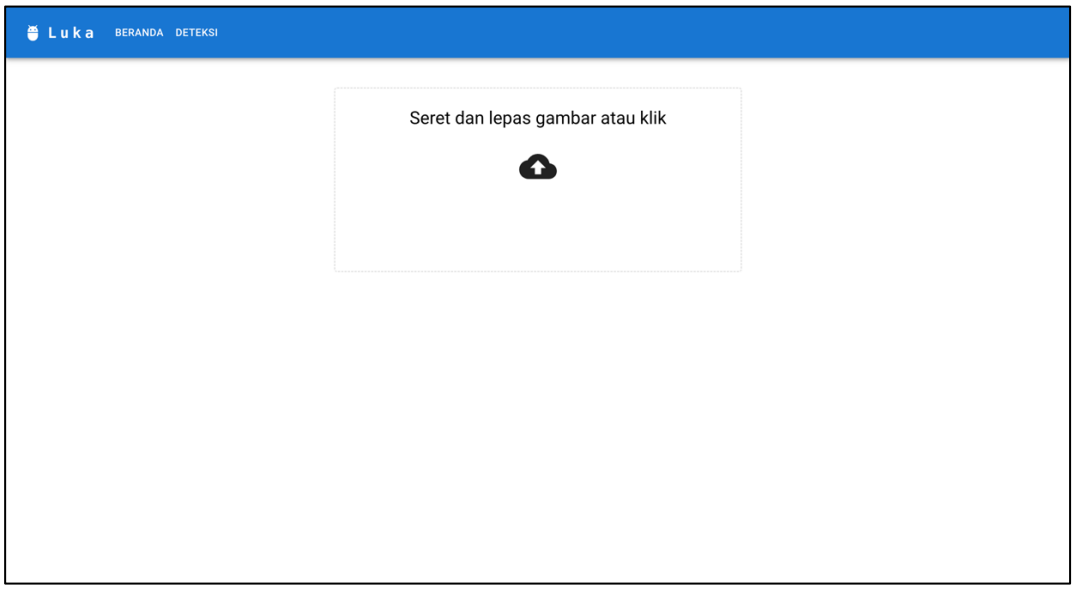
Hasil Deteksi dengan *Web*:

Pada tahap ini dilakukan pengujian dalam pendeteksian gambar dengan aplikasi berbasis *website* yang telah terintegrasi dengan REST API Model Identifikasi Luka. *Website* ini dikembangkan dengan menggunakan sebuah *microframework* Python yang bernama Flask. Dalam penelitian ini, aplikasi dijalankan pada sebuah url, yaitu http://127.0.0.1:5000. Halaman utama *web* dapat dilihat pada Gambar 61.

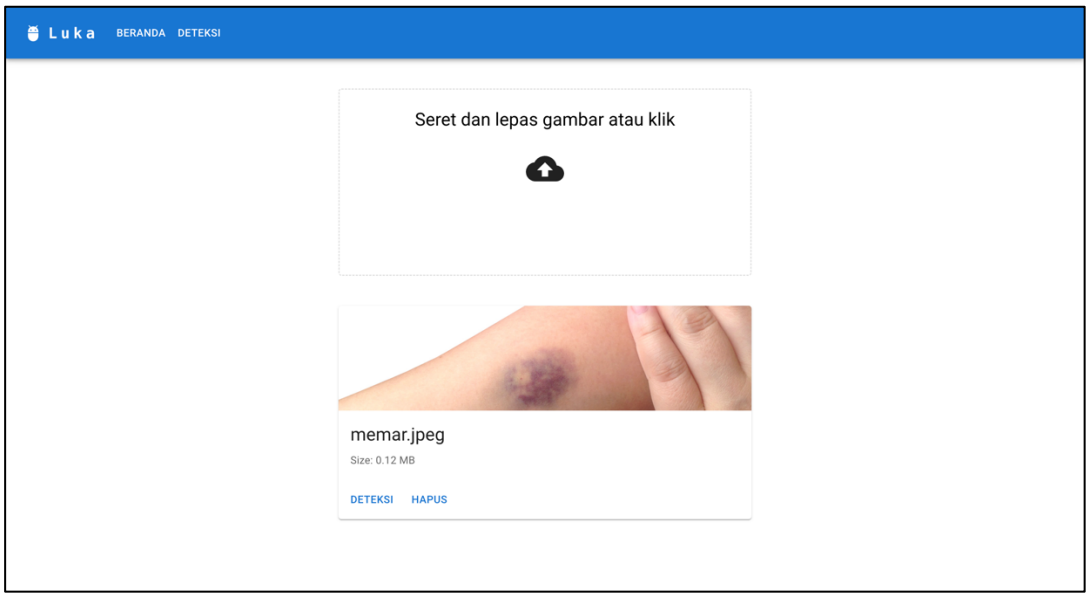


Gambar 61. Halaman *Home Web*

Pada halaman utama (*home*), terdapat *list* label atau jenis luka yang dapat dideteksi yang di bawahnya terdapat tombol 'predict' yang mengarahkan *client/user* ke halaman *predict*. Pada halaman *predict*, *user* dapat mengunggah gambar yang ingin diprediksi dengan melakukan *drag and drop* atau memindahkan gambar dari direktori lokal ke kotak *upload* atau klik kotak *upload*.



Gambar 62. Halaman *Upload Web*



Gambar 63. Halaman *Predict Web*

Setelah di-*upload*, akan ditampilkan gambar yang berhasil di-*upload* dan *user* diberi pilihan untuk mulai melakukan prediksi dengan tombol 'deteksi atau melakukan *upload* kembali dengan tombol 'hapus. Apabila *user* memilih untuk mengulang proses *upload*, *user* pun diarahkan untuk melakukan *upload* gambar kembali. Sebaliknya, jika ingiin melakukan deteksi, maka *user* akan diarahkan ke halaman *result* (hasil). Selanjutnya, pada halaman hasil, gambar yang dilakukan proses deteksi akan ditampilkan, dan di bawahnya akan ditampilkan hasil identifikasi label luka dan informasi pertolongan pertamanya seperti pada Gambar 65.

Gambar 64. Halaman *Result Web*

Sistem deteksi dan identifikasi luka luar ini diharapkan dapat membantu masyarakat dalam melakukan deteksi dini pada luka yang terjadi. Dengan sistem ini pula, masyarakat diharapkan dapat mengobati luka dengan benar sesuai dengan arahan pertolongan pertama pada luka yang berhasil terdeteksi melalui sistem deteksi dan identifikasi luka luar yang telah dibangun. Selain berdampak pada masyarakat, sistem ini juga diharapkan menjadi dasar atau pemicu penelitian lainnya yang berkaitan dengan pembelajaran mesin (*machine learning*) dan lebih kompleks, di bidang kesehatan untuk rumah sakit atau fasilitas kesehatan lainnya. Dengan penelitian berkelanjutan yang dimulai dari deteksi luka dini, diharapkan dapat memudahkan proses penanganan pertama pada luka pasien yang mungkin di kemudian hari dapat meningkatkan efektivitas dan efisiensi dengan bantuan hasil penelitian terkait *machine learning* tersebut.

V. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Selama penelitian berlangsung, pembangunan model identifikasi luka luar manusia berhasil dilakukan dengan metode *transfer learning* dengan mengimpor tiga *pre-trained model* yaitu VGG-16, Inception-v3, dan ResNet-50, yang mana selanjutnya ditambah dengan beberapa lapisan seperti *dropout*, *flatten,* dan lapisan *fully connected*.
2. Dari proses pelatihan dan pengujian tiga model yang dibangun, didapatkan hasil pelatihan dan pengujian dengan nilai pada model dengan *transfer learning* VGG16, InceptionV3, dan ResNet50 masing-masing yaitu: akurasi validasi= 97%, 94%, 94%; *loss* validasi= 25%, 24%, 26%, *precision*= 97%, 94%, 95%; *recall*= 97%, 94%, 94%; f1 *score*= 97%, 94%, 94%; dan akurasi uji= 97%, 94%, 94%. Dengan kesimpulan, model nilai terbaik yaitu model dengan *pretrained model* VGG16 dengan nilai akurasi validasi= 97%, *loss* validasi= 25%, *precision*= 97%, *recall*= 97%, *f1 score*= 97%, dan akurasi uji= 97% yang berhasil diimplementasikan pada *web* identifikasi jenis luka luar manusia.

5.2. Saran

Berdasarkan penelitian yang dilakukan, beberapa saran untuk perbaikan pada penelitian selanjutnya adalah sebagai berikut:

1. Menambahkan beberapa *hyperparameter* sebagai pembanding untuk memperoleh arsitektur CNN yang menghasilkan akurasi yang lebih baik lagi.
2. Menambahkan beberapa *layer* tambahan sebagai perbaikan proses *fine tuning* apabila menggunakan metode *transfer learning* dan memanfaatkan *pre-trained model* yang sama sebagai pembanding untuk memperoleh arsitektur CNN yang menghasilkan akurasi yang lebih baik lagi.
3. Sektor kesehatan seperti dinas kesehatan dan layanan kesehatan lainnya didorong untuk melakukan dokumentasi terhadap jenis-jenis luka baik luka ringan maupun luka berat yang lebih kompleks dan lebih bersih. Hal ini diperlukan untuk mempermudah penulis dalam memperoleh dan memperbanyak jumlah *dataset* yang dapat dipelajari oleh model sehingga pendeteksian menjadi lebih akurat.

4. Pada penelitian selanjutnya, diperlukan *dataset* yang lebih bersih dari interupsi elemen lain serta terlihat jelas pada bagian kulit yang terluka sehingga model dapat mendeteksi lebih jelas perbedaan antara jenis luka satu dengan yang lainnya.
5. Selain itu, perlu dilakukan peningkatan pada proses augmentasi dan metode evaluasi data hasil augmentasi agar data yang dihasilkan untuk proses pelatihan dapat lebih maksimal menggambarkan kelasnya dan meningkatkan performa model yang telah dibangun.

DAFTAR PUSTAKA

Aggarwal, K., Mijwil, M. M., Sonia, Al-Mistarehi, A. H., Alomari, S., Gök, M., Zein Alaabdin, A. M., & Abdulrhman, S. H. (2022). Has the Future Started? The Current Growth of Artificial Intelligence, Machine Learning, and Deep Learning. *Iraqi Journal for Computer Science and Mathematics*, *3*(1), 115–123. https://doi.org/10.52866/ijcsm.2022.01.01.013

Agustina, D., Mustafidah, H., & Purbowati, M. R. (2016). Sistem Pakar Diagnosa Penyakit Kulit Akibat Infeksi Jamur. *Juita*, *IV*(2), 67–77.

Alzubaidi, L., Fadhel, M. A., Oleiwi, S. R., Al-Shamma, O., & Zhang, J. (2020). DFU_QUTNet: diabetic foot ulcer classification using novel deep convolutional neural network. *Multimedia Tools and Applications*, *79*(21–22), 15655–15677. https://doi.org/10.1007/s11042-019-07820-w

Anggraini, N. A., Mufidah, A., Putro, D. S., Permatasari, I. S., Putra, I. N. A., Hidayat, M. A., Kusumaningrum, R. W., Prasiwi, W. F., & Suryanto, A. (2018). Pendidikan Kesehatan Pertolongan Pertama pada Kecelakaan pada Masyarakat di Kelurahan Dandangan. *Journal of Community Engagement in Health*, *1*(2), 21–24. https://doi.org/10.30994/jceh.v1i2.10

Anisuzzaman, D. M., Patel, Y., Niezgoda, J. A., Gopalakrishnan, S., & Yu, Z. (2022). A Mobile App for Wound Localization Using Deep Learning. *IEEE Access*, *10*, 61398–61409. https://doi.org/10.1109/ACCESS.2022.3179137

Bera, S., & Shrivastava, V. K. (2020). Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *International Journal of Remote Sensing*, *41*(7), 2664–2683. https://doi.org/10.1080/01431161.2019.1694725

Biomi, A. A., & Swandewi, N. P. D. (2020). Faktor Penyebab Terjadinya Kecelakaan Pada Murid Tk Denpasar. *Bali Health Journal*, *3*(2).

Cholissodin, I., & Soebroto, A. A. (2021). *AI , MACHINE LEARNING & DEEP LEARNING (Teori & Implementasi). July 2019*.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002). Unraveling the Web services Web: An introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, *6*(2), 86–93. https://doi.org/10.1109/4236.991449

Davison, A. M. (2004). The Incised Wound. *Essentials of Autopsy Practice*, 187–220. https://doi.org/10.1007/978-1-4471-0637-1_8

Deeny, P. (1994). *gunshot and stab wounds*. *10*, 523–525.

Defi, I. R., Iskandar, S., Charismawati, S., Turnip, A., & Novita, D. (2022).

Healthcare Workers' Point of View on Medical Robotics During COVID-19 Pandemic – A Scoping Review. *International Journal of General Medicine*, *15*(March), 3767–3777. https://doi.org/10.2147/IJGM.S355734

Despo, O. (2015). *BURNED : Efficient and Accurate Burn Prognosis Using Deep Learning*. *77*(2008), 3440.

Diwan, T., Anirudh, G., & Tembhurne, J. V. (2023). Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, *82*(6), 9243–9275. https://doi.org/10.1007/s11042-022-13644-y

Elreedy, D., & Atiya, A. F. (2019). A Comprehensive Analysis of Synthetic Minority Oversampling Technique (SMOTE) for handling class imbalance. *Information Sciences*, *505*, 32–64. https://doi.org/10.1016/j.ins.2019.07.070

Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*, *61*, 863–905. https://doi.org/10.1613/jair.1.11192

Fisichella, M., Deng, F., & Nejdl, W. (2010). Efficient incremental near duplicate detection based on locality sensitive hashing. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6261 LNCS*(PART 1), 152–166. https://doi.org/10.1007/978-3-642-15364-8_11

Fourcade, A., & Khonsari, R. H. (2019). Deep learning in medical image analysis: A third eye for doctors. *Journal of Stomatology, Oral and Maxillofacial Surgery*, *120*(4), 279–288. https://doi.org/10.1016/j.jormas.2019.06.002

Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. *O'Reilly Media*, ```.

Goldberg, S. R., & Diegelmann, R. F. (2020). What Makes Wounds Chronic. *Surgical Clinics of North America*, *100*(4), 681–693. https://doi.org/10.1016/j.suc.2020.05.001

Goyal, M., Reeves, N. D., Davison, A. K., Rajbhandari, S., Spragg, J., & Yap, M. H. (2018). DFUNet: Convolutional Neural Networks for Diabetic Foot Ulcer Classification. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *4*(5), 728–739. https://doi.org/10.1109/tetci.2018.2866254

Great Learning. (2021). *Everything you need to know about VGG16*. Medium.Com. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 770–778. https://doi.org/10.1109/CVPR.2016.90

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9908 LNCS*, 630–645. https://doi.org/10.1007/978-3-319-46493-0_38

Irfan, D., Rosnelly, R., Wahyuni, M., Samudra, J. T., & Rangga, A. (2022). Perbandingan Optimasi Sgd, Adadelta, Dan Adam Dalam Klasifikasi Hydrangea Menggunakan Cnn. *Journal of Science and Social Research*, *5*(2), 244. https://doi.org/10.54314/jssr.v5i2.789

Jennett, P. A., Affleck Hall, L., Hailey, D., Ohinmaa, A., Anderson, C., Thomas, R., Young, B., Lorenzetti, D., & Scott, R. E. (2003). The socio-economic impact of telehealth: A systematic review. *Journal of Telemedicine and Telecare*, *9*(6), 311–320. https://doi.org/10.1258/135763303771005207

Jumlah Kecelakaan, Korban Mati, Luka Berat, Luka Ringan, dan Kerugian Materi 2019-2021. (2023). Badan Pusat Statistik (BPS - Statistics Indonesia). https://www.bps.go.id/indicator/17/513/1/jumlah-kecelakaan-korban-mati-luka-berat-luka-ringan-dan-kerugian-materi.html

Khairina, N., Harahap, M. K., & Lubis, J. H. (2018). The Authenticity of Image using Hash MD5 and Steganography Least Significant Bit. *IJISTECH (International Journal Of Information System & Technology)*, *2*(1), 1. https://doi.org/10.30645/ijistech.v2i1.13

Kido, S., Hirano, Y., & Hashimoto, N. (2018). Detection and classification of lung abnormalities by use of convolutional neural network (CNN) and regions with CNN features (R-CNN). *2018 International Workshop on Advanced Image Technology, IWAIT 2018*, 1–4. https://doi.org/10.1109/IWAIT.2018.8369798

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Li, J., Chen, J., & Kirsner, R. (2007). Pathophysiology of acute wound healing.

Clinics in Dermatology, *25*(1), 9–18. https://doi.org/10.1016/j.clindermatol.2006.09.007

Lin, W. C., Tsai, C. F., Hu, Y. H., & Jhang, J. S. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences*, *409–410*, 17–26. https://doi.org/10.1016/j.ins.2017.05.008

Liu, Y., Yu, Y., Wang, L., Nyima, T., Zhaxi, N., Huang, H., & Deng, Q. (2020). Classification of Tank Images Using Convolutional Neural Network. *Proceedings of 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2020*, 210–214. https://doi.org/10.1109/ITNEC48623.2020.9085151

Manning, J. (2018). Sepsis in the Burn Patient. *Critical Care Nursing Clinics of North America*, *30*(3), 423–430. https://doi.org/10.1016/j.cnc.2018.05.010

Mao, X., Yamada, Y., Akiyama, Y., & Okamoto, S. (2017). Characteristics of Dummy Skin Contact Mechanics During Developing Process of Skin Abrasion Trauma. *Tribology Letters*, *65*(4), 1–12. https://doi.org/10.1007/s11249-017-0916-7

Mashita, S. N. (2020). Implementasi Deep Learning Object Detection Rambu K3 Pada Video Menggunakan Metode Convolutional Neural Network (CNN) dengan Tensorflow. *Skripsi, Statistika, Fakultas Matematika Dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia, Yogyakarta*, i–89. https://dspace.uii.ac.id/handle/123456789/28781

Mayo Clinic. (2022a). *Bruise: First aid*. St. Clair Health. https://www.stclair.org/services/mayo-clinic-health-information/article/ART-20056663

Mayo Clinic. (2022b). *Burns : First aid From Mayo Clinic to your inbox Sign up for free , and stay up to date on research*. Mayo Clinic. https://www.mayoclinic.org/first-aid/first-aid-burns/basics/art-20056649

Muttaqien, K., Sugiarto, & Sarifudin, S. (2019). Upaya Meningkatkan Kesadaran Masyarakat Terhadap Kesehatan Lingkungan Melalui Program Bank Sampah. *Indonesian Journal of Adult and Community Education*, *1*(1), 6–10. https://ejournal.upi.edu/index.php/IJACE/article/view/19997

Nasution, R. E. P. (2020). *Panduan Bantuan Hidup Dasar dan Pertolongan Pertama Pada Luka*. Whitecoathunter. https://books.google.co.id/books?id=icHdDwAAQBAJ

Ners. I Made Sukma Wijaya, M. K. W. O. C. N. (2018). *Perawatan Luka Dengan Pendekatan Multidisiplin*. Penerbit Andi. https://books.google.co.id/books?id=pVJtDwAAQBAJ

Paraijun, F., Aziza, R. N., & Kuswardani, D. (2022). Implementasi Algoritma

Convolutional Neural Network Dalam Mengklasifikasi Kesegaran Buah Berdasarkan Citra Buah. *Kilat*, *11*(1), 1–9. https://doi.org/10.33322/kilat.v10i2.1458

Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of Deep Learning for Object Detection. *Procedia Computer Science*, *132*(Iccids), 1706–1717. https://doi.org/10.1016/j.procs.2018.05.144

Petroc Taylor. (2023). *Number of smartphone mobile network subscriptions worldwide from 2016 to 2022, with forecasts from 2023 to 2028*. Statista. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

Phillip L Rice, Jr, MDDennis P Orgill, MD, P. (2023). *Assessment and classification of burn injury*. UpToDate. https://www.uptodate.com/contents/assessment-and-classification-of-burn-injury/print#:~:text=Superficial or epidermal burns involve only the epidermal layer of skin.,-•&text=Partial-thickness burns involve the epidermis and portions of the dermis.,-•&text

Riset: Kesadaran Masyarakat Indonesia akan Kebersihan Masih Rendah. (2018). Badan Penelitian Dan Pengembangan Kementrian Dalam Negeri Indonesia. https://litbang.kemendagri.go.id/website/riset-kesadaran-masyarakat-indonesia-akan-kebersihan-masih-rendah/

Rozzi, H. V. (2014). Laceration or Incised Wound: Know the Difference. In *American College of Emergency Physicians*.

Salomon, D. (2011). Data Compression. In *Handbook of Computer Networks* (Vol. 1). https://doi.org/10.1002/9781118256053.ch13

Shelke, M. S., Deshmukh, P. R., & Shandilya, P. V. K. (2017). A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique. *International Journal of Recent Trends in Engineering and Research*, *3*(4), 444–449. https://doi.org/10.23883/ijrter.2017.3168.0uwxm

Shenoy, V. N., Foster, E., Aalami, L., Majeed, B., & Aalami, O. (2019). Deepwound: Automated Postoperative Wound Assessment and Surgical Site Surveillance through Convolutional Neural Networks. *Proceedings - 2018 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018*, 1017–1021. https://doi.org/10.1109/BIBM.2018.8621130

Shrestha, R., Krishan, K., & Kanchan, T. (2022). Abrasion - StatPearls - NCBI Bookshelf. In *StatPearls*. https://www.ncbi.nlm.nih.gov/books/NBK554465/

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for

large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–14.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 2818–2826. https://doi.org/10.1109/CVPR.2016.308

Taber, J. M., Leyva, B., & Persoskie, A. (2015). Why do People Avoid Medical Care? A Qualitative Study Using National Data. *Journal of General Internal Medicine*, *30*(3), 290–297. https://doi.org/10.1007/s11606-014-3089-1

Tammina, S. (2019). Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *International Journal of Scientific and Research Publications (IJSRP)*, *9*(10), p9420. https://doi.org/10.29322/ijsrp.9.10.2019.p9420

Tammina, S. (2022). CovidSORT: Detection of Novel COVID-19 in Chest X-ray Images by Leveraging Deep Transfer Learning Models. *Lecture Notes in Electrical Engineering*, *783*(October), 431–447. https://doi.org/10.1007/978-981-16-3690-5_37

Tintinalli, J. E., Stapczynski, J. S., Ma, O. J., Cline, D. M., & Meckler, G. D. (2016). *Tintinalli's Emergency Medicine: A Comprehensive Study Guide, 8th edition*. McGraw Hill LLC. https://books.google.co.id/books?id=FNKLCgAAQBAJ

Wagh, K., & Thool, R. (2012). A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host. *Journal of Information Engineering and Applications*, *2*(5), 12–16. http://www.iiste.org/Journals/index.php/JIEA/article/view/2063

Wang, C., Yan, X., Smith, M., Kochhar, K., Rubin, M., Warren, S. M., Wrobel, J., & Lee, H. (2015). A unified framework for automatic wound segmentation and analysis with deep convolutional neural networks. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, *2015-Novem*(August), 2415–2418. https://doi.org/10.1109/EMBC.2015.7318881

Wang, G., Yang, F., Zhou, W., Xiao, N., Luo, M., & Tang, Z. (2023). The initiation of oxidative stress and therapeutic strategies in wound healing. *Biomedicine and Pharmacotherapy*, *157*(October 2022), 114004. https://doi.org/10.1016/j.biopha.2022.114004

WebMD Editorial Contributors. (2021). *Bruises*. WebMD LLC. https://www.webmd.com/skin-problems-and-treatments/guide/bruises-article#:~:text=For Your Skin%3F-,Symptoms of a Bruise,away as the color

fades.

Wikarta, A., Sigit Pramono, A., & Ariatedja, J. B. (2020). Analisa Bermacam Optimizer Pada Convolutional Neural Network Untuk Deteksi Pemakaian Masker Pengemudi Kendaraan. *Seminar Nasional Informatika*, *2020*(Semnasif), 69–72.

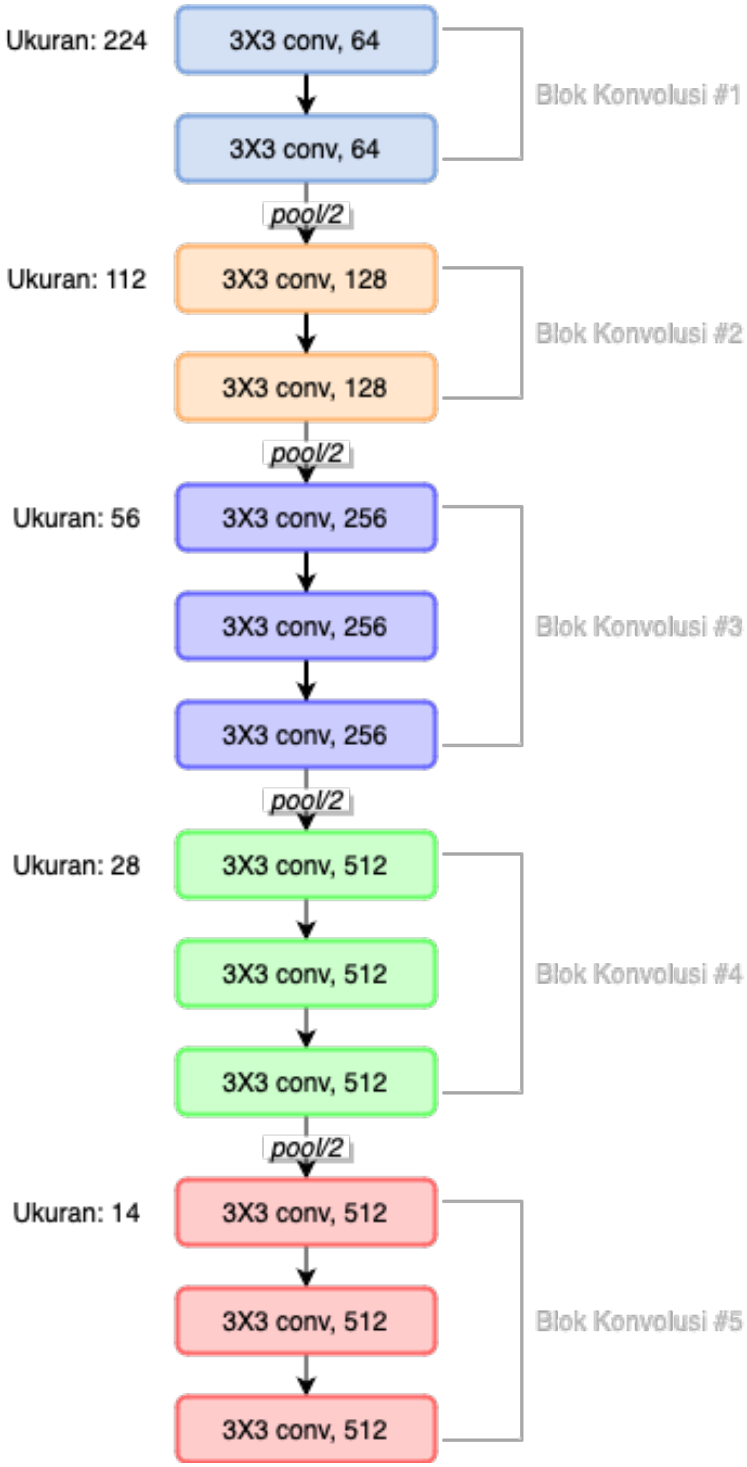
Yu, W., Jin, Y., Yang, J., Ma, G., Qiu, Y., Chen, H., Yang, X., Chang, L., & Lin, X. (2017). Occurrence of bruise, hematoma, and pain in upper blepharoplasty using blunt-needle vs sharp-needle anesthetic injection in upper blepharoplasty: A randomized clinical trial. *JAMA Facial Plastic Surgery*, *19*(2), 128–132. https://doi.org/10.1001/jamafacial.2016.1376

Zhi-Hua Zhou - Machine Learning-Springer Singapore (2021). (n.d.).

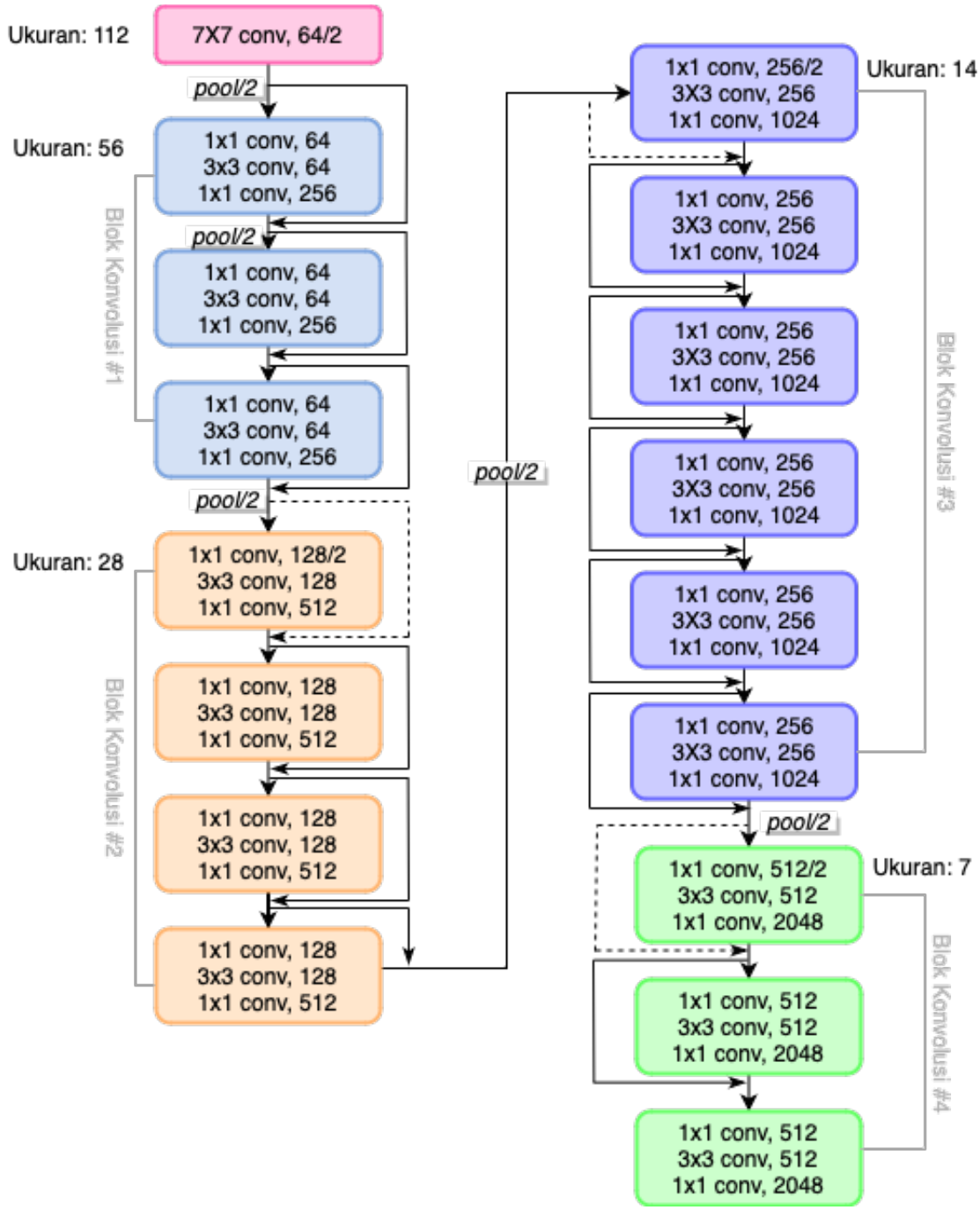
Zhu, J., Zhong, K., Zong, Y., Wang, S., Yang, H., Zhen, L., Tao, S., Sun, L., Yang, J., & Li, J. (2022). A mussel-inspired wet-adhesion hydrogel with hemostasis and local anti-inflammation for managing the development of acute wounds. *Materials and Design*, *213*, 110347. https://doi.org/10.1016/j.matdes.2021.110347

LAMPIRAN

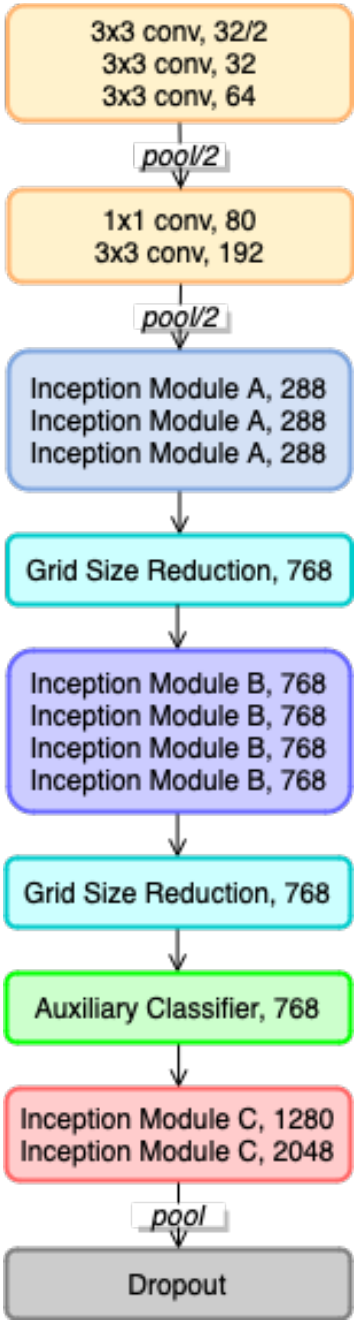
Lampiran 1. Arsitektur VGG-16 yang Digunakan



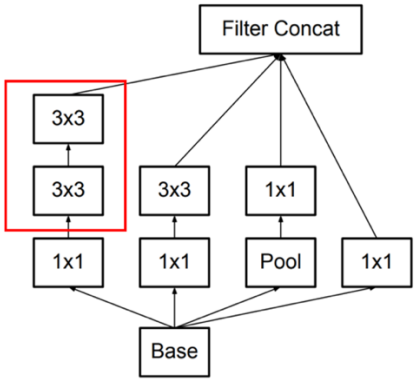
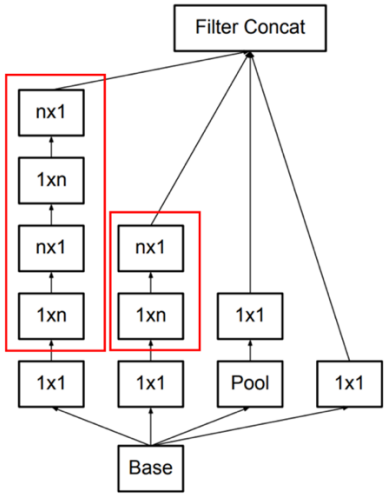
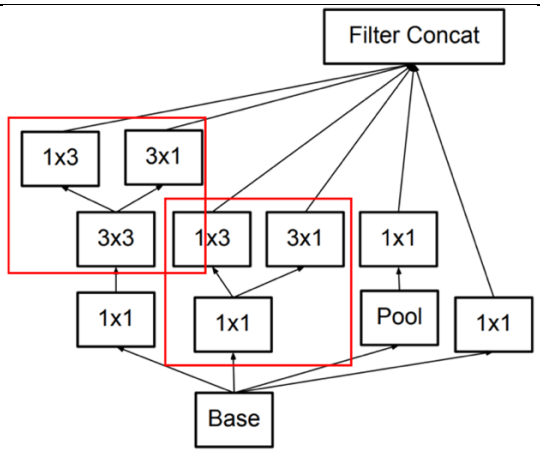
Lampiran 2. Arsitektur ResNet-50 yang Digunakan

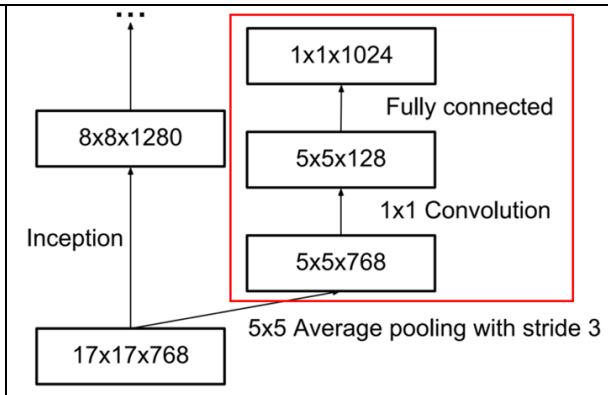
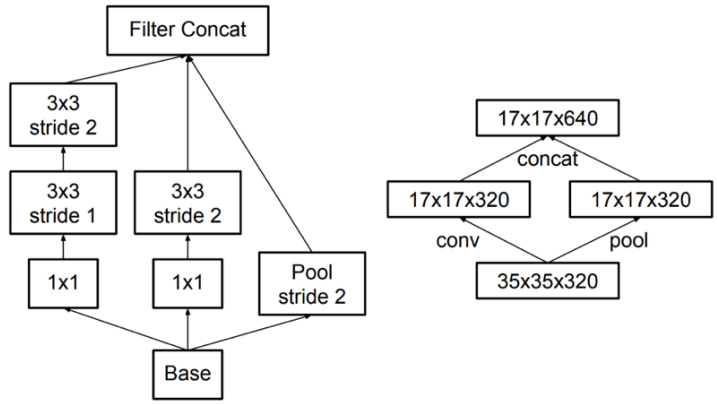


Lampiran 3. Arsitektur Inception-v3 yang Digunakan



Lampiran 4. Rincian Modul pada Arsitektur Inception-v3 yang Digunakan

Inception *Module*	**Arsitektur**
Inception Module A	
Inception Module B	
Inception Module C	

Auxiliary Classifier	
Grid Size Reduction	

Lampiran 5. Kode Program Pembangunan Model

```
"""DATA PREPARATION"""
"""Load Data"""
from google.colab import drive
drive.mount('/content/drive/')

"""Extract Dataset"""
!unzip '/content/drive/MyDrive/Dataset/Dataset.zip'

"""Data Understanding"""
"Check length of each label"
import os
path = '/content/Dataset'
dataset_dir = os.listdir(path)
total_data = 0  #save total data
for i in dataset_dir:
    label_dir = os.path.join(path, i)
    if os.path.isdir(label_dir):
        files = os.listdir(label_dir)
        num_files = len(files)
        total_data += num_files  # add number of data to total_data
        a = '{} label has a total of '.format(i).ljust(50)
        a += ':'
        print(a, num_files, ' images')
    else:
        print('\nPath {} is not a directory.\n'.format(label_dir))

# after looping is done, print the result
print("\nTotal data across all labels:", total_data)

# Recursively delete all .DS_Store files from the given directory
and its subdirectories
def delete_ds_store(path):
    for root, dirs, files in os.walk(path):
        for file in files:
            if file == ".DS_Store":
                os.remove(os.path.join(root, file))
                print(f"Deleted .DS_Store file: {os.path.join(root,
file)}")
# Call the function to delete .DS_Store files in your desired
directory
delete_ds_store(path)

"Plot image from each label"
import numpy as np
import cv2
import os
from matplotlib import pyplot as plt
import matplotlib.image as mpimg
import random
from math import ceil

# create figure
fig = plt.figure(figsize=(10, 10))
```

```
# show image
cols = 4
rows = ceil(len(dataset_dir) / cols)
x = 0
for i in dataset_dir:
  label_dir = os.path.join(path, i)
  if os.path.isdir(label_dir):
    files = os.listdir(label_dir)
    # rest of the code
    # pick random image
    rnd = random.choice(files)
    imgg = os.path.join(label_dir, rnd)
    Image = mpimg.imread(imgg)
    plt.subplot(cols, rows, x+1)
    plt.imshow(Image)
    plt.title(i)
    plt.axis('off')
    x += 1
  else:
    print("Skipping non-directory file:", label_dir)

"""DATA PREPROCESSING"""
"""Remove duplicate images from each label"""
!pip install Pillow
import hashlib
from matplotlib.pyplot import imread
import imageio
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import numpy as np
import shutil

"Define the functions"
def find_duplicates(col_name):
    os.chdir(col_name)
    os.getcwd()
    files_list = os.listdir('.')
    print("Number of files in the current directory:",
len(files_list))

    duplicates = []
    hash_keys = dict()
    for index, filename in enumerate(os.listdir('.')):
        if os.path.isfile(filename):
            with open(filename, 'rb') as f:
                filehash = hashlib.md5(f.read()).hexdigest()
            if filehash not in hash_keys:
                hash_keys[filehash] = index
            else:
                duplicates.append((index, hash_keys[filehash]))
    if not duplicates:
        print('Result of Duplicates: 0')
```

```
    else:
        duplicates.sort(key=lambda x: x[1])
        print('Result of Duplicates:', duplicates)
        return duplicates, files_list

# Define a function to move files to a new directory
def move_dups_to_new_dir(source_dir, destination_dir, dup_indexes):
    for index in dup_indexes:
        source_file = files_list[index]
        destination_file = os.path.join(destination_dir,
os.path.basename(source_file))
        shutil.move(source_file, destination_file)

"Checking and Moving The Duplicates"
path = '/content/Dataset'
duplicate_dir = '/content/DuplicateFiles'

# Loop through all directories in the path
for col_name in os.listdir(path):
    if os.path.isdir(os.path.join(path, col_name)):
        col_path = os.path.join(path, col_name)
        print(f"Checking duplicates in {col_path}")

        result = find_duplicates(col_path)
        if result is not None:
            duplicates, files_list = result
            os.makedirs(duplicate_dir, exist_ok=True) # Create the
duplicate directory if it doesn't exist
            # Showing result of duplicates
            for file_indexes in duplicates[:30]:
                try:
                    file_index_0, file_index_1 = file_indexes
                    if file_index_0 < len(files_list) and
file_index_1 < len(files_list):
                        plt.subplot(121),
plt.imshow(imread(files_list[file_index_1]))
                        plt.title('index {}'.format(file_index_1)),
plt.xticks([]), plt.yticks([])
                        plt.subplot(122),
plt.imshow(imread(files_list[file_index_0]))
                        plt.title('index {} is the
duplicate'.format(file_index_0)), plt.xticks([]), plt.yticks([])
                        plt.show()
                except OSError as e:
                    continue
            # Move duplicate files to the new directory
            move_dups_to_new_dir(col_path, duplicate_dir, [index[0]
for index in duplicates])
        print()

"Checking each label's length"
import os
```

```
path = '/content/Dataset'
dataset_dir = os.listdir(path)
total_data = 0  # Inisialisasi variabel untuk menyimpan total data

for i in dataset_dir:
    label_dir = os.path.join(path, i)
    if os.path.isdir(label_dir):
        files = os.listdir(label_dir)
        num_files = len(files)
        total_data += num_files  # Menambahkan jumlah data label ke
total_data
        a = '{} label has a total of '.format(i).ljust(50)
        a += ':'
        print(a, num_files, ' images')
    else:
        print('\nPath {} is not a directory.\n'.format(label_dir))

print("\nTotal data across all labels:", total_data)

"""SMOTE for imbalanced dataset"""
import pandas as pd
from skimage.transform import resize
from skimage.io import imread
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from collections import Counter

Categories_1= os.listdir(path)

flat_data_arr_1=[]
target_arr_1=[]
for i in Categories_1:
  print(f'loading... category : {i}')
  path1 = os.path.join(path,i)
  for img in os.listdir(path1):
    if img != '.DS_Store':
      img_array=imread(os.path.join(path1,img))
      img_resized=resize(img_array,(150,150,3))
      flat_data_arr_1.append(img_resized.flatten())
      target_arr_1.append(Categories_1.index(i))
  print(f'loaded category:{i} successfully')
flat_data_1=np.array(flat_data_arr_1)
target=np.array(target_arr_1)

df_1=pd.DataFrame(flat_data_1)
df_1['Target']=target
df_1

x_1=df_1.iloc[:,:-1]
y_1=df_1.iloc[:,-1]

y_1 = LabelEncoder().fit_transform(y_1)
y_1
```

```
"Oversampling Methode"
from imblearn.over_sampling import SMOTE
from collections import Counter

oversample_1 = SMOTE()
x_1, y_1 = oversample_1.fit_resample(x_1, y_1)
# summarize the new class distribution
counter = Counter(y_1)
print(counter)

"Transforming into pictures again"
# Create new dir for dataset after being resampled
!mkdir '/content/new_dataset_after_resampling'

from PIL import Image
new_dir = '/content/new_dataset_after_resampling'
for i in range(x_1.shape[0]):
  if not os.path.isdir(os.path.join(new_dir, Categories_1[y_1[i]])):
    os.mkdir(os.path.join(new_dir, Categories_1[y_1[i]]))
  name_dir = os.path.join(new_dir, Categories_1[y_1[i]])
  a = np.array(x_1.iloc[i]).reshape(150,150,3)
  img = Image.fromarray((a * 255).astype(np.uint8))
  img = img.save(os.path.join(name_dir, Categories_1[y_1[i]]
+'{}.png'.format(i)))

dataset_dir = os.listdir(new_dir)
total_data = 0  # Inisialisasi variabel untuk menyimpan total data

for i in dataset_dir:
    label_dir = os.path.join(new_dir, i)
    if os.path.isdir(label_dir):
        files = os.listdir(label_dir)
        num_files = len(files)
        total_data += num_files  # Menambahkan jumlah data label ke
total_data
        a = '{} label has a total of '.format(i).ljust(50)
        a += ':'
        print(a, num_files, ' images')
    else:
        print('\nPath {} is not a directory.\n'.format(label_dir))

# Setelah selesai perulangan, cetak total keseluruhan data
print("\nTotal data across all labels:", total_data)


"""Augmentation Data"""
!pip install split-folders

import splitfolders
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
# Augmentation Parameters
BATCH_SIZE = 16
TARGET_SIZE = 150
AUGMENTATION_FACTOR = 8  # Number of augmentations

dataset_dir = '/content/new_dataset_after_resampling'
output_dir = '/content/Dataset_split'

# Splitting Dataset
splitfolders.ratio(dataset_dir, output=output_dir, seed=42,
ratio=(0.8, 0.2), group_prefix=None)
train_dir = '/content/Dataset_split/train'
val_dir = '/content/Dataset_split/val'

# Create new dir for final augemented dataset
augmented_train_dir = '/content/Dataset_split/train_augmented'
os.makedirs(augmented_train_dir, exist_ok=True)

# Create subdir in augmented_train_dir (for classes)
class_names = os.listdir(dataset_dir)
for class_name in class_names:
    class_augmented_dir = os.path.join(augmented_train_dir,
class_name)
    os.makedirs(class_augmented_dir, exist_ok=True)

# Calling classes function
def get_class_name_from_image(image_path):
    # Ambil nama direktori terakhir dari path gambar sebagai nama
kelas
    class_name = os.path.basename(os.path.dirname(image_path))
    return class_name

# Inisialitation Augementation Generator
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.4,
    zoom_range=0.4,
    rotation_range=40,
    horizontal_flip=True,
    fill_mode='nearest',
)
val_datagen = ImageDataGenerator(
    rescale=1./255,
)

# Augmentation Process
for root, _, files in os.walk(train_dir):
    for file in files:
        image_path = os.path.join(root, file)
        img = tf.keras.preprocessing.image.load_img(image_path,
target_size=(TARGET_SIZE, TARGET_SIZE))
        x = tf.keras.preprocessing.image.img_to_array(img)
        x = x.reshape((1,) + x.shape)
```

```
        i = 0
        class_name = get_class_name_from_image(image_path)
        class_augmented_dir = os.path.join(augmented_train_dir,
class_name)
        os.makedirs(class_augmented_dir, exist_ok=True)
        for batch in train_datagen.flow(x, batch_size=1):
            augmented_image =
tf.keras.preprocessing.image.array_to_img(batch[0])
            augmented_image.save(os.path.join(class_augmented_dir,
f'augmented_{i}_{file}'))
            i += 1
            if i >= AUGMENTATION_FACTOR:
                break

"Evaluation of The Augmentation Results"
# Checking PSNR
from skimage.metrics import peak_signal_noise_ratio as psnr
from skimage import io, color
import shutil

# Dir inisialitation
original_images_dir = "/content/Dataset_split/train"
augmented_images_dir = "/content/Dataset_split/train_augmented"
output_dir = "/content/resoluted_dataset"  # output dir to save imgs
with psnr > 11,99
os.makedirs(output_dir, exist_ok=True)

# Looping each subdir classes
class_names = os.listdir(augmented_images_dir)  # Use
train_augmented dir
for class_name in class_names:
    class_original_dir = os.path.join(original_images_dir,
class_name)
    class_augmented_dir = os.path.join(augmented_images_dir,
class_name)

    # Variables inisialitation for current class
    total_psnr = 0

    count = 0

    original_images = os.listdir(class_original_dir)

    # Looping imgs in class subdir
    for image_name in original_images:
        original_image_path = os.path.join(class_original_dir,
image_name)

        original_image = io.imread(original_image_path)

        # Making sure the images are in grayscale mode
        original_image = color.rgb2gray(original_image)
```

```
        for augmented_image_name in os.listdir(class_augmented_dir):
            augmented_image_path = os.path.join(class_augmented_dir,
augmented_image_name)
            augmented_image = io.imread(augmented_image_path)

            # Making sure the images are in grayscale mode
            augmented_image = color.rgb2gray(augmented_image)

            # ssim_score = ssim(original_image, augmented_image)
            psnr_score = psnr(original_image, augmented_image)

            # If PSNR score > 11.992841544610027, copy the imgs to
resoluted_dataset
            if psnr_score > 11.992841544610027:
                target_dir = os.path.join(output_dir, class_name)
                os.makedirs(target_dir, exist_ok=True)
                shutil.copy(augmented_image_path, target_dir)

            total_psnr += psnr_score
            count += 1

    # Calculating PSNR means for current class
    average_psnr = total_psnr / count

    # Tampilkan rata-rata PSNR untuk kelas saat ini
    print(f"Class: {class_name}, Average PSNR Score:
{average_psnr}")

train_final = '/content/resoluted_dataset'
# Generator for augmented dataset
train_generator = train_datagen.flow_from_directory(
    train_final,
    target_size=(TARGET_SIZE, TARGET_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
)

val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(TARGET_SIZE, TARGET_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,
)

"Checking Data After Augmented"
# Checking the length of resoluted_dataset dir
# Dir resoluted_dataset
resoluted_dataset_dir = "/content/resoluted_dataset"

# Dictionary inisialitation to save data of each class
class_counts = {}
```

```
# Looping through each subdir
class_names = os.listdir(resoluted_dataset_dir)
total_data = 0
for class_name in class_names:
    class_dir = os.path.join(resoluted_dataset_dir, class_name)
    class_count = len(os.listdir(class_dir))  # Count the data of
current class
    class_counts[class_name] = class_count  # Save the number of
data to dictionary
    total_data += class_count  # Add number of data to total_data

# Showing the number of data in current class
for class_name, count in class_counts.items():
    print(f"Class: {class_name}, Count: {count}")

# Showing number of whole data
print("Total Data: ", total_data)

"Result of Augmentations"
result = sorted(os.listdir(train_final))

plt.figure(figsize=(12, 12))

for i, class_name in enumerate(result):
    class_augmented_dir = os.path.join(train_final, class_name)
    files = os.listdir(class_augmented_dir)
    if files:
        # Mengambil satu gambar hasil augmentasi dari setiap kelas
        image_path = os.path.join(class_augmented_dir, files[0])
        img = tf.keras.preprocessing.image.load_img(image_path)

        plt.subplot(4, 4, i + 1)  # Mengatur tata letak subplot
        plt.imshow(img)
        plt.title(class_name)
        plt.axis("off")

plt.show()

"""MODELLING"""

"""Build CNN"""
# INCEPTION-V3
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.layers import Input

model = tf.keras.models.Sequential([
    InceptionV3(weights="imagenet", include_top=False,
input_tensor=Input(shape=(TARGET_SIZE, TARGET_SIZE, 3))),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.25),
```

```
    tf.keras.layers.Dense(8, activation='softmax')
])
model.layers[0].trainable = False

model.summary()

# VGG-16
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Input

model_vgg = tf.keras.models.Sequential([
    VGG16(weights="imagenet", include_top=False,
input_tensor=Input(shape=(TARGET_SIZE, TARGET_SIZE, 3))),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(8, activation='softmax')
])
model_vgg.layers[0].trainable = False

model_vgg.summary()

# RESNET-50
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Input

model_res = tf.keras.models.Sequential([
    ResNet50(weights="imagenet", include_top=False,
input_tensor=Input(shape=(TARGET_SIZE, TARGET_SIZE, 3))),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(8, activation='softmax')
])
model_res.layers[0].trainable = True

model_res.summary()

# INCEPTION-V3
model.layers

# VGG-16
model_vgg.layers

# RESNET-50
model_res.layers

# INCEPTION-V3
tf.keras.utils.plot_model(model)
```

```
# VGG-16
tf.keras.utils.plot_model(model_vgg)

# RESNET-50
tf.keras.utils.plot_model(model_res)

# INCEPTION-V3
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
lr_reduce = ReduceLROnPlateau(
              monitor='val_loss',
              factor=0.6,
              patience=6,
              verbose=1,
              mode='min',
              min_lr=5e-5
              )
callbacks = EarlyStopping(
              monitor='val_accuracy',
              mode='max',
              patience=10,
              restore_best_weights=True
              )
train_steps = train_generator.samples
val_steps = val_generator.samples
epoch = 100
model.compile(optimizer=tf.optimizers.Adam(learning_rate=0.001),
                loss='categorical_crossentropy',
                metrics=['accuracy']
              )
history = model.fit(
            train_generator,
            steps_per_epoch= train_steps // BATCH_SIZE,
            epochs=epoch,
            validation_data=val_generator,
            validation_steps= val_steps // BATCH_SIZE,
            callbacks= [lr_reduce, callbacks],
            verbose=1,
                    )
# Showing best results
best_val_accuracy = max(history.history['val_accuracy'])
best_val_loss = min(history.history['val_loss'])
best_train_accuracy = max(history.history['accuracy'])
best_train_loss = min(history.history['loss'])
print(f"Best val_acc: {best_val_accuracy:.4f}")
print(f"Best val_loss: {best_val_loss:.4f}")
print(f"Best train_acc: {best_train_accuracy:.4f}")
print(f"Best train_loss: {best_train_loss:.4f}")

# VGG-16
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
lr_reduce = ReduceLROnPlateau(
              monitor='val_loss',
```

```
                factor=0.6,
                patience=6,
                verbose=1,
                mode='min',
                min_lr=5e-5
                )
callbacks = EarlyStopping(
                monitor='val_accuracy',
                mode='max',
                patience=10,
                restore_best_weights=True
                )
train_steps = train_generator.samples
val_steps = val_generator.samples
epoch = 100
model_vgg.compile(optimizer=tf.optimizers.Adam(learning_rate=0.001),
                  loss='categorical_crossentropy',
                  metrics=['accuracy']
                )
history = model_vgg.fit(
              train_generator,
              steps_per_epoch= train_steps // BATCH_SIZE,
              epochs=epoch,
              validation_data=val_generator,
              validation_steps= val_steps // BATCH_SIZE,
              callbacks= [lr_reduce, callbacks],
              verbose=1,
                      )
# Showing best results
best_val_accuracy = max(history.history['val_accuracy'])
best_val_loss = min(history.history['val_loss'])
best_train_accuracy = max(history.history['accuracy'])
best_train_loss = min(history.history['loss'])
print(f"Best val_acc: {best_val_accuracy:.4f}")
print(f"Best val_loss: {best_val_loss:.4f}")
print(f"Best train_acc: {best_train_accuracy:.4f}")
print(f"Best train_loss: {best_train_loss:.4f}")

# RESNET-50
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
lr_reduce = ReduceLROnPlateau(
                monitor='val_loss',
                factor=0.6,
                patience=6,
                verbose=1,
                mode='min',
                min_lr=5e-5
                )
callbacks = EarlyStopping(
                monitor='val_accuracy',
                mode='max',
                patience=10,
                restore_best_weights=True
```

```
                )
train_steps = train_generator.samples
val_steps = val_generator.samples
epoch =100
model_res.compile(optimizer=tf.optimizers.Adam(learning_rate=0.001),
                  loss='categorical_crossentropy',
                  metrics=['accuracy']
                )
history = model_res.fit(
            train_generator,
            steps_per_epoch= train_steps // BATCH_SIZE,
            epochs=epoch,
            validation_data=val_generator,
            validation_steps= val_steps // BATCH_SIZE,
            callbacks= [lr_reduce, callbacks],
            verbose=1,
                    )

# Showing best results
best_val_accuracy = max(history.history['val_accuracy'])
best_val_loss = min(history.history['val_loss'])
best_train_accuracy = max(history.history['accuracy'])
best_train_loss = min(history.history['loss'])
print(f"Best val_acc: {best_val_accuracy:.4f}")
print(f"Best val_loss: {best_val_loss:.4f}")
print(f"Best train_acc: {best_train_accuracy:.4f}")
print(f"Best train_loss: {best_train_loss:.4f}")


"""""EVALUATION"""""

"""Classification Report"""
# INCEPTION-V3
from sklearn.metrics import classification_report, confusion_matrix
Y_pred = model.predict(val_generator, val_steps // BATCH_SIZE+1)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cf_mtx = confusion_matrix(val_generator.classes, y_pred)
print('Classification Report')
target_names = list(train_generator.class_indices.keys())
print(classification_report(val_generator.classes, y_pred,
target_names=target_names))

# VGG-16
Y_pred = model_vgg.predict(val_generator, val_steps // BATCH_SIZE+1)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cf_mtx = confusion_matrix(val_generator.classes, y_pred)
print('Classification Report')
target_names = list(train_generator.class_indices.keys())
print(classification_report(val_generator.classes, y_pred,
target_names=target_names))
```

```
# RESNET50
Y_pred = model_res.predict(val_generator, val_steps // BATCH_SIZE+1)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cf_mtx = confusion_matrix(val_generator.classes, y_pred)
print('Classification Report')
target_names = list(train_generator.class_indices.keys())
print(classification_report(val_generator.classes, y_pred,
target_names=target_names))


"""Plot False Detection"""
# INCEPTION-V3
# Number of classes with errors to display
classes_with_errors_num = 5
# Sizing
img_ns = (150, 150)  # Define image size here
# Plotting
plt.figure(figsize=(20, 4))
class_indices_with_errors = []  # List to store class indices with
errors
# Find all incorrect indices
incorrect_indices = np.where(val_generator.classes != y_pred)[0]
for class_ind in range(len(target_names)):
    # Filter indeks kesalahan untuk kelas tertentu
    incorrect_indices_for_class = np.intersect1d(
        np.where(val_generator.classes == class_ind),
        incorrect_indices
    )
    if len(incorrect_indices_for_class) > 0:
        class_indices_with_errors.append(class_ind)
for i in range(min(classes_with_errors_num,
len(class_indices_with_errors))):
    class_ind = class_indices_with_errors[i]

    # Filter indeks kesalahan untuk kelas tertentu
    incorrect_indices_for_class = np.intersect1d(
        np.where(val_generator.classes == class_ind),
        incorrect_indices
    )
    index = incorrect_indices_for_class[0]  # Select the first error
for this class
    true_class = val_generator.classes[index]
    predicted_class = y_pred[index]
    # Loading incorrect prediction images using PIL
    img_path = val_generator.filepaths[index]
    img = Image.open(img_path)
    # Sizing
    img = img.resize(img_ns)

    # Subplotting
    plt.subplot(1, classes_with_errors_num, i + 1)
    plt.imshow(img)
```

```
    plt.title(f"True: {target_names[true_class]}\nPredicted:
{target_names[predicted_class]}", fontsize=10)
    plt.axis('off')
# Showing results
plt.show()

# VGG-16
# Number of classes with errors to display
classes_with_errors_num = 5
# Sizing
img_ns = (150, 150)  # Define image size here
# Plotting
plt.figure(figsize=(20, 4))
class_indices_with_errors = []  # List to store class indices with
errors
# Find all incorrect indices
incorrect_indices = np.where(val_generator.classes != y_pred)[0]
for class_ind in range(len(target_names)):
    # Filter indeks kesalahan untuk kelas tertentu
    incorrect_indices_for_class = np.intersect1d(
        np.where(val_generator.classes == class_ind),
        incorrect_indices
    )
    if len(incorrect_indices_for_class) > 0:
        class_indices_with_errors.append(class_ind)
for i in range(min(classes_with_errors_num,
len(class_indices_with_errors))):
    class_ind = class_indices_with_errors[i]
    # Filter indeks kesalahan untuk kelas tertentu
    incorrect_indices_for_class = np.intersect1d(
        np.where(val_generator.classes == class_ind),
        incorrect_indices
    )
    index = incorrect_indices_for_class[0]  # Select the first error
for this class
    true_class = val_generator.classes[index]
    predicted_class = y_pred[index]
    # Loading incorrect prediction images using PIL
    img_path = val_generator.filepaths[index]
    img = Image.open(img_path)
    # Sizing
    img = img.resize(img_ns)
    # Subplotting
    plt.subplot(1, classes_with_errors_num, i + 1)
    plt.imshow(img)
    plt.title(f"True: {target_names[true_class]}\nPredicted:
{target_names[predicted_class]}", fontsize=10)
    plt.axis('off')
# Showing results
plt.show()

# RESNET-50
# Number of classes with errors to display
```

```
classes_with_errors_num = 5
# Sizing
img_ns = (150, 150)  # Define image size here
# Plotting
plt.figure(figsize=(20, 4))
class_indices_with_errors = []  # List to store class indices with
errors
# Find all incorrect indices
incorrect_indices = np.where(val_generator.classes != y_pred)[0]
for class_ind in range(len(target_names)):
    # Filter indeks kesalahan untuk kelas tertentu
    incorrect_indices_for_class = np.intersect1d(
        np.where(val_generator.classes == class_ind),
        incorrect_indices
    )
    if len(incorrect_indices_for_class) > 0:
        class_indices_with_errors.append(class_ind)
for i in range(min(classes_with_errors_num,
len(class_indices_with_errors))):
    class_ind = class_indices_with_errors[i]
    # Filter indeks kesalahan untuk kelas tertentu
    incorrect_indices_for_class = np.intersect1d(
        np.where(val_generator.classes == class_ind),
        incorrect_indices
    )
    index = incorrect_indices_for_class[0]  # Select the first error
for this class
    true_class = val_generator.classes[index]
    predicted_class = y_pred[index]
    # Loading incorrect prediction images using PIL
    img_path = val_generator.filepaths[index]
    img = Image.open(img_path)
    # Sizing
    img = img.resize(img_ns)
    # Subplotting
    plt.subplot(1, classes_with_errors_num, i + 1)
    plt.imshow(img)
    plt.title(f"True: {target_names[true_class]}\nPredicted:
{target_names[predicted_class]}", fontsize=10)
    plt.axis('off')
# Showing results
plt.show()


"""Plot Confusion Matrix"""
# INCEPTION-V3
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Creating DataFrame and Heatmap
cm_df = pd.DataFrame(cf_mtx, index=target_names,
columns=target_names)
plt.figure(figsize=(10, 15))
```

```
# Menggunakan orientasi sumbu
sns.heatmap(cm_df, annot=True, cmap='CMRmap', fmt='d', cbar=False,
square=True,
            xticklabels=target_names, yticklabels=target_names)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()

# VGG-16
# Creating DataFrame and Heatmap
cm_df = pd.DataFrame(cf_mtx, index=target_names,
columns=target_names)
plt.figure(figsize=(10, 15))
# Using axis orientation
sns.heatmap(cm_df, annot=True, cmap='CMRmap', fmt='d', cbar=False,
square=True,
            xticklabels=target_names, yticklabels=target_names)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()

# RESNET-50
# Creating DataFrame and Heatmap
cm_df = pd.DataFrame(cf_mtx, index=target_names,
columns=target_names)
plt.figure(figsize=(10, 15))  # Ubah ukuran gambar agar lebih tinggi
daripada lebarnya
# Using axis orientation
sns.heatmap(cm_df, annot=True, cmap='CMRmap', fmt='d', cbar=False,
square=True,
            xticklabels=target_names, yticklabels=target_names)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()


"""Plot Accuracy & Loss"""
# INCEPTION-V3
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.title('Accuracy')

plt.subplot(1,2,2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.grid()
plt.title('Loss')
plt.xlabel('epoch')
plt.show()

# VGG-16
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.title('Accuracy')

plt.subplot(1,2,2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.grid()
plt.title('Loss')
plt.xlabel('epoch')
plt.show()

# RESNET-50
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.title('Accuracy')

plt.subplot(1,2,2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.grid()
plt.title('Loss')
plt.xlabel('epoch')
plt.show()


"""SAVE MODEL"""
""" Save to .h5 file"""
# INCEPTION-V3
model.save('/content/model_Inceptionv3.h5')
# VGG-16
model_vgg.save('/content/model_VGG16.h5')
# RESNET-50
model.save('/content/model_ResNet50.h5')

"""Save to .tflite file"""
# INCEPTION-V3
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with tf.io.gfile.GFile('/content/model_Inceptionv3.tflite', 'wb') as
f:
  f.write(tflite_model)

# VGG-16
converter = tf.lite.TFLiteConverter.from_keras_model(model_vgg)
tflite_model = converter.convert()
with tf.io.gfile.GFile('/content/model_VGG16.tflite', 'wb') as f:
  f.write(tflite_model)

# RESNET-50
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with tf.io.gfile.GFile('/content/model_ResNet50.tflite', 'wb') as f:
  f.write(tflite_model)
```

Lampiran 6. Kode Program *Rest API*

```
import io
import tensorflow as tf
from flask import Flask, jsonify, request
from tensorflow.keras.preprocessing.image import load_img,
img_to_array


model = tf.keras.models.load_model('model.h5')
classes = sorted([ "full_thickness_burn", "luka_memar",
"luka_tusuk", "luka_sayat", "luka_robek",
          "superficial_dermal_burn", "partial_thickness_burn",
"luka_lecet"])


app = Flask(__name__)

def predict_image(img):
    img = load_img(io.BytesIO(img), target_size=(150, 150))
    img = img_to_array(img)
    img = img.reshape(1, 150, 150, 3)
    img = img.astype('float32')
    img = img / 255.0
    pred = model.predict(img)
    return classes[pred.argmax()]

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        img = request.files['image'].read()
        class_name = predict_image(img)
        return jsonify({'result': class_name})

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Lampiran 7. Kode Program Pembangunan *Web*

```
import * as React from "react";

import Grid from "@mui/material/Grid";
import Box from "@mui/material/Box";
import Container from "@mui/material/Container";
import pc from "../assets/images/pc.png";
import Typography from "@mui/material/Typography";
import Button from "@mui/material/Button";
import List from '@mui/material/List';
import ListItem from '@mui/material/ListItem';
import ListItemText from '@mui/material/ListItemText';
import { useNavigate } from "react-router-dom";

const LandingCom = () => {
  const navigate = useNavigate();
  return (
    <Container maxWidth="lg">
      <Box sx={{ flexGrow: 1, mt: 5 }}>
        <Grid container spacing={2} alignItems="center">
          <Grid item xs={12} sm={6}>
            <Box
              component="img"
              sx={{
                height: 500,
                width: "100%",
                maxHeight: { xs: 300, sm: 350, md: 500 },
              }}
              alt=""
              src={pc}
            />
          </Grid>
          <Grid item xs={12} sm={6}>
            <Box textAlign="center">
              <Typography variant="h4" align="center">
                Human Skin Wounds Identification
              </Typography>
              <List>
                <ListItem>
                  <ListItemText primary="List of wounds" />
                </ListItem>
                <ListItem>
                  <ListItemText primary="1. Abrasions" />
                </ListItem>
                <ListItem>
                  <ListItemText primary="2. Lacerations" />
                </ListItem>
                <ListItem>
                  <ListItemText primary="3. Cuts" />
                </ListItem>
                <ListItem>
                  <ListItemText primary="4. Bruises" />
                </ListItem>
                <ListItem>
```

```
                  <ListItemText primary="5. Stab Wounds" />
                </ListItem>
                <ListItem>
                  <ListItemText primary="6. Superficial-thickness
burns" />
                </ListItem>
                <ListItem>
                  <ListItemText primary="7. Partial-thickness
burns" />
                </ListItem>
                <ListItem>
                  <ListItemText primary="8. Deep/full-thickness
burns" />
                </ListItem>
              </List>
              <Button
                onClick={() => navigate("/predict")}
                sx={{ mt: 3 }}
                variant="contained"
              >
                Predict
              </Button>
            </Box>
          </Grid>
        </Grid>
      </Box>
    </Container>
  );
};

export default LandingCom;
```

```
import LandingCom from "../components/LandingCom";

function Home() {
  return (
    <>
        <LandingCom></LandingCom>
    </>
  );
}

export default Home;
```

```
import { Container } from "@mui/material";
import { useState } from "react";
import DisplayCom from "./DisplayCom";
import DragDropCom from "./DragDropCom";

const PredictCom = () => {
    const [imageUrl, setImage] = useState("")
    const [file, setFile] = useState("")
    const imageHandler = (data) => {
        setFile(data)
        setImage(URL.createObjectURL(data[0]))
    }


  return (
    <Container maxWidth="sm" sx={{ marginY: 5 }}>
        <DragDropCom sendFile={imageHandler}></DragDropCom>
        {!imageUrl && !file ? "" : <DisplayCom dataImage={{imageUrl,
file}}></DisplayCom>}
    </Container>
  );
};

export default PredictCom;
```

```
import PredictCom from "../components/PredictCom";

function Predict() {
  return (
    <>
        <PredictCom></PredictCom>
    </>
  );
}

export default Predict;
```

```
import Card from "@mui/material/Card";
import CardActions from "@mui/material/CardActions";
import CardContent from "@mui/material/CardContent";
import CardMedia from "@mui/material/CardMedia";
import { List, ListItem, ListItemText } from "@mui/material";
import Typography from "@mui/material/Typography";
import Grid from "@mui/material/Grid";
import { useLocation } from "react-router-dom";
import MOCK from "./MOCK";

const DisplayResult = () => {
  const { state } = useLocation();
  const cleanedLabel = state.label.replace(/[_#\\W]+/g, " ");
  const conditionData = MOCK.find((item) => item.label ===
state.label);
  const instructions = conditionData ? conditionData.instructions :
[];
  return (
    <>
      <Grid container spacing={0}>
        <Typography variant="h5" sx={{ marginTop: 1 }}>
          Result: {cleanedLabel}
        </Typography>
        <Grid item xs={12}>
          <Card sx={{ maxWidth: "100%", marginTop: 2 }}>
            <CardMedia
              sx={{ height: 300 }}
              image={state.imageUrl}
              title={cleanedLabel}
            />
          </Card>
        </Grid>

        <Grid item xs={12}>
          <Card sx={{ maxWidth: "100%", marginTop: 5 }}>
            <CardContent sx={{ padding: 3 }}>
              <Typography variant="h6">
                Pertolongan pertama untuk {cleanedLabel}
              </Typography>
              <List dense={true}>
                {instructions.map((instruction, index) => (
                  <ListItem>
                    <ListItemText primary={`${index+1}. $
{instruction}`} />
                  </ListItem>
                ))}
              </List>
            </CardContent>
          </Card>
        </Grid>
      </Grid>
    </>
  );
```

```
};

export default DisplayResult;
```

```
import { Container } from "@mui/material";
import DisplayResult from "../components/DisplayResult";

function Result() {
  return (
    <>
      <Container maxWidth="md" sx={{ marginY: 5 }}>
        <DisplayResult></DisplayResult>
      </Container>
    </>
  );
}

export default Result;
```

```
// MOCK.js
const MOCK = [
    {
      label: "luka_sayat",
      instructions: [
        "Mencuci tangan dengan air bersih dan sabun sebelum
membersihkan luka",
        "Mencuci luka dengan air bersih mengalir.",
        "Tekanlah luka dengan kain bersih atau kasa steril",
        "Posisikan bagian tubuh yang terluka lebih tinggi daripada
dada untuk mengontrol pendarahan dan pembengkakan",
        "Jika lukanya cukup besar, tutup dengan kasa steril dan
perban. Sedangkan untuk luka yang kecil dapat dibiarkan terbuka
hingga sembuh dengan sendirinya",
        "Oleskan salep antibiotik pada luka sayatan yang dangkal
untuk mempercepat penyembuhan",
        "Apabila terasa nyeri, gunakan obat pereda nyeri seperti
paracetamol. Hindari mengonsumsi aspirin untuk meredakan nyeri
karena berisiko menimbulkan pendarahan",
        "Jika ada pembengkakan atau memar pada daerah luka, kompres
dengan es batu yang dibungkus kain. Hindari menempelkan es batu
langsung pada luka.",
        "Jaga luka tetap kering dan bersih selama 5-7 hari",
        "Jangan menggaruk atau mengelupas bekas luka atau koreng
yang terbentuk di atas luka.",
      ],
    },
    {
      label: "luka_lecet",
      instructions: [
        "Mencucui tangan dengan air bersih dan sabun sebelum
membersihkan luka",
        "Beri tekanan lembut pada daerah luka dengan kain bersih",
        "Posisikan bagian tubuh yang terluka lebih tinggi",
        "Bersihkan luka dengan air mengalir dan mengambil kotoran
atau benda asing dari luka dengan pinset steril. Jika masih ada
benda yang tertancap, segera pergi ke layanan kesehatan terdetak
agar luka sepenuhnya bersih ",
        "Oleskan salep antibiotik pada luka",
        "Tidak disarankan menggunakan cairan hidrogen peroksida,
obat merah, atau larutan antiseptik yang mengandung iodine, karena
dapat menimbulkan iritasi pada luka",
        "Tutup luka dengan perban untuk menghindari infeksi karena
bakteri"
      ]
    },
    {
      label: "luka_robek",
      instructions: [
        "Jauhi benda tajam dan amati bagian luka",
        "Tekanlah luka dengan kain bersih atau kasa steril",
        "Posisikan bagian tubuh yang terluka lebih tinggi daripada
dada untuk mengontrol pendarahan dan pembengkakan",
```

```
        "Apabila perdarahan masih sulit dihentikan, cobalah menekan
luka robek dengan menekuk siku tangan atau kaki jika memang luka
robek berada di bagian tangan atau kaki",
        "Jika pendarahan yang terjadi sangat deras, selama
menghentikan pendarahan, segera hubungi nomor darurat",
        "Setelah berhasil menghentikan pendarahan, bersihkan luka
dan area sekitarnya menggunakan air bersih",
        "Apabila kedalaman luka > 1,2cm dan pendarahan tidak
berhenti setelah 10 menit, segera bawa ke layanan kesehatan untuk
dijahit",
        "Tutup luka dengan perban untuk menghindari infeksi karena
bakteri",
        "Jika bengkak & nyeri, kompress dengan es batu "
      ]
    },
    {
      label: 'superficial_dermal_burn',
      instructions: [
        "Rendam luka dalam air dingin selama lima menit atau lebih",
        "Jika terasa sangat nyeri, ambil ibuprofen atau
acetaminophen untuk menghilangkannya",
        "Tutup bagian luka dengan kasa longgar untuk menghindari
infeksi",
        "Perhatian: Dalam 7-10 hari, seluruh luka bakar akan sembuh
tanpa jaringan parut"
      ]
    },
    {
      label: 'partial_thickness_burn',
      instructions: [
        "Jika lepuhnya parah, temui dokter sesegera mungkin, karena
pencangkokan kulit mungkin diperlukan",
        "Segera cari pertolongan medis jika luka bakar mengenai
wajah, tangan, atau kaki",
        "Alirkan luka dengan air dingin selama lima belas menit atau
lebih",
        "Ambil ibuprofen atau acetaminophen untuk menghilangkan rasa
sakit",
        "Gunakan salep antibiotik dan oleskan pada luka bakar",
        "Tutup dan bungkus bagian luka dengan kasa longgar untuk
menghindari infeksi",
        "Jangan memecahkan gelembung lepuhan kulit, jika pecah,
kembali bersihkan luka dengan air dingin dan gunakan salep
antibiotik",
        "Perhatian: Sebagian besar sembuh tanpa jaringan parut dari
dua hingga tiga minggu, tetapi beberapa membutuhkan waktu lebih lama
dari tiga minggu"
      ]
    },
    {
      label: 'full_thickness_burn',
      instructions: [
        'Jika tidak ada rasa sakit yang dirasakan, bisa terjadi
```

```
kerusakan saraf',
        'Pastikan tidak ada pakaian yang menempel pada luka bakar',
        'Angkat bagian tubuh yang terluka lebih tinggi dari posisi
jantung',
        'Jangan sentuh luka bakar dan segera hubungi nomor darurat
(112) atau pihak rumah sakit terdekat',
        'Perhatian: Tidak ada batas waktu untuk penyembuhan, dan
akan ada jaringan parut dan kontraktur yang parah'
      ]
    },
    {
      label: 'luka_tusuk',
      instructions: [
        "Mencuci tangan dengan air bersih dan sabun sebelum
membersihkan luka",
        "Tekanlah luka dengan kain bersih atau kasa steril",
        "Jika terjadi di dada, harus segera ditutup dengan tangan
atau dengan balutan yang tidak memungkinkan udara mengalir",
        "Mencuci luka tusuk dengan air hangat dan sekitaran luka
dengan sabun yang lembut",
        "Jika luka tidak terlalu parah, berikan salep antiseptik",
        "Balut atau tutup luka dengan perban",
        "Jika parah, segera hubungi nomor darurat atau langsung ke
layanan kesehatan terdekat",
        "Jika tertusuk benda berkarat, segera ke layanan keseheatan
agar diberikan vaksin tetanus untuk pencegahan infeksi"
      ]
    },
    {
      label: 'luka_memar',
      instructions: [
        'Oleskan es segera setelah cedera untuk mengurangi aliran
darah disekitar area tersebut selama 10-20 menit',
        'Cara 1: Kompres dengan kantong es yang dibungkus dengan
kain atau handuk',
        'Cara 2: Bungkus area yang memar dengan perban elastis.
Tindakan ini bertujuan untuk memeras jaringan dan membantu mencegah
pembuluh darah bocor',
        'Cara 3: Tinggikan area yang memar sehingga berada di atas
jantung. Ini membantu menghilangkan rasa sakit dan mengalirkan
cairan dari area yang memar',
        'Cara 4: Oleskan krim atau salep vitamin K dengan lembut ke
memar setidaknya dua kali sehari'
      ]
    },
  ];

  export default MOCK;
```