UNIVERSITAS GUNADARMA
FAKULTAS TEKNOLOGI INDUSTRI

PENULISAN ILMIAH

IMPLEMENTASI CONVOLUTIONAL NEURAL NETWORK BERBASIS TENSORFLOW PADA KLASIFIKASI JENIS JERAWAT MENGGUNAKAN ARSITEKTUR MOBILENETV2

Nama	**: Sinatrio Bimo Wahyudi**
NPM	**: 56418732**
Jurusan	**: Teknik Informatika**
Pembimbing	**: Dr. Yulia Chalri, SKom., MMSI**

Diajukan Guna Melengkapi Sebagian Syarat Dalam Mencapai
Gelar Setara Sarjana Muda

Jakarta
2021

PERNYATAAN ORIGINALITAS DAN PUBLIKASI

Saya yang bertanda tangan dibawah ini,

Nama	:	Sinatrio Bimo Wahyudi
NPM	:	56418732
Judul Penulisan Ilmiah	:	Implementasi Convolutional Neural Network Berbasis Tensorflow Pada Klasifikasi Jenis Jerawat Menggunakan Arsitektur MobileNetV2
Tanggal Sidang	:	
Tanggal Lulus	:	

Menyatakan bahwa tulisan ini adalah merupakan hasil karya saya sendiri dan dapat dipublikasikan sepenuhnya oleh Universitas Gunadarma. Segala kutipan dalam bentuk apa pun telah mengikuti kaidah, etika yang berlaku. Mengenai isi dan tulisan adalah merupakan tanggung jawab penulis, bukan Universitas Gunadarma.

Demikian pernyataan ini dibuat dengan sebenarnya dan dengan penuh kesadaran.

Depok, 29 Juli 2021

(Sinatrio Bimo Wahyudi)

LEMBAR PENGESAHAN

Judul PI	:	Implementasi Convolutional Neural Network Berbasis Tensorflow Pada Klasifikasi Jenis Jerawat Menggunakan Arsitektur MobileNetV2
Nama	:	Sinatrio Bimo Wahyudi
NPM	:	56418732
Tanggal Sidang	:	
Tanggal Lulus	:	

Menyetujui,

Pembimbing	Koordinator PI
(Dr. Yulia Chalri, SKom., MMSI)	(Dr. Achmad Fahrurozi S.Si, M.Si)

Ketua Jurusan

(Dr. Lintang Yuniar Banowosari, S.Kom., M.Sc)

ABSTRAK

Sinatrio Bimo Wahyudi.56418732

IMPLEMENTASI CONVOLUTIONAL NEURAL NETWORK BERBASIS TENSORFLOW PADA KLASIFIKASI JENIS JERAWAT MENGGUNAKAN ARSITEKTUR MOBILENETV2

PI, Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Gunadarma, 2021

Kata Kunci : Machine Learning, Computer Vision, Convolutional Neural Network, Tensorflow, MobileNetV2, Image Classification, Image Augmentation, Image Processing, API.

(xcix + 99 + lampiran)

Jerawat adalah suatu keadaan dimana pori-pori kulit tersumbat sehingga menimbulkan kantung nanah yang meradang. Penyebab timbulnya jerawat disebabkan oleh perubahan hormonal pada masa pubertas yang merangsang kelenjar minyak atau efek dari siklus menstruasi, hingga stress. Penderita jerawat seringkali salah dalam upaya penyembuhan jerawat yang mengakibatkan jerawat mengalami inflamasi, hal tersebut disebabkan karena variasi jerawat yang berbeda mengindikasikan cara penanganan yang juga berbeda. Bahkan seorang ahli dermatologis tidak jarang mengalami kesulitan pada saat identifikasi jerawat karena beberapa jenis jerawat memiliki kemiripan yang tidak dapat ditentukan hanya melalui mata manusia, Oleh karena itu diperlukan adanya pendekatan teknologi agar dapat melakukan prediksi jerawat dengan mudah. Salah satunya dengan deep learning. Convolutional Neural Network atau CNN adalah salah satu algoritma deep learning yang mampu melakukan klasifikasi terhadap sebuah citra layaknya pengelihatan manusia. CNN dapat dimanfaatkan untuk mengklasifikasi jenis jerawat melalui sebuah model. Model akan dilatih terlebih dahulu di dalam sebuah *data pipeline* menggunakan MobileNetV2. MobileNetV2 adalah arsitektur model yang mampu mengatasi kebutuhan sumber daya komputasi yang berlebih dengan tetap menjaga performa dan kecepatan pelatihan pada berbagai platform. Sesuai dengan namanya yakni mobile, para peneliti membuat arsitektur CNN yang dapat digunakan untuk ponsel. Berdasarkan evaluasi model, diperoleh akurasi sebesar 90.6%. Sebuah aplikasi berhasil dibuat untuk memudahkan melakukan klasifikasi. Klasifikasi jenis jerawat dapat dilakukan dengan cara mengunggah gambar yang telah dikonversi dari format gambar menjadi format base64 guna meringankan beban kerja model dan mempercepat pelatihan data.

Daftar Pustaka (2012-2020)

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kehadirat Allah S.W.T yang Maha Kuasa yang telah memberikan berkat, anugerah, dan karunia yang melimpah sehingga penulis dapat menyelesaikan Penulisan Ilmiah ini pada waktu yang telah ditentukan.

Penulisan Ilmiah ini disusun guna melengkapi sebagian syarat untuk memperoleh gelar Sarjana Teknik Universitas Gunadarma. Penulis menyadari bahwa penulisan ilmiah ini masih jauh dari kesempurnaan, untuk itu diharapkan dengan segala kerendahan hati kiranya dapat diberikan saran dan kritik yang bersifat membangun kepada penulis guna tercapainya karya yang lebih baik di masa depan nanti. Adapun judul Penulisan Ilmiah ini adalah **Implementasi Convolutional Neural Network Berbasis Tensorflow Pada Klasifikasi Jenis Jerawat Menggunakan Arsitektur MobileNetV2**.

Walaupun banyak kesulitan yang penulis harus hadapi ketika menyusun Penulisan Ilmiah ini, namun berkat bantuan dan dorongan dari berbagai pihak, pada akhirnya Penulisan Ilmiah ini dapat diselesaikan dengan baik. Di dalam menyusun laporan penulisan ilmiah ini, banyak sekali pihak-pihak yang membantu baik secara materil ataupun spiritual serta saran-saran dari berbagai macam pihak, dan juga yang telah banyak memberikan pengarahan serta koreksi yang sangat bermanfaat dalam membimbing penulis sampai dengan terselesainya penulisan ilmiah ini.

Pada kesempatan ini juga, penulis tidak lupa untuk mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Ibu Prof. Dr. E. S. Margianti, SE., MM., selaku Rektor Universitas Gunadarma.
2. Bapak Prof. Dr. Ing. Adang Suhendra, SSi, SKom, MSc selaku Dekan

Fakultas Teknologi Industri Universitas Gunadarma.

3. Ibu Dr. Lintang Yuniar Banowosari, S.Kom., M.Sc., selaku Ketua Jurusan Teknik Informatika Universitas Gunadarma.
4. Bapak. Dr. Achmad Fahrurozi S.Si, M.Si selaku koordinator Penulisan Ilmiah.
5. Ibu. Dr. Yulia Chalri, SKom., MMSI, selaku dosen pembimbing yang telah meluangkan waktu dan pikirannya dalam bentuk bimbingan kepada penulis.
6. Bapak Hugeng Tunggul Wahyudi (Bapak), Ibu Diah Ernawati (Ibu) dan Sinatrio Seto Wahyudi (Adik), serta saudara-saudara penulis yang telah memberikan dorongan, motivasi, doa dan menjadi tokoh inspirasi penulis sehingga dapat menyelesaikan Penulisan Ilmiah ini.
7. Semua teman-teman dan rekan-rekan seperjuangan Penulisan Ilmiah yang telah memberi dukungan dan semangat serta turut membantu dalam penulisan ini atas segala inspirasi dan pendapat yang sangat bermanfaat bagi penulis.

Semoga Allah SWT melimpahkan berkat dan karunia-Nya kepada semua pihak atas segala bantuan dan bimbingannya.

Dengan segala kerendahan hati, penulis sangat menyadari bahwa masih banyak kekurangan dalam Penulisan Ilmiah ini. Oleh karena itu penulis mohon maaf atas kekurangannya. Akhir kata, penulis mengharapkan semoga hasil dari penulisan ilmiah ini dapat bermanfaat bagi penulis khususnya dan pembaca pada umumnya.

Depok, 29 Juli 2021

Sinatrio Bimo Wahyudi

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Iklim merupakan faktor utama dari persebaran variasi suatu penyakit di setiap negara. Sebagai negara beriklim tropis, Indonesia memiliki tingkat kelembapan ekstrim yang memengaruhi rentang, perilaku, siklus biologis, sejarah hidup patogen dan vektor spesies penyakit. Tingginya curah hujan dan intensitas sinar ultraviolet juga menyebabkan bakteri berkembang biak dengan subur. Sehubungan dengan aktivitas iklim tropis tersebut, penyakit kulit dan subkutan (lemak yang terletak dibawah kulit) menjadi isu kesehatan yang umum menjangkiti penduduk di Indonesia. Permasalahan tersebut juga didukung dengan publikasi Pusat Data dan Informasi Kementrian Kesehatan Republik Indonesia tahun 2010 yang menunjukkan bahwa penyakit kulit menempati peringkat ketiga dari sepuluh kasus penyakit terbanyak pada pasien rawat jalan. Menurut data Departemen Kesehatan RI prevalensi penyakit kulit diseluruh Indonesia di tahun 2012 adalah 8,46% kemudian meningkat ditahun 2013 sebesar 9% [Badan Penelitian dan Pengembangan Kesehatan Republik Indonesia 2013]. Penyakit kulit adalah penyakit yang erat kaitannya dengan lingkungan luar. Material fisika, kimiawi, dan biologik di alam bebas yang tidak terlihat oleh mata seringkali merangsang proses infeksius dari entitas mirkoorganisme pada manusia yang tidak menerapkan gaya hidup bersih. Sanitasi dan higienitas yang buruk menyebabkan setidaknya 120 juta kasus penyakit setiap tahunnya, termasuk penyakit kulit [Aide Medicale Internationale, 2009].

Meninjau permasalahan serupa melalui kacamata internasional, menurut jurnal dari *Kings College London and the International Foundation for Dermatology*, penyakit kulit diperkirakan menyerang lebih dari 900.000.000 orang di seluruh dunia setiap tahun. Kemudian juga disebutkan bahwa sekitar 20-40% konsultasi baru di tingkat perawatan primer dimotivasi oleh masalah kulit. Kondisi kulit berkontribusi 1,79% terhadap beban penyakit global yang diukur

dengan DALY (Disability-Adjusted Life Year) atau tingkat kehidupan berdasarkan kecatatan. Dari 306 penyakit dan cedera pada tahun 2013, sekitar 0,29% disebabkan oleh jerawat vulgaris [Kings College London and the International Foundation for Dermatology]

Jerawat adalah suatu keadaan dimana pori-pori kulit tersumbat sehingga menimbulkan kantung nanah yang meradang, Penyebab timbulnya jerawat datang dari berbagai faktor. Salah satunya disebabkan oleh perubahan hormonal pada masa pubertas yang merangsang kelenjar minyak atau efek dari siklus menstruasi, kehamilan, pemakaian pil KB, hingga stress. Adapun penyebab lain dari timbulnya jerawat. Menurut jurnal *Acne Vaccines: Therapeutic Option for the Treatment of Acne Vulgaris* karya Jenny J. Kim, MD, PhD menyebutkan bahwa jerawat disebabkan oleh bakteri Propionibacterium acnes. Studi menunjukkan bahwa tanggapan terhadap bakteri Propionibacterium acnes oleh imunitas inang memainkan peran penting dalam patogenesisnya [Journal of Investigative Dermatology Volume 128, Issue 10, October 2008]. Jerawat paling sering terjadi pada usia 10-13 tahun. Anak perempuan memiliki onset yang lebih awal yang dengan mudah berkontribusi pada timbulnya pubertas pada anak perempuan daripada anak laki-laki. Itu keparahan penyakit lebih banyak pada anak laki-laki selama masa remaja akhir. Jerawat sebagian besar berkembang di area kulit yang memiliki kelenjar minyak yang melimpah, seperti wajah, dada, punggung dan bahu.

Seiring moderenisasi zaman, problematika kesehatan semacam ini dapat diasosiasikan dengan pemanfaatan sumber daya teknologi sebagai alat untuk mendefinisikan solusi. Upaya tersebut juga tercantum dalam Rencana Strategis Kesehatan Republik Indonesia (Renstra Kemenkes) 2020-2024 yang menyebutkan bahwa pemanfaatan teknologi tengah diaplikasikan dalam penyelenggaraan pelayanan kesehatan terpadu berbasis IT yang bertujuan untuk meningkatkan Sistem Informasi Kesehatan. Gagasan penerapan SIK dipersiapkan sebagai kiat dalam mengoptimalkan penggunaan inovasi kesehatan digital dan internet untuk perolehan data surveilans *real-time* dan membuat perubahan bertahap dari pelaporan agregat ke pelaporan individu yang merupakan investasi

jangka panjang yang harus dilakukan untuk penguatan pelaporan data rutin [Rencana Strategis Kemenkes 2020-2024].

Tujuan pemerintah dari perencanaan strategis ini adalah penguatan sistem kesehatan di semua level pemerintahan menjadi responsif dan tangguh, dalam upaya mencapai derajat kesehatan masyarakat yang setinggi-tingginya dengan didukung inovasi teknologi. Salah satu inovasi teknologi yang sudah meluas di Indonesia dan dunia adalah *electronic-health* atau E-Health. Pengembangan eHealth sangat menjanjikan untuk meningkatkan akses pasien sebagai konsumen dan penyedia ke informasi kesehatan yang relevan, meningkatkan kualitas perawatan, mengurangi kesalahan perawatan kesehatan, meningkatkan kolaborasi, dan mendorong penerapan perilaku sehat.

Definisi E-health dapat ditemui antara lain sebagaimana diutarakan oleh World Health Organization (WHO), yaitu "the use of information and communication technologies (ICT) for health to, for example, treat patients, pursue research, educate students, track diseases and monitor public health". Sementara dalam KepMenKes Nomor 192/MENKES/SK/VI/2012 disebutkan bahwa E-health adalah pemanfaatan TIK di sektor kesehatan terutama untuk meningkatkan pelayanan kesehatan [Roadmap Rencana Aksi Penguatan Sistem Informasi Kesehatan Indonesia].

E-Health meliputi aplikasi untuk para profesional dan otoritas kesehatan yang lebih baik daripada sistem kesehatan pribadi untuk masyarakat dan pasien. Sebagai contoh adalah health information networks, electronic medical records, telemedicine services, personal wearable and portable communicable systems, health portals, dan banyak teknologi komunikasi dan informasi lain yang bertujuan membantu pencegahan, diagnosa, perawatan, monitoring kesehatan, dan manajemen gaya hidup. E-Service merupakan layanan melalui internet yang biasanya mengacu pada peran teknologi dalam memfasilitasi pelayanan yang membuat mereka lebih dari layanan elektronik [Sundari, 2016].

Dalam perkembangan tren machine learning atau pembelajaran mesin pada sektor industri, E-Health juga ikut memaksimalkan peran machine learning dalam melakukan aktivitas komputasi yang mandiri dan terautomasi. Istilah

machine learning pertama kali didefinisikan oleh Arthur Samuel pada tahun 1959. Menurut Arthur Samuel, machine learning adalah suatu bidang ilmu komputer yang memberikan kemampuan pembelajaran kepada komputer untuk mengetahui sesuatu tanpa menuliskan kode pemrograman secara eksplisit [Arthur Samuel, 1959]. Machine learning merupakan serangkaian teknik yang dapat membantu dalam menangani dan memprediksi data yang sangat besar dengan cara mempresentasikan data-data tersebut dengan algoritma pembelajaran [Danukusumo, 2017].

Machine learning merupakan pendekatan teknologi yang berorientasi terhadap cara kerja indra pada manusia, salah satu sub keilmuan machine learning mampu mereplikasi indra penglihatan manusia yang dimanifestasikan oleh jaringan syaraf konvolusi atau *Convolutional Neural Network* (*CNN)*. CNN terinspirasi oleh korteks mamalia visual sel sederhana dan kompleks. Model ini dapat mengurangi sejumlah parameter bebas dan dapat menangani deformasi gambar input seperti translasi, rotasi dan skala. Berdasarkan penjelasan kelebihan CNN tersebut, dapat diambil kesimpulan bahwa CNN memiliki kemampuan klasifikasi yang diperuntutkan untuk data gambar sehingga pada penelitian ini model CNN akan digunakan sebagai metode klasifikasi jenis jerawat.

Convolutional Neural Network merupakan salah satu jenis algoritma pembelajaran mendalam atau *deep learning* yang dapat menerima input berupa gambar, menentukan aspek atau obyek apa saja dalam sebuah gambar yang bisa digunakan mesin untuk belajar mengenali gambar, dan membedakan antara satu gambar dengan yang lainnya. Arsitektur CNN serupa dengan pola koneksi neuron atau sel saraf dalam otak manusia. CNN terinspirasi dari *visual cortex*, yaitu bagian pada otak yang bertugas untuk memroses informasi dalam bentuk visual. Dengan arsitektur seperti itu, CNN dapat dilatih untuk memahami detail sebuah gambar dengan lebih baik. Dengan begitu, CNN dapat menangkap dependensi spasial dan temporal dalam sebuah gambar setelah kamu memberikan filter yang relevan [Jurnal Sains dan Seni ITS Vol. 5 No. 2].

Berkat kapabilitasnya yang menyerupai indra pengelihatan manusia, dasar-dasar metode CNN secara garis besar dimanfaatkan peneliti guna pengembangan

visi komputer atau lebih dikenal dengan *computer vision*. Computer vision adalah sebuah analisis gambar dan video secara otomatis oleh komputer guna mendapatkan suatu pemahaman mengenai dunia [Kenneth Dawson & Howe, 2014]. Computer vision memiliki kemampuan dalam melatih komputer untuk menafsirkan dan memahami dunia visual. Menggunakan gambar digital dari kamera dan video serta model pembelajaran mendalam, mesin-mesin dapat dengan akurat mengidentifikasi dan mengklasifikasikan objek kemudian bereaksi terhadap apa yang dilihatnya.

Pada penelitian ini, penulis memanfaatkan kemampuan kecerdasan buatan yakni computer vision khususnya menggunakan metode *convolutional neural network* dalam pengembangan model machine learning untuk mengklasifikasikan jenis jerawat. Tantangan dalam pembuatan model pendeteksi jerawat adalah kecilnya jerawat sehingga dibutuhkan algoritma yang tepat untuk pendeteksian melalui ekstraksi fitur dan klasifikasi. Salah satu parameter algoritma yang tepat dapat mengatasi overfitting, kecepatan konvergensi yang tepat dan dapat belajar dari data training dengan tepat.

Penelitian-penelitian terdahulu belum banyak yang melakukan klasifikasi jenis jerawat, melainkan hanya sebatas deteksi eksistensi jerawat pada anggota tubuh dan juga identifikasi jerawat dengan penyakit kulit lainnya, bintik/noda, bekas luka ataupun kulit normal. Ada penelitian terhadap klasifikasi jerawat yang sudah pernah dilakukan, tetapi hasil akurasinya masih belum maksimal sehingga penulis merasa perlunya dilakukan penelitian lain terhadap klasifikasi jenis jerawat dengan metode lain untuk mendapat hasil akurasi yang lebih maksimal. Oleh karena itu, dalam penelitian ini penulis mengajukan metode klasifikasi jerawat dengan menggunakan arsitektur MobileNetV2 dengan judul "Implementasi Convolutional Neural Network Berbasis Tensorflow Pada Klasifikasi Jenis Jerawat Menggunakan Arsitektur MobileNetV2".

Dalam proses penelitian, penulis menggunakan framework Tensorflow sebagai *library* dan ekosistem dalam mengembangkan dan melatih model yang telah menyediakan kumpulan alur kerja sehingga dapat mempercepat proses pengkodean. Tensorflow adalah sistem pembelajaran mesin yang beroperasi di

skala besar dan di lingkungan heterogen. TensorFlow menggunakan grafik aliran data untuk mewakili komputasi, status bersama, dan operasi yang mengubah status tersebut. Tensorflow memetakan simpul grafik aliran data di banyak mesin dalam klaster, dan dalam mesin di beberapa perangkat komputasi, termasuk CPU multicore, GPU *general purpose*, dan ASIC yang dirancang khusus serta dikenal sebagai Unit Pengolahan Tensor (TPUs). Kerangka atau *framework* ini memberikan fleksibilitas kepada pengembang aplikasi. TensorFlow memungkinkan pengembang untuk bereksperimen dengan optimasi baru dan algoritma pelatihan.

TensorFlow mendukung berbagai aplikasi, dengan dukungan yang sangat kuat untuk pelatihan dan inferensi pada jaringan saraf mendalam atau *deep neural network*. Beberapa layanan Google menggunakan TensorFlow dalam produksi, Google telah merilisnya sebagai proyek sumber terbuka atau *open source*, dan telah banyak digunakan untuk penelitian pembelajaran mesin [Google Brain, 2016].

Pemilihan arsitektur MobileNetV2 dinilai relevan karena MobileNetV2 diciptakan oleh tim Google Brain untuk beroperasi secara optimal pada perangkat seluler. Arsitektur ini memberikan hasil akurasi tinggi dengan tetap menjaga stablitas parameter dan operasi matematis serendah mungkin dalam rangka membawa jaringan saraf mendalam atau *deep neural network* menuju sistem perangkat seluler. MobileNetV2 adalah produk peningkatan atas versi pendahulunya yakni MobileNetV1. MobileNetV2 memiliki fitur tambahan yang berfokus pada pengenalan visual mobile termasuk klasifikasi, deteksi objek dan segmentasi semantik. MobileNetV2 dirilis sebagai bagian dari TensorFlow-Slim Image Classification Library [Mark Sandler, 2019].

MobileNetV2 dibangun di atas ide-ide dari MobileNetV1, menggunakan konvolusi yang dapat dipisahkan secara mendalam sebagai blok bangunan yang efisien. Namun, V2 memperkenalkan dua fitur baru pada arsitektur yaitu kemacetan linier antara lapisan dan koneksi pintasan antara kemacetan. Modul ini dapat diimplementasikan secara efisien menggunakan operasi standar dalam kerangka kerja modern apa pun. Selain itu, modul konvolusi ini sangat cocok

untuk desain seluler, karena memungkinkan untuk mengurangi jejak memori yang diperlukan selama inferensi secara signifikan tanpa harus sepenuhnya sampai ke perantara besar tensor. Hal tersebut mengurangi kebutuhan akses memori utama pada seluruh desain perangkat keras tertanam atau *embedded hardware*, yang menyediakan jumlah memori cache yang dikendalikan perangkat lunak yang sangat cepat. [MobileNetV2: Inverted Residuals and Linear Bottlenecks]

Dalam penelitian ini, pertama-tama penulis memberikan gambaran umum tentang kecerdasan buatan dan machine learning hingga bagaimana algoritma dikembangkan. Kedua, penulis memeriksa arus model machine learning yang relevan dengan dermatologis. Terakhir, penulis mengeksplorasi tantangan potensial dan batasan untuk pengembangan masa depan model machine learning pada penulisan ini.

Untuk itu perlu dibuat nya model machine learning yang dapat ditanamkan pada segala platform terutama perangkat seluler, baik berupa sistem maupun aplikasi yang bertujuan untuk mendukung pelayanan kesehatan maupun klinik kecantikan dan membantu dokter spesialis kulit dalam mengidentifikasi jenis jerawat secara efektif dan cepat sehingga penanganan dapat dilaksanakan dengan sebaik-baiknya.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dijelaskan sebelumnya, rumusan masalah dalam penulisan ini adalah bagaimana membuat model machine learning untuk melakukan klasifikasi jenis-jenis jerawat berdasarkan segmentasi warna dan analisa tekstur kulit

1.3 Batasan Masalah

Berdasarkan perumusan masalah diatas, penulis menentukan ruang lingkupnya, yaitu:

1. Pembuatan model machine learning yang dapat memprediksi persentase dan mengklasifikasikan jenis-jenis jerawat berdasarkan

kondisi kulit wajah dan anggota badan melalui file gambar yang diinput.

2. Set data yang digunakan dalam penelitian hanya sejumlah 750 data gambar yang dibagi menjadi 2 kelas jenis jerawat yakni acne comedonal dan acne inflammatory.
3. Pembahasan model machine learning mencakup augmentasi gambar dan penggunaan arsitektur MobileNetV2.
4. Pembuatan backend microframework Flask sebagai medium pemrosesan gambar yang di prediksi oleh model machine learning.
5. Bentuk laporan model berupa nilai akurasi prediksi dan label jenis jerawat.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dari penulisan ilmiah ini adalah:

1. Merancang desain arsitektur jaringan untuk mengenal dan mengklasifikasi jenis-jenis jerawat dengan menggunakan MobileNetV2
2. Mengimplementasikan struktur jaringan klasifikasi jenis jerawat dengan menggunakan transfer learning MobileNetV2
3. Mengukur tingkat akurasi foto klasifikasi jerawat dengan menggunakan framework Tensorflow

1.5 Metode Penelitian

Metode yang digunakan dalam penulisan ini adalah metode Studi Literatur dan SDLC (System Development Lifecyle) Air Terjun (Waterfall).

Metode Studi Literatur adalah tahapan observasi dan pemahaman konsep, teori dan materi pendukung penelitian yang digunakan untuk mencari referensi – referensi yang berhubungan dengan penelitian, pada penelitian ini berkaitan dengan *Machine Learning, Computer Vision, K-Means, Convolutional Neural Network, Tensorflow, MobileNetV2, Image Classification, Image Augmentation,*

Image Processing, dan API. Referensi yang diperoleh dari buku, jurnal, skripsi, internet.

Metode pengembangan perangkat lunak dilakukan dengan metode waterfall. Model SDLC air terjun (waterfall) sering juga disebut model sequential linier [Pressman, 2002]. Model waterfall menyediakan alur hidup perangkat lunak secara sekuensial atau urut dimulai dari analisis, desain, pengkodean, pengujian dan tahap support.

Pada penelitian ini menggunakan metode modern waterfall agar jika suatu saat ada kesalahan pada salah satu tahap dapat dikembalikan ke tahap sebelumnya. Berikut pengertian dari tahap-tahap pada model waterfall menurut [Sommerville, 2015]:

1. Analisa

   Tahap Analisa merupakan tahap yang berfungsi untuk menganalisa konsep dan metode yang digunakan dalam pembangunan model yang akan dibuat dan pengumpulan informasi dalam bentuk literatur tertulis yang diperoleh melalui buku, jurnal, situs web, dan lain-lain, sebagai landasan teori dalam penelitian ini. Pada tahap ini juga dilakukan identifikasi kebutuhan guna mempersiapkan model yang akan dibangun. Semua hal tersebut akan ditetapkan secara rinci dan berfungsi sebagai spesifikasi sistem.

2. Perancangan

   Dalam tahapan ini akan dilakukan perancangan skenario implementasi, skenario preprocessing, dan skenario pemilihan model dan optimizernya, dan skenario pengujian. Selain itu dilakukan pengumpulan data dari dermnet dan google image serta dilakukan pembagian dataset menjadi tiga bagian, yaitu data training, testing, dan validation. Rasio perbandingan yang digunakan dalam split data adalah 60:20:20

3. Implementasi dan Pengujian

   Coding adalah tahap proses implementasi dari perancangan, dalam tahapan ini, hasil persiapan dataset dilanjutkan pada tahap pra proses melalui metode image segmentation dan image augmentation, kemudian pengembangan model dan pelatihan data gambar menggunakan arsitektur

MobileNetV2 serta pengujian Application Programming Interface (API) dari model machine learning yang telah dilatih menggunakan aplikasi postman.

4. Integrasi

   Proses testing atau pengujian dilakukan pada logika internal untuk memastikan semua pernyataan sudah diuji. Dalam tahapan ini, setiap unit program akan diintegrasikan dalam satu pemanggilan API yang dibangun menggunakan microframework Flask dan diuji sebagai satu sistem yang utuh untuk memastikan sistem sudah memenuhi persyaratan yang ada.

1.6 Sistematika Penelitian

Dalam Penulisan Ilmiah ini terbagi menjadi 4 bab yaitu:

1. BAB I menguraikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penulisan, metode penelitian, dan sistematika penulisan.
2. BAB II berisikan penjelasan mengenai teori yang menunjang dalam pembuatan aplikasi. Teori yang digunakan tentang Machine Learning, Tensorflow, MobileNetV2, bahasa pemrograman yang digunakan, dan landasan lain yang mendukung.
3. BAB III menjelaskan secara terperinci mengenai perancangan, implementasi, proses pembuatan dan uji coba dari model machine learning yang dibuat oleh penulis.
4. BAB IV berisi mengenai kesimpulan dari pembahasan masalah dalam pembuatan model machine learning dan saran yang dapat membantu penulis dalam upaya membangun model ini agar lebih baik

BAB 2

LANDASAN TEORI

2.1 Jerawat

Jerawat adalah suatu penyakit yang disebabkan oleh inflamasi kronik dari unit pilosebasea yang ditandai pembentukan komedo, papul, pustul, nodul yang bisa tumbuh di daerah wajah, punggung, dada, lengan, kaki yang menyebabkan terjadinya perubahan citra. Pada kulit yang semula dalam kondisi normal, sering kali terjadi penumpukan kotoran dan sel kulit mati karena kurangnya perawatan dan pemeliharaan, khususnya pada kulit yang memiliki tingkat reproduksi minyak yang tinggi. Akibatnya saluran kandung rambut (folikel) menjadi tersumbat menghasilkan komedo. Sel kulit mati dan kotoran yang menumpuk tersebut, kemudian terkena bakteri acne, maka timbulah jerawat. Jerawat yang tidak diobati akan mengalami pembengkakan (membesar dan berwarna kemerahan) disebut papul. Bila peradangan semakin parah, sel darah putih mulai naik ke permukaan kulit dalam bentuk nanah (pus), jerawat tersebut disebut pustul [Mitsui, 1997].

Jerawat radang terjadi akibat folikel yang ada di dalam dermis mengembang karena berisi lemak padat, kemudian pecah, menyebabkan serbuan sel darah putih ke area folikel sebaseus, sehingga terjadilah reaksi 8 radang. Peradangan akan semakin parah jika kuman dari luar ikut masuk ke dalam jerawat akibat perlakuan yang salah seperti dipijat dengan kuku atau benda lain yang tidak steril. Jerawat radang mempunyai ciri berwarna merah, cepat membesar, berisi nanah dan terasa nyeri. Pustul yang tidak terawat, maka jaringan kolagen akan mengalami kerusakan sampai pada lapisan dermis, sehingga kulit/wajah menjadi bekas luka [Mitsui, 1997]. Jerawat adalah penyakit kulit umum, terutama pada remaja dan dewasa muda. Jerawat mempengaruhi sekitar 85% remaja, tetapi dapat terjadi di sebagian besar kelompok usia dan dapat bertahan hingga dewasa. Prevalensi jerawat pada wanita dewasa adalah sekitar 12%. Tidak ada kematian yang terkait dengan jerawat, tetapi sering ada morbiditas fisik dan psikologis yang

signifikan, seperti jaringan parut permanen, citra diri yang buruk, depresi, dan kecemasan. Biaya langsung penyakit ini diperkirakan melebihi $3 miliar per tahun [Journal of the American Academy of Dermatology].

Hasil penelitian menunjukkan sekitar 90 % dari seluruh remaja mengalami jerawat dalam derajat yang berbeda-beda dan 20 % memerlukan pertolongan dokter, pada umumnya keluhan penderita lebih bersifat estetis, sehingga perlu diperhatikan dampak psikososial pada remaja yang dapat mempengaruhi interaksi sosial, prestasi sekolah dan juga pekerjaan [Soetjingsih, 2010].

Terdapat dua jenis jerawat atau *acne* yang secara umum terjadi di kehidupan masyarakat Indonesia, yakni acne comedonal dan acne inflammatory. Berikut detail penjelasan terkait dua jenis jerawat tersebut:

1) Acne Comedonal

Jenis jerawat comedonal acne merupakan jerawat yang berada dibawah permukaan kulit. Jenis jerawat ini biasa muncul karena adanya pori-pori atau folikel rambut yang tersumbat oleh kotoran, bakteri ataupun sel kulit mati.

Gambar 2.1 Acne Comedonal

Pada gambar 2.1, jenis jerawat ini sering muncul atau dialami pada seseorang yang memiliki jenis kulit berminyak. Jenis jerawat comedonal ini pun ada dua, yaitu whiteheads dan blackheads. Untuk whiteheads sendiri terjadi karena jerawat ini masih tertutup dengan kulit. Sehingga tak ada kotoran lain yang menyumbat. Sedangan untuk blackheads merupakan jenis jerawat yang terbuka, sehingga folikel pada kulit pun tertutup dengan debu atau kotoran.

2) Acne Infammatory

Inflammatory memiliki dua jenis jerawat, yaitu papules dan pustules. Kedua jenis jerawat ini akan berwarna kemerahan dan menimbulkan rasa sakit atau pegal jika tertekan.

Gambar 2.2 Acne Inflammatory

Namun yang membedakan ialah, pada papules hanya berwarna merah dan pegal. Akan tetapi pada pustules ini selain berwarna merah dan pegal, jerawat ini pun berisi nanah seperti pada gambar 2.2. Jenis jerawat inflammatory ini pun tergolong pada jenis jerawat sedang.

2.2 Python

Python adalah interpreted high-level programming language for general-purpose programming. Dapat diartikan sebagai pemrograman tingkat tinggi yang dapat melakukan eksekusi sejumlah instruksi multi guna secara langsung dengan metode Object Oriented Programming serta dapat menggunakan semantik dinamis untuk memberikan tingkat keterbacaan syntax yang berguna untuk berbagai jenis tujuan (general-purpose).

Gambar 2.3 Logo Python Programming Language

Gambar 2.3 merupakan logo bahasa pemrograman python yang merupakan salah satu bahasa pemrograman tingkat tinggi. Sebutan bahasa pemrograman tingkat tinggi pada Python merujuk pada tingkat kedekatan sebuah bahasa pemrograman. Semakin tinggi kedekatannya, maka semakin serupa dengan bahasa manusia atau dalam hal ini bahasa inggris. Bahasa pemrograman Python merupakan yang paling sederhana dibandingkan bahasa yang lainnya karena memakai perintah dalam bahasa inggris konvensional.

Perintah bahasa pemrograman Python mudah untuk dipelajari, rapi, dan sederhana, bahkan beberapa aspek jauh lebih baik daripada bahasa pemrograman yang lainnya. Implementasi dari bahasa pemrograman Python termasuk beragam dan terus berkembang. Beberapa implementasi tersebut seperti web (Flask), data science (Numpy dan Pandas), data mining (Sklearn), deep learning (Keras), hingga machine learning (Tensorflow). Umumnya bahasa Python digunakan peneliti data dalam melakukan penelitian.

Saat ini syntax python dapat dijalankan dan ditulis untuk membangun perangkat lunak hinggal model machine learning di berbagai platform sistem operasi, beberapa diantaranya adalah:

1) Android
2) Microsoft/Windows
3) Linux/Unix
4) Mac OS
5) Java Virtual Machine

Dengan kode yang sederhana dan mudah diimplementasikan, programmer dapat mengutamakan pengembangan perangkat lunak yang dibuat. Selain itu, python merupakan salah satu produk open source dan multiplatform. Beberapa fitur yang dimiliki Python adalah:

1) Memiliki *library* atau perpustakaan yang luas. Dalam distribusi Python telah menyediakan fasilitas berupa modul yang siap pakai untuk berbagai keperluan dan memiliki tata bahasa yang mudah dipelajari.
2) Memiliki keteraturan layout kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber berorientasi objek.
3) Memiliki sistem pengelolaan memori otomatis (garbage collection) modular, mudah dikembangkan dengan menciptakan modul-modul baru.
4) Python memiliki pengaturan penggunaan ingatan komputer sehingga para pemakai tidak perlu melakukan pengaturan ingatan komputer secara langsung.

Python didistribusi dengan beberapa lisensi yang berbeda dari beberapa versi. Namun pada prinsipnya, Python dapat diperoleh dan digunakan secara bebas, bahkan untuk keperluan komersial. Lisensi Python tidak bertentangan baik menurut definsi open source maupun General Public License (GPL)

2.2.1 Numpy

Untuk melakukan struktur data dasar yang digunakan untuk data dan parameter model dapat menggunakan Numpy. Library numpy menyediakan struktur array multidimensional tingkat tinggi serta memungkinkan kontrol kerja yang sangat baik atas kinerja dan manajemen memori. Data masukan disajikan sebagai array didalam numpy, sehingga berintegrasi dengan *library* Python lainnya. Model memori berbasis Numpy memberikan batasan Salinan, bahkan ketika melakukan binding dengan kode yang dikompilasi. Numpy juga menyediakan operasi aritmatika dasar [Pedregosa, 2011].

2.2.2 Pandas

Library Pandas, dikembangkan sejak 2008, dibuat untuk dapat mempermudah kekayaan tools analisis data yang tersedia pada Python, sistem umum, bahasa komputasi ilmiah, dan berbagai domain platform komputasi statistik yang dengan basis data. Pandas tidak hanya bertujuan untuk menyediakan fungsionalitas yang setara tetapi juga mengimplementasikan banyak fitur, seperti penyelarasan data otomatis dan pengindeksan hierarkis yang tidak tersedia dalam cara yang diintegrasikan pada *library* lain atau lingkungan komputasi lainnya. Kumpulan datasets pada Pandas umumnya hadir dalam format tabel, yaitu sebagai daftar pengamatan dalam dua dimensi bidang untuk setiap observasi, Suatu pengamatan dapat diidentifikasi secara unik oleh satu atau lebih nilai atau label [McKinney, 2011].

2.2.3 Matplotlib

Matplotlib adalah *plotting library* atau pustaka dari Python yang berperan sebagai pemeta objek 2D. Matplotlib menghasilkan angka yang berkualitas dalam berbagai format hardcopy dan lingkungan interaktif diseluruh platform. Matplotlib dapat digunakan dalam script Python, shell Python, Jupyter Notebook, Google Colaboratory, server aplikasi web, dan toolkit antarmuka pengguna grafis. Matplotlib membantu menyediakan fasilitas pembuatan plot, histogram, spektrum daya, diagram batang, tabel kesalahan, scatterplots, dan lainnya. Untuk plot sederhana, modul pyplot menyediakan antarmuka seperti MATLAB [Website Resmi Matplotlib, 2018].

2.2.4 Scikit Learn

Scikit-learn atau Sklearn adalah modul Python yang menintegrasikan berbagai algoritma machine learning, baik dalam supervised atau unsupervised learning dalam skala menengah. Package ini berfokus membawa machine learning menggunakan bahasa tingkat tinggi. Library ini memiliki dependencies yang minimal dan didistribusikan untuk kemudahan penggunaan, kinerja, dokumentasi, dan konsistensi API [Pedregosa, 2011].

Scikit-learn menggunakan antarmuka yang bersifat *task oriented*, sehingga memungkinkan membandingkan metode yang mudah untuk aplikasi yang digunakan, Dengan penggunaannya dalam environtment pada Python, library ini dapat dengan mudah diintegrasikan ke dalam aplikasi di luar jangkauan analisis tradisional dalam data statistik. Algoritma yang diimplementasikan dalam bahasa tingkat tinggi dapat digunakan sebagai blok dalam pendekatan khusus untuk kasus berbagai user, sebagai contoh dalam pencitraan medis [Pedregosa, 2011].

2.3 Google Colaboratory

Google Colaboratory atau Google Colab merupakan tools yang berbasis cloud dan gratis untuk tujuan penelitian. Google Colab dibuat khusus untuk para programmer atau researcher yang mungkin kesulitan untuk mendapatkan akses komputer dengan spesifikasi yang mempuni.

Gambar 2.4 Logo Google Colaboratory

Google Colaboratory seperti logo yang ditunjukkan pada gambar 2.4, dibuat dengan environtment jupyter notebook dan mendukung hampir semua *library* yang dibutuhkan dalam lingkungan pengembangan Artificial Intelligence (AI).

Pada dasarnya colab sama dengan jupyter notebook dan dapat dikatakan bahwa google colab adalah jupyter notebook yang dijalankan via online. Dengan

kata lain, google meminjam komputer secara gratis. Berikut merupakan beberapa fitur yang ada pada google colab:

1) Spesifikasi tinggi

   Google colab memudahkan untuk menjalankan program pada komputer spesifikasi yang tinggi (GPU Tesla, RAM 13 GB, Disk 130 GB yang tersambung dengan layanan Google Drive) dan dapat melakukan running pelatihan model dalam waktu yang lama hingga 12 jam.

2) Kolaborasi

   Memudahkan untuk berkolaborasi dengan orang lain dengan cara membagi pengkodean secara online. Dapat disimpan dengan script ke dalam project github ataupun berbagi link dengan orang lain.

3) Zero configuration

   Dalam menggunakan google colab, pengguna tidak memerlukan konfigurasi apapun. Namun, ketika ingin menambahkan library baru, maka perlu melakukan install library package.

4) Fleksibel

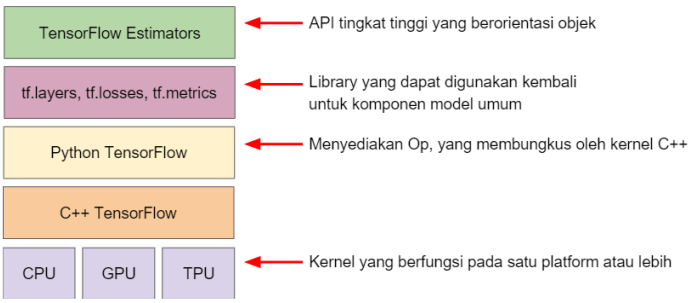
   Pada esensinya google colab hanya perlu running di browser, pengguna dapat mengawasi proses training via browser smartphone pengguna selama smartphone terhubung dengan Google Drive yang sama.

2.4 Tensorflow

TensorFlow adalah platform open source end-to-end untuk machine learning. Ini memiliki ekosistem alat, pustaka, dan sumber daya komunitas yang komprehensif dan fleksibel yang memungkinkan peneliti mendorong ML mutakhir dan pengembang dengan mudah membangun dan menerapkan aplikasi yang didukung ML.

Gambar 2.5 Logo Tensorflow

Tensorflow seperti yang digambarkan sebagai logo pada gambar 2.5 adalah sebuah framework komputasional untuk membuat model machine learning. TensorFlow menyediakan berbagai toolkit yang memungkinkan pengguna membuat model pada tingkat abstraksi yang disukai. Pengguna dapat menggunakan API dengan tingkat yang lebih rendah untuk membuat model dengan menentukan serangkaian operasi matematis. Sebagai alternatif, Pengguna juga dapat menggunakan API dengan tingkat yang lebih tinggi (seperti tf.estimator) untuk menentukan arsitektur yang telah ditetapkan, seperti regresi linier atau jaringan neural [Website Resmi Tensorflow].



Gambar 2.6 Hierarki Toolkit Tensorflow

TensorFlow terdiri dari dua komponen yakni, buffer protocol grafik dan waktu proses yang menjalankan grafik terdistribusi. Kedua komponen ini bersifat analog terhadap kode Python dan penafsir Python. Sama seperti penafsir Python yang diterapkan pada beberapa platform hardware untuk menjalankan kode Python, TensorFlow dapat menjalankan grafik pada beberapa platform hardware, termasuk CPU, GPU, dan TPU.

Tensorflow menyediakan berbagai pilihan API. Pengguna dapat menggunakan tingkat abstraksi tertinggi untuk dapat memecahkan masalah. Tingkat abstraksi yang lebih tinggi lebih mudah digunakan, tetapi juga (berdasarkan desain) kurang fleksibel. Berdasarkan dokumentasi tensorflow pada laman resminya, disarankan untuk memulai dengan API tingkat tertinggi terlebih dahulu dan pastikan semuanya berfungsi. Jika pengguna memerlukan fleksibilitas tambahan untuk beberapa masalah pemodelan khusus, berpindahlah ke satu tingkat lebih rendah. Perhatikan bahwa setiap tingkat dibuat menggunakan API di tingkat yang lebih rendah, sehingga menurunkan hierarki seharusnya cukup mudah.

Tensorflow diciptakan oleh tim Google Brain yang mana framework tersebut ialah library open source yang digunakan untuk komputasi numerik dan proyek machine learning berskala besar. Tensorflow menggabungkan banyak model dan algoritma machine learning termasuk deep learning (neural network). Tensorflow dibangun menggunakan Python front-end API untuk membuat suatu aplikasi penggunaannya, dan menggunakan C++ yang memiliki kinerja terbaik dalam proses eksekusi kode.

Tensorflow dapat melatih dan menjalankan neural network untuk keperluan klasifikasi tulisan tangan, pengenalan gambar/objek, serta menggabungkan kata. Selanjutnya adalah recurrent neural network yang merupakan model sequential. Umumnya diimplementasikan pada proyek Natural Language Processing (NLP), PDE (Partial Differential Equation) berdasarkan simulasi.

TensorFlow memungkinkan Anda membuat grafik dan struktur aliran data untuk menentukan bagaimana data bergerak melalui grafik dengan mengambil input sebagai larik multidimensi yang disebut Tensor. Ini memungkinkan Anda untuk membuat diagram alur operasi yang dapat dilakukan pada input ini, yang berjalan di satu ujung dan datang di ujung lain sebagai output.

2.5 Machine Learning

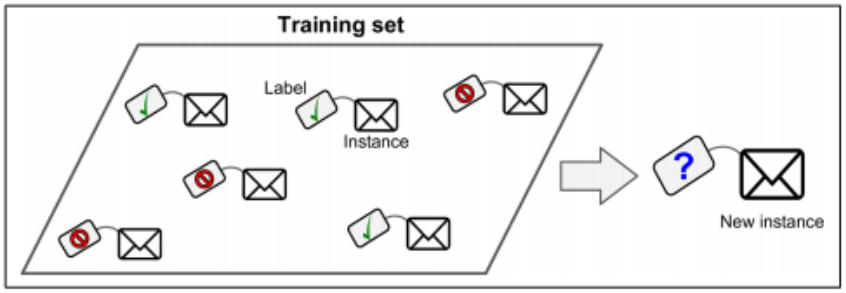
Machine Learning adalah bagian dari kecerdasan buatan atau Artificial Intelligence yang membuat sistem memiliki kemampuan belajar secara mandiri dengan meningkatkan kemampuannya berdasarkan pengalaman tanpa deprogram secara eksplisit. Dalam hal ini, program computer tidak ditulis secara statis.

Menurut definisi lainnya, machine learning merupakan ilmu yang memungkinkan komputer berperilaku seperti manusia, dimana komputer dapat meningkatkan pemahamannya melalui pengalaman atau dengan berjalannya waktu secara otomatis [Fragella, 2020]. Kemampuan belajar tersebut diperoleh dengan mempelajari sejumlah data sehingga dapat menghasilkan suatu model untuk melakukan proses input-ouput tanpa menggunakan kode program yang dibuat secara eksplisit.

Machine learning atau pembelajaran mesin berfokus pada pengembangan program komputer yang dapat mengakses data dan menggunakannya untuk belajar sendiri. Proses pembelajaran dimulai dari observasi data berdasarkan pengalaman langsung atau instruksi atau mencari pola data untuk membuat keputusan strategis yang lebih baik dimasa depan berdasarkan contoh yang telah dipelajari sebelumnya. Tujuan utamanya adalah membiarkan komputer belajar secara otomatis tanpa intervensi atau bantuan manusia dan menyesuaikan aktifitas yang sesuai. Karena teknologi komputasi, machine learning telah memiliki peningkatan dibandingkan machine learning masa lalu.

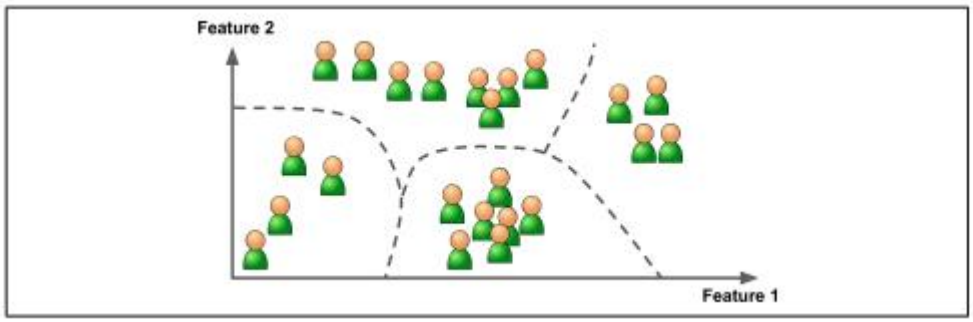
Pada perkembangannya di industri, terdapat berbagai macam algoritma machine learning yang dipetakan sesuai dengan efisiensi dan spesifikasi kasus yang berbeda-beda. Secara garis besar terdapat 2 teknik dalam machine learning, yaitu belajar dengan pengawasan (supervised learning) dan belajar tanpa pengawasan (unsupervised learning). Supervised learning adalah sebuah pendekatan yang mana sudah terdapat data yang dilatih, dan terdapat variabel yang ditargetkan sehingga tujuan dari pendekatan ini adalah mengelompokkan suatu data ke data yang sudah ada. Selain itu juga terdapat pendekatan lain yakni unsupervised learning. Unsupervised learning adalah sebuah pendekatan yang

tidak memiliki data latih, sehingga dari data yang ada akan dikelompokan menjadi beberapa bagian.



Gambar 2.7 Spam Email Classifier pada Supervised Learning

Tugas supervised learning yang khas adalah klasifikasi. Gambar 2.7 adalah ilustrasi dari klasifikasi spam pada email. Filter spam adalah contoh yang tepat untuk menggambarkan cara kerja supervised learning. Filter spam dilatih dengan banyak contoh email bersama dengan kelas mereka (spam atau ham), machine learning akan belajar bagaimana mengklasifikasikan email baru. Tugas tipikal lainnya adalah memprediksi nilai numerik target, seperti harga mobil, diberikan serangkaian fitur (jarak tempuh, usia, merek, dll.) yang disebut prediktor. Untuk melatih sistem, Anda perlu memberikan banyak contoh mobil, termasuk prediktor dan labelnya (yaitu, harganya) [Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow, Aurelien Geron].



Gambar 2.8 Customer Segmentation pada Unsupervised Learning

Sedangkan tugas unsupervised learning yang khas adalah clustering. Misalnya, terdapat pengguna yang bekerja sebagai admin memiliki banyak data tentang pengunjung blognya. Admin ingin menjalankan algoritma pengelompokan untuk mencoba mendeteksi kelompok pengunjung yang serupa seperti yang ditunjukan pada Gambar 2.8. Pada kasus ini, unsupervised learning mampu membantu admin dalam melakukan training dengan data yang tidak berlabel atau tidak dikenali sebelumnya, sehingga algoritma unsupervised learning akan belajar sepenuhnya secara mandiri tanpa petunjuk. Unsupervised akan menemukan koneksi tanpa bantuan admin. Mungkin terlihat bahwa 40% pengunjung pada grafik diatas adalah laki-laki yang menyukai buku komik dan biasanya membaca blog Anda di malam hari, sedangkan 20% adalah pecinta sci-fi yang berkunjung selama akhir pekan. Dengan demikian admin dapat menggunakan hierarki algoritma clustering untuk membagi setiap kelompok menjadi kelompok yang lebih kecil dan kedepannya admin juga dapat memasang strategi dalam menargetkan produk untuk pelanggan berdasarkan data yang telah di training oleh algoritma clustering [Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow, Aurelien Geron].

Pada penulisan ini, Teknik yang digunakan adalah supervised learning dengan data latih yang sudah tersedia. Algoritma pembelajaran mesin yang digunakan adalah classification (klasifikasi) yang merupakan bagian dari supervised learning atau belajar dengan pengawasan. Permasalahan yang dapat diselesaikan dengan menggunakan pembelajaran mesin dengan pengawasan (supervised learning) dapat dikelompokkan ke dalam dua kategori masalah yaitu Classification dan Regression.

Classification atau klasifikasi adalah suatu permasalahan dimana variabel keluarannya adalah berupa kategori (benar atau salah, iya atau tidak, baik atau buruk, dan lain sebagainya). Sedangkan Regression atau regresi adalah suatu permasalahan yang nilai keluarannya berupa nilai rill atau nilai yang bersifat kontinu (berat, mata uang, jarak, dan lain sebagainya). Terdapat beberapa masalah yang terbentuk dari kombinasi klasifikasi dan regresi, seperti masalah

rekomendasi dan prediksi terhadap data berbasis deret waktu [Jason Brownlee, 2016].

Beberapa contoh algoritma supervised learning, diantara lain:

1. K-Nearest Neighbors
2. Linear Regression dan Logistic Regression
3. Support Vector Machines
4. Decision Trees dan Random Forests
5. Neural Network

Beberapa contoh algoritma unsupervised learning, diantara lain:

1) Clustering
    a. K-Means
    b. DBSCAN
    c. Hierarchical Cluster Analysis (HCA)
2) Anomaly detection and novelty detection
    a. One-class SVM
    b. Isolation Forest
3) Visualization and dimensionality reduction
    a. Principal Component Analysis (PCA)
    b. Kernel PCA
    c. Locally Linear Embedding (LLE)
    d. t-Distributed Stochastic Neighbor Embedding (t-SNE)
4) Association rule learning
    a. Apriori
    b. Eclat

2.6 Computer Vision

Computer vision adalah bidang ilmu komputer yang berfokus pada replikasi bagian dari kompleksitas sistem penglihatan manusia dan memungkinkan komputer untuk mengidentifikasi dan memproses objek dalam gambar dan video

dengan cara yang sama seperti yang dilakukan manusia. Sampai saat ini, computer vision hanya bekerja dalam kapasitas yang terbatas.

Berkat kemajuan kecerdasan buatan dan inovasi dalam pembelajaran mendalam dan jaringan saraf, bidang ini telah mampu mengambil langkah besar dalam beberapa tahun terakhir dan telah mampu melampaui manusia dalam beberapa tugas yang berkaitan dengan mendeteksi dan memberi label objek. Salah satu faktor pendorong pertumbuhan computer vision adalah jumlah data yang dihasilkan hingga hari ini yang kemudian digunakan untuk melatih dan membuat computer vision menjadi lebih baik.

Seiring dengan sejumlah besar data visual (lebih dari 3 miliar gambar dibagikan secara online setiap hari), daya komputasi yang diperlukan untuk menganalisis data kini dapat diakses. Karena bidang visi komputer telah berkembang dengan perangkat keras dan algoritma baru, demikian juga tingkat akurasi untuk identifikasi objek. Dalam waktu kurang dari satu dekade, sistem saat ini telah mencapai akurasi 99 persen dari 50 persen membuatnya lebih akurat daripada manusia dalam bereaksi cepat terhadap input visual.

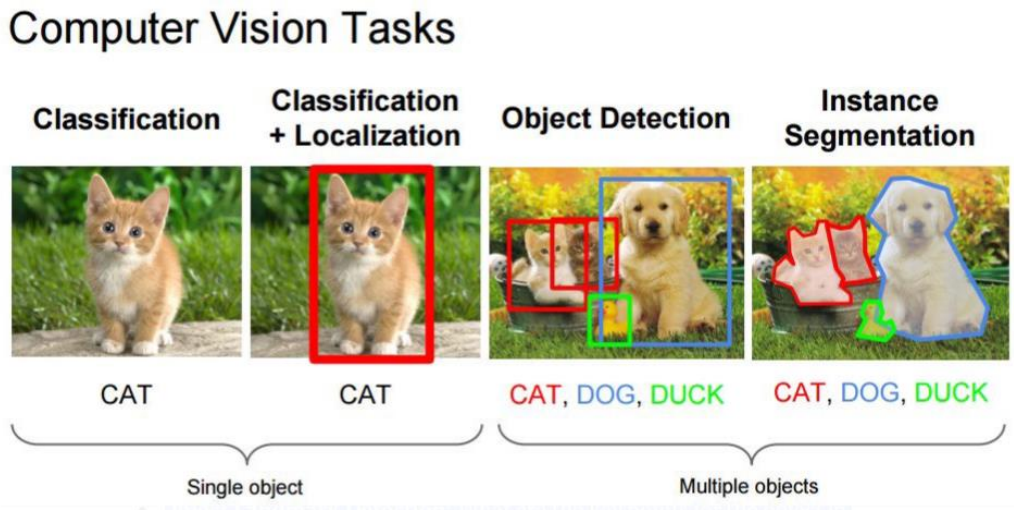
Eksperimen awal dalam visi komputer dimulai pada 1950-an dan pertama kali digunakan secara komersial untuk membedakan antara teks yang diketik dan tulisan tangan pada 1970-an, hari ini aplikasi untuk visi komputer telah berkembang secara eksponensial. Terdapat tiga langkah dari cara kerja computer vision mempelajari objek yang dilihat.

1. Mendapatkan Gambar

   Gambar, bahkan set besar, dapat diperoleh secara real-time melalui video, foto, atau teknologi 3D untuk analisis.
2. Processing Data Gambar

   Model pembelajaran mendalam mengotomatiskan banyak proses ini, tetapi model sering dilatih dengan terlebih dahulu diberi makan ribuan gambar berlabel atau pra-identifikasi.
3. Memahami Gambar

Langkah terakhir adalah langkah interpretatif, di mana suatu objek diidentifikasi atau diklasifikasikan.



Gambar 2.9 Kemampuan Visi Komputer

Banyak aplikasi visi komputer yang populer melibatkan upaya mengenali hal-hal dalam foto seperti pada gambar 2.9; sebagai contoh:

a. Object Classification: Apa kategori objek dalam foto ini?
b. Object Identification: Jenis objek apa yang ada dalam foto ini?
c. Object Verification: Apakah objek dalam foto?
d. Object Detection: Di mana objek dalam foto?
e. Object Landmark Detection: Apa poin penting untuk objek dalam foto?
f. Object Segmentation: Piksel apa saja yang termasuk dalam objek dalam gambar tersebut?
g. Object Recognition: Objek apa yang ada di foto ini dan di mana mereka?

Di luar pengakuan saja, metode analisis lainnya meliputi:

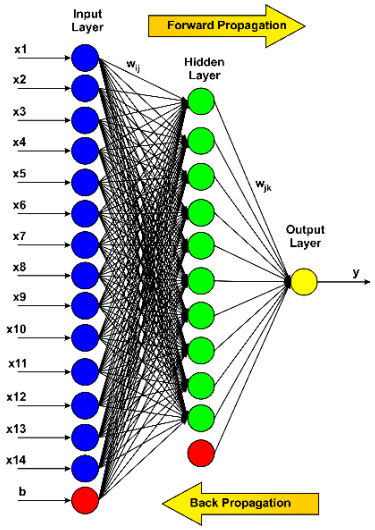
a. Video Motion Analysis: menggunakan visi komputer untuk memperkirakan kecepatan objek dalam video, atau kamera itu sendiri.

b. Image Segmentation: algoritma mempartisi gambar menjadi beberapa set tampilan.
c. Scene Reconstruction: menciptakan model 3D dari pemandangan yang dimasukkan melalui gambar atau video.
d. Image Restoration: noise seperti blur dihilangkan dari foto menggunakan filter berbasis Machine Learning.

Aplikasi lain yang melibatkan pemahaman piksel melalui perangkat lunak dapat dengan diberi label sebagai visi komputer (computer vision).

2.7 Jaringan Saraf Tiruan

Jaringan saraf tiruan bisa dibayangkan seperti otak buatan di dalam cerita-cerita fiksi ilmiah. Otak buatan ini dapat berpikir seperti manusia, dan juga sepandai manusia dalam menyimpulkan sesuatu dari potongan-potongan informasi yang diterimanya. Khayalan manusia tersebut mendorong para peneliti untuk mewujudkannya. Komputer diusahakan agar bisa berpikir sama seperti cara berpikir manusia. Caranya adalah dengan melakukan peniruan terhadap aktivitas-aktivitas yang terjadi di dalam sebuah jaringan saraf biologis [Aji , 2016].



Gambar 2.10 Arsitektur Jaringan Saraf Tiruan

Pembagian arsitektur jaringan saraf tiruan bisa dilihat dari kerangka kerja dan skema interkoneksi seperti yang diilustrasikan pada gambar 2.10. Kerangka kerja

jaringan saraf tiruan bisa dilihat dari jumlah lapisan (layer) dan jumlah node pada setiap lapisan. Lapisan-lapisan penyusun jaringan saraf tiruan dapat dibagi menjadi tiga, yaitu lapisan input, lapisan tersembunyi, lapisan output:

1. Lapisan input Node-node di dalam lapisan input disebut unit-unit input. Unit-unit input menerima input dari dunia luar. Input yang dimasukkan merupakan penggambaran suatu masalah.
2. Lapisan tersembunyi Node-node di dalam lapisan tersembunyi disebut unit-unit tersembunyi. Output dari lapisan ini tidak secara langsung dapat diamati.
3. Lapisan output Node-node pada lapisan output disebut unit-unit output. Keluaran atau output dari lapisan ini merupakan output jaringan syaraf tiruan terhadap suatu permasalahan.

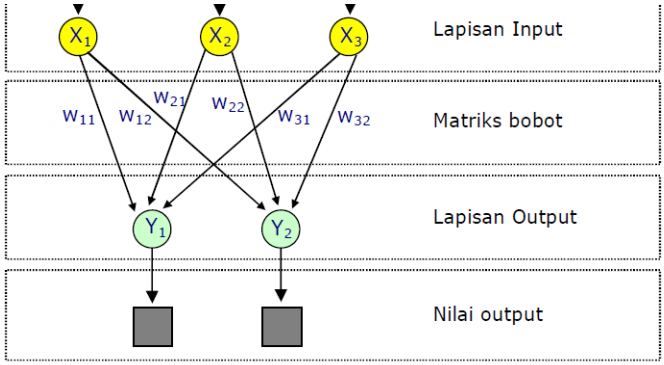
2.8 Multi Layer Perceptron

Jaringan Saraf Tiruan (JST) disusun dengan struktur dan fungsi otak manusia sebagai model untuk ditiru. Pada sebuah jaringan saraf tiruan terdapat sejumlah neuron. Satu neuron bisa terhubung ke banyak neuron lain, dan setiap koneksi (link) tersebut mempunyai bobot (weight). Tabel 2.1 merupakan analogi bagian-bagian dari jaringan saraf tiruan terhadap jaringan saraf biologis. Pembelajaran merupakan karakteristik dasar dari jaringan saraf biologis. Jaringan saraf tiruan melakukan proses pembelajaran melalui penyesuaian bobot pada koneksi antar neuronnya [Negnevitsky, Michael, 2005].

Tabel 2.1 Perbandingan Analogi Saraf Biologis dan Tiruan

Jaringan Saraf Biologis	**Jaringan Saraf Tiruan**
Soma	Neuron
Dendrit	Masukan (Input)
Axon	Keluaran (Output)
Synapse	Bobot (Weight)

Multilayer Perceptron adalah topologi paling umum dari JST, di mana perceptron-perceptron terhubung membentuk beberapa lapisan (layer). Sebuah MLP mempunyai lapisan masukan (input layer), minimal satu lapisan tersembunyi (hidden layer), dan lapisan luaran (output layer). Arsitektur JST ditunjukkan pada Gambar 2.11 [Negnevitsky, Michael, 2005].



Gambar 2.11 Arsitektur Multilayer Perceptron

Metode yang paling banyak digunakan dalam pembelajaran atau pelatihan MLP adalah propagasi balik (back-propagation). Terdapat empat langkah yang harus dilakukan dalam metode ini yaitu inisialisasi (initialization), aktivasi (activation), pelatihan bobot (weight training), dan iterasi (iteration). Pada langkah inisialisasi, nilai awal bobot dan ambang batas (threshold) ditentukan secara acak namun dalam batasan tertentu. Pada tahapan aktivasi, diberikan masukan dan nilai keluaran yang diharapkan (desired output). Proses penyesuaian bobot terjadi pada tahap pelatihan bobot, nilai luaran sebenarnya (actual output) dibandingkan dengan desired output dan dilakukan penyesuaian bobot. Langkah kedua dan ketiga diulangi sampai dengan tercapai kondisi yang ditentukan. Persamaan yang digunakan untuk pelatihan MLP Backpropagation sebagai berikut.

$$v = \Sigma_{i=1}^{r} x_i \,.\, w_i \qquad \textbf{(2.1)}$$

1. v = Nilai keluaran hidden layer.

2. x_i = Nilai input/fitur
3. w_i = Nilai bobot

Untuk merambatkan sinyal error, dimulai dari layer keluaran dan berjalan kembali ke hidden layer. Sinyal error di neuron k pada iterasi p diberikan pada persamaan berikut :

$$e_k(P) = y_{dk}(P) - y_k(P) \quad \textbf{(2.2)}$$

1. e_k = Nilai selisih.
2. y_{dk} = Nilai sebenarnya
3. y_k = Nilai prediksi

Prosedur yang digunakan untuk memperbaharui bobot pada koneksi hidden layer ke output layer.

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk} \quad \textbf{(2.3)}$$

1. Δw_{jk} = Koreksi bobot.
2. w_{jk} = Nilai bobot

Kondisi yang dialami adalah masukan neuron pada output layer berbeda dari input neuron pada input layer xi. Oleh karena itu yang digunakan untuk menghitung koreksi bobot adalah sinyal output neuron j pada hidden layer yj untuk menggantikan xi, Koreksi bobot dalam MLP dihitung dengan persamaan :

$$\Delta w_{jk}(p) = \eta * x_i(p) * \delta_j(p) \quad \textbf{(2.4)}$$

1. η = Learning rate
2. $\delta_j(p)$ = Gradien error
3. p = Iterasi
4. Δw_{jk} = Koreksi bobot

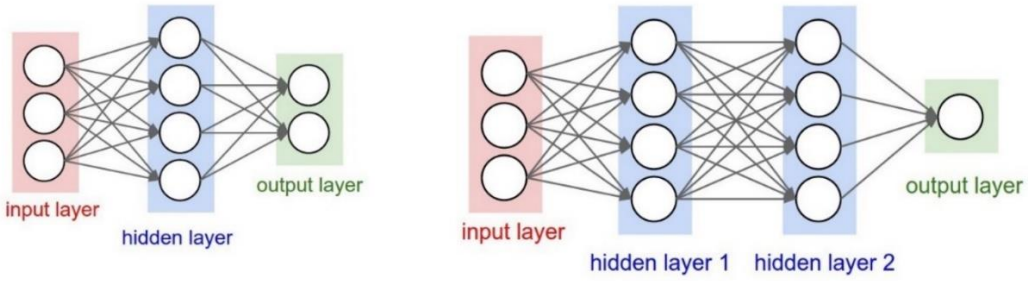
η adalah laju pembelajaran, sedangkan $\delta_j(p)$ adalah gradien error pada neuron k dalam output layer pada iterasi ke p.

2.9 Convolution Neural Network

Convolutional Neural Network (CNN) adalah pengembangan dari Multilayer Perceptron (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis Deep Neural Network karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik. CNN pertama kali dikembangkan dengan nama NeoCognitron oleh Kunihiko Fukushima, seorang peneliti dari NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Jepang. Konsep tersebut kemudian dimatangkan oleh Yann LeChun, seorang peneliti dari AT&T Bell Laboratories di Holmdel, New Jersey, USA. Model CNN dengan nama LeNet berhasil diterapkan oleh LeChun pada penelitiannya mengenai pengenalan angka dan tulisan tangan.

Pada tahun 2012, Alex Krizhevsky dengan penerapan CNN miliknya berhasil menjuarai kompetisi ImageNet Large Scale Visual Recognition Challenge 2012. Prestasi tersebut menjadi momen pembuktian bahwa metode Deep Learning, khususnya CNN. Metode CNN terbukti berhasil mengungguli metode Machine Learning lainnya seperti SVM pada kasus klasifikasi objek pada citra.

Cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi.



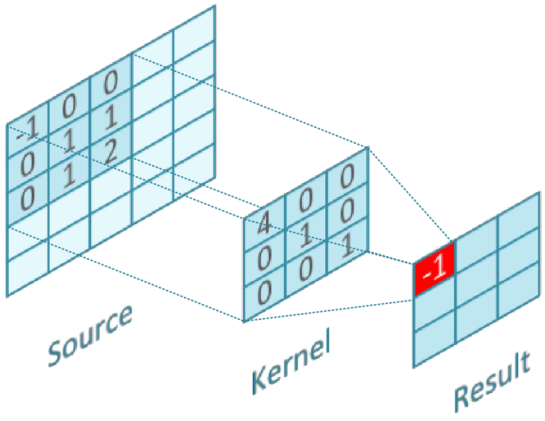
Gambar 2.12 Arsitektur Convolution Neural Network

Sebuah MLP seperti pada Gambar 2.12 memiliki i layer (kotak merah dan biru) dengan masing-masing layer berisi j_i neuron (lingkaran putih). MLP menerima input data satu dimensi dan mempropagasikan data tersebut pada jaringan hingga menghasilkan output. Setiap hubungan antar neuron pada dua layer yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode. Disetiap data input pada layer dilakukan operasi linear dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi non linear yang disebut sebagai fungsi aktivasi.

Pada CNN, data yang dipropagasikan pada jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar 2.13. Persamaan dimensi bobot pada CNN dijelaskan pada persamaan 2.5.

$$dimensi\ bobot = neuron\ input * neuron\ output * tinggi * lebar \quad \textbf{(2.5)}$$

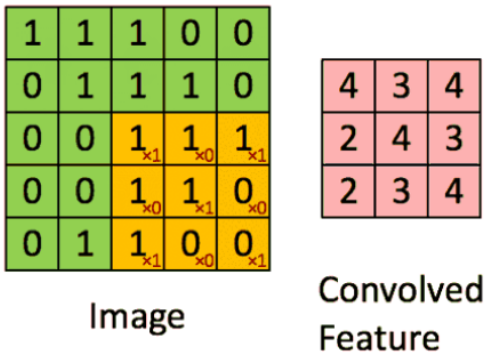
Karena sifat proses konvolusi, maka CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara.



Gambar 2.13 Ilustrasi Proses Konvolusi

2.10 Convolutional Layer

Convolution Layer merupakan blok bangunan inti CNN, dimana sebagian komputasi dilakukan di lapis ini. Convolution layer dengan satu lembaran neuron berisi 28 x 28. Masing-masing terhubung dengan suatu area kecil dalam (citra) masukan, masukkan 5 x 5 (piksel yang merupakan bidang reseptif (receptive field) untuk setiap neuron dan menyatakan bahwa filter yang digunakan berukuran 5 x 5, seluruh bidang reseptif akan ditelusuri secara tumpang tindih parsial, maka semua neuron tersebut pasti berbagi bobot koneksi (weight sharing).



Gambar 2.14 Illustrasi Perhitungan Konvolusi

Gambar 2.13 merupakan ilustrasi dari proses konvolusi dimana Convolution layer dalam arsitektur CNN umumnya menggunakan lebih dari satu filter. Jika menggunakan empat filter maka convolution layer. Neuron ke (i,j) pada hidden layer, memiliki nilai keaktifan y yang dihitung sesuai dengan Persamaan 2.19, dimana nilai (m,n) pada persamaan tersebut menunjukkan ukuran local receptive fields/kernel.

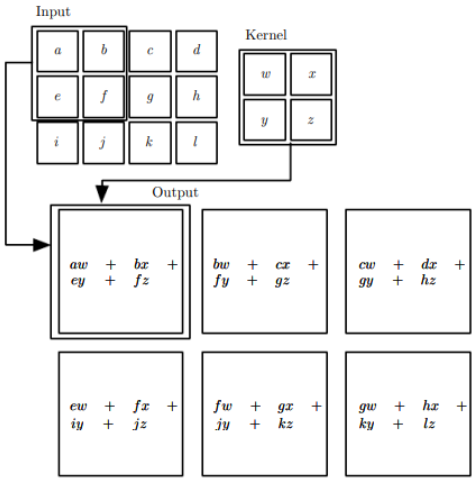
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)\, K(m,n) \qquad \textbf{(2.6)}$$

Perkalian antara input dengan kernel di atas (Persamaan 2.6) yang biasa disebut konvolusi. Namun, menurut Goodfellow, dkk. (2016) konvolusi dilakukan pada kernel yang terbalik, seperti pada Persamaan 2.7. Sedangakan jika kernel tidak dibalik maka fungsi itu disebut cross-correlation. Walaupun begitu, banyak

kode pustaka machine learning yang menggunakan rumus cross-correlation dan menyebutnya sebagai rumus konvolusi.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)\, K(m,n) \qquad \textbf{(2.7)}$$

Ukuran citra hasil konvolusi berkurang dibandingkan dengan citra awal dan dapat dinyatakan dengan Persamaan diatas. Dalam hal ini jika citra dengan ukuran 28x28 dikenai konvolusi dengan ukuran kernel 3x3 maka ukuran akhir menjadi 28-3+1 x 28- 3+1 = 26x26.



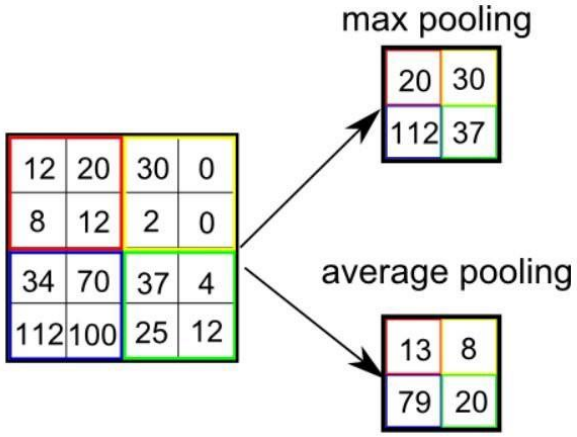
Gambar 2.15 Konvolusi Array Dua Dimensi

Pada gambar 2.15 diatas menunjukkan ilustrasi proses konvolusi pada citra, yang merupakan array dua dimensi I, dengan bobot K (dua dimensi). Pada gambar tersebut, citra berukuran 4x3 dikonvolusi dengan menggunakan kernel berukuran 2x2. Citra yang dihasilkan adalah berukuran 3x2. Elemen pertama pada citra hasil konvolusi adalah merupakan jumlah dari perkalian bobot kernel dengan nilai citra yang bersangkutan.

2.11 Pooling Layer

Pooling atau subsampling adalah pengurangan ukuran matriks dengan menggunakan operasi pooling. Terdapat dua macam pooling yang sering dipakai yaitu average pooling dan max - pooling. Dalam average pooling, nilai yang

diambil adalah nilai rata-rata, sementara pada max pooling, nilai yang diambil adalah nilai maksimal seperti yang diilustrasikan pada gambar 2.16.



Gambar 2.16 Perbedaan Max Pooling dan Average Pooling

Output dari proses pooling adalah matriks dengan dimensi yang lebih kecil dibanding dengan matrik awal. Proses konvolusi dan pooling dilakukan beberapa kali sehingga didapatkan peta fitur dengan ukuran yang dikehendaki.

Pooling layer digunakan untuk mengurangi ukuran tensor dan mempercepat perhitungan. Lapisan ini sederhana, kita perlu membagi gambar kita menjadi wilayah yang berbeda, dan kemudian melakukan beberapa operasi untuk masing-masing bagian tersebut. Misalnya, untuk max pool layer, pengguna dapat memilih nilai maksimum dari setiap wilayah dan meletakkannya di tempat yang sesuai di output. Seperti dalam kasus convolutional layer, konvolusi memiliki dua hyperparameter yang tersedia yaitu ukuran filter dan langkah. Pengguna dapat melakukan penyatuan untuk gambar multi-saluran (multi-channel), penyatuan untuk setiap saluran harus dilakukan secara terpisah.

Karena dalam lapisan jenis ini, pengguna tidak memiliki parameter apa pun yang harus diperbarui, tugas pengguna hanya mendistribusikan gradien dengan tepat. Dalam feedforward untuk max pooling, pengguna memilih nilai maksimum dari setiap wilayah dan mentransfernya ke lapisan berikutnya. Oleh karena itu jelas bahwa selama perambatan balik, gradien seharusnya tidak mempengaruhi

elemen matriks yang tidak termasuk dalam lintasan maju. Dalam praktiknya, ini dicapai dengan membuat masking yang mengingat posisi nilai yang digunakan pada fase pertama, yang nantinya dapat dimanfaatkan untuk mentransfer gradien.

2.12 Skenario Preprocessing

Dataset terkait jerawat diberi label dikarenakan jenis pembelajaran adalah pembelajaran supervised. Data pelatihan digunakan untuk melatih sebuah algoritma. Semakin banyak data pelatihan maka proses klasifikasi akan semakin baik.

2.12.1 Dataset Training

Dataset pelatihan adalah kumpulan contoh yang digunakan selama proses pembelajaran dan digunakan agar sesuai dengan parameter (mis. bobot) dari klasifikasi jerawat. Sebagian besar pendekatan yang mencari melalui data pelatihan untuk hubungan empiris cenderung sesuai dengan data, yang berarti bahwa mereka dapat mengidentifikasi dan mengeksploitasi hubungan nyata dalam data pelatihan yang tidak berlaku secara umum.

2.12.2 Dataset Testing

Dataset uji adalah dataset yang tidak tergantung pada dataset pelatihan, tetapi yang mengikuti distribusi probabilitas yang sama dengan dataset pelatihan. Jika model cocok dengan dataset pelatihan juga cocok dengan dataset uji dengan baik, overfitting minimal telah terjadi. Pemasangan yang lebih baik dari dataset pelatihan dibandingkan dengan dataset tes biasanya menunjuk pada overfitting.

2.12.3 Dataset Validation

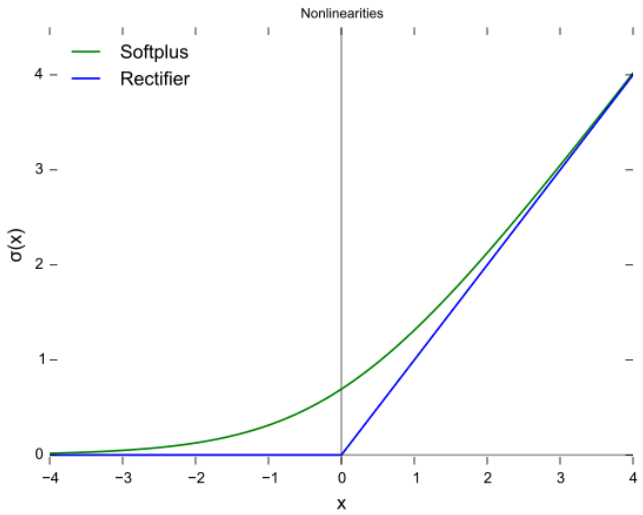
Dataset validasi masih bagian dari training set tetapi tidak terlebat dalam pembuatannya. Dataset validasi bisa untuk mengecek overfitting atau pengubahan nilai Hyperparameter

2.13 Rectified Linear Unit Layer

Rectified Linear Unit (ReLU) adalah bentuk fungsi aktivasi yang umum digunakan dalam model pembelajaran mendalam. Intinya, fungsi mengembalikan 0 jika menerima input negatif, dan jika menerima nilai positif. Fungsi akan

mengembalikan nilai positif yang sama. Fungsi tersebut diterjemahkan dalam persamaan:

$$f(x) = \max(0, x) \quad \textbf{(2.8)}$$



Gambar 2.17 Grafik Fungsi ReLU

ReLU memungkinkan model pembelajaran mendalam (deep learning) untuk memperhitungkan non-linier dan efek interaksi spesifik. Gambar 2.17 menampilkan representasi grafis dari fungsi ReLU. Perhatikan bahwa nilai untuk setiap input X negatif menghasilkan output 0, dan hanya setelah nilai positif dimasukkan, fungsi mulai miring ke atas.

Untuk memahami cara kerja ReLU, penting untuk memahami efeknya pada efek interaksi variabel. Efek interaksi adalah ketika variabel mempengaruhi prediksi tergantung pada nilai variabel terkait. Misalnya, membandingkan skor IQ dari dua sekolah yang berbeda mungkin memiliki efek interaksi IQ dan usia. IQ seorang siswa di sekolah menengah lebih baik daripada IQ seorang siswa sekolah dasar, karena usia dan IQ berinteraksi satu sama lain tanpa memandang sekolah. Ini dikenal sebagai efek interaksi dan ReLU dapat diterapkan untuk meminimalkan efek interaksi. Misalnya, jika A=1 dan B=2, dan keduanya

memiliki bobot terkait masing-masing 2 dan 3, fungsinya adalah, f(2A+3B). Jika A meningkat, output akan meningkat juga. Namun, jika B adalah nilai negatif yang besar, outputnya akan menjadi 0.

Manfaat menggunakan fungsi ReLU adalah kesederhanaannya membuatnya menjadi fungsi yang relatif murah untuk dihitung. Karena tidak ada matematika yang rumit, model dapat dilatih dan dijalankan dalam waktu yang relatif singkat. Demikian pula, konvergen lebih cepat, yang berarti kemiringan tidak mendatar karena nilai X semakin besar. Masalah gradien hilang ini dihindari di ReLU, tidak seperti fungsi alternatif seperti sigmoid atau tanh. Terakhir, ReLU jarang diaktifkan karena untuk semua input negatif, outputnya nol. Sparsity adalah prinsip bahwa fungsi tertentu hanya diaktifkan dalam situasi yang ringkas. Ini adalah fitur yang diinginkan untuk jaringan saraf modern, karena dalam jaringan yang jarang, kemungkinan besar neuron memproses bagian berharga dari suatu masalah dengan tepat. Misalnya, model yang memproses gambar ikan mungkin berisi neuron yang dikhususkan untuk mengidentifikasi mata ikan. Neuron spesifik itu tidak akan diaktifkan jika modelnya memproses gambar pesawat sebagai gantinya. Penggunaan fungsi neuron yang ditentukan ini menyebabkan sparity jaringan.

2.14 Softmax Layer

Fungsi softmax adalah fungsi yang mengubah vektor nilai real K menjadi vektor nilai real K yang berjumlah 1. Nilai input bisa positif, negatif, nol, atau lebih besar dari satu, tetapi softmax mengubahnya menjadi nilai antara 0 dan 1, sehingga dapat diinterpretasikan sebagai probabilitas. Jika salah satu inputnya kecil atau negatif, softmax mengubahnya menjadi probabilitas kecil, dan jika inputnya besar, maka itu mengubahnya menjadi probabilitas besar, tetapi akan selalu tetap antara 0 dan 1.

Fungsi softmax kadang-kadang disebut fungsi softargmax, atau regresi logistik multi-kelas. Hal ini karena softmax adalah generalisasi dari regresi logistik yang dapat digunakan untuk klasifikasi multi-kelas, dan rumusnya sangat mirip dengan fungsi sigmoid yang digunakan untuk regresi logistik. Fungsi

softmax dapat digunakan dalam pengklasifikasi hanya jika kelas-kelasnya saling eksklusif.

Banyak jaringan saraf multi-layer berakhir di lapisan kedua dari belakang yang menghasilkan skor bernilai nyata yang tidak mudah diskalakan dan yang mungkin sulit untuk dikerjakan. Di sini softmax sangat berguna karena mengubah skor menjadi distribusi probabilitas yang dinormalisasi, yang dapat ditampilkan ke pengguna atau digunakan sebagai input ke sistem lain. Untuk alasan ini biasanya ditambahkan fungsi softmax sebagai lapisan terakhir dari jaringan saraf.

Definisi matematis dari fungsi softmax ditunjukan pada persamaan 2.9.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \textbf{(2.9)}$$

1. $\vec{z}$ = Vektor input ke fungsi softmax, terdiri dari (z0, ... zK)
2. z_i = Elemen dari vektor input ke fungsi softmax, dan mereka dapat mengambil nilai real apa pun, positif, nol, atau negatif.
3. e^{z_i} = Fungsi eksponensial standar diterapkan pada setiap elemen dari vektor input. Ini memberikan nilai positif di atas 0, yang akan sangat kecil jika inputnya negatif, dan sangat besar jika inputnya besar. Namun, itu masih belum ditetapkan dalam kisaran (0, 1) yang diperlukan dari suatu probabilitas.
4. $\sum_{j=1}^{K} e^{z_j}$ = Normalisasi. Normalisasi memastikan bahwa semua nilai keluaran fungsi akan berjumlah 1 dan masing-masing berada dalam kisaran (0, 1), sehingga merupakan distribusi probabilitas yang valid.
5. K = Jumlah kelas dalam multi-class classifier.

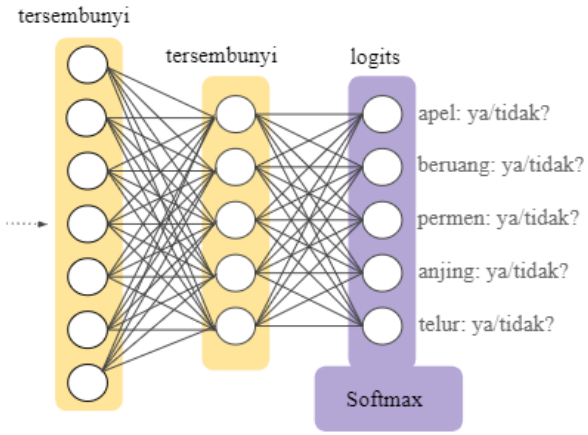
di mana semua nilai zi adalah elemen dari vektor input dan dapat mengambil nilai riil apa pun. Istilah di bagian bawah rumus adalah istilah normalisasi yang memastikan bahwa semua nilai keluaran fungsi akan berjumlah 1, sehingga merupakan distribusi probabilitas yang valid.

Softmax memperluas ide ini ke dunia kelas jamak atau multiclass. Dengan demikian, Softmax memberikan kemungkinan desimal ke setiap kelas dalam masalah kelas jamak. Total probabilitas desimal tersebut harus berjumlah 1,0. Batasan tambahan ini membantu agar pelatihan dikonvergensi dengan lebih cepat dari yang seharusnya. Misalnya, kembali ke analisis gambar pada Tabel 2.2, Softmax dapat menghasilkan kemungkinan sebuah gambar tergolong dalam kelas tertentu berikut:

Tabel 2.2 Tabel Kelas Label dan Persentase Peluang

Kelas	**Peluang**
Apel	0,001
Beruang	0,04
Permen	0,008
Anjing	0,95
Telur	0,001

Softmax diimplementasikan melalui lapisan jaringan neural tepat sebelum lapisan keluaran. Lapisan Softmax harus memiliki jumlah simpul yang sama dengan lapisan keluaran seperti pada gambar 2.18.



Gambar 2.18 Lapisan Softmax pada Jaringan Saraf

2.15 Optimizer

Optimizer atau pengoptimalan adalah algoritma atau metode yang digunakan untuk meminimalkan fungsi kesalahan (fungsi kerugian) atau untuk memaksimalkan efisiensi produksi. Optimizer adalah fungsi matematika yang bergantung pada parameter model yang dapat dipelajari, yaitu bobot & bias. Optimizer membantu mengetahui cara mengubah bobot dan kecepatan pembelajaran jaringan saraf untuk mengurangi kerugian.

2.15.1 Gradient Descent

Gradient descent atau penurunan gradien adalah algoritma optimasi yang paling dasar tetapi paling banyak digunakan. Ini banyak digunakan dalam regresi linier dan algoritma klasifikasi. Backpropagation dalam jaringan saraf juga menggunakan algoritma penurunan gradien.

Gradient descent adalah algoritma optimasi orde pertama yang bergantung pada turunan orde pertama dari fungsi kerugian. Ini menghitung ke arah mana bobot harus diubah sehingga fungsi dapat mencapai minimum. Melalui backpropagation, kerugian ditransfer dari satu lapisan ke lapisan lain dan parameter model yang juga dikenal sebagai bobot dimodifikasi tergantung pada kerugian sehingga kerugian dapat diminimalkan. Gradient Descent memiliki algoritma sebagai berikut:

$$\theta = \theta - a \,.\nabla J\,(\theta) \qquad \textbf{(2.10)}$$

Kelebihan:

1. Perhitungan yang mudah.
2. Mudah diimplementasikan.
3. Mudah dimengerti.

Kekurangan:

1. Dapat menjebak di minimum lokal.

2. Bobot diubah setelah menghitung gradien pada seluruh dataset. Jadi, jika kumpulan data terlalu besar dari ini, mungkin perlu waktu bertahun-tahun untuk konvergen ke minimum.
3. Memerlukan memori yang besar untuk menghitung gradien pada seluruh dataset.

2.15.2 Stochastic Gradient Descent

Stochastic Gradient Descent atau SGD adalah varian dari Gradient Descent. SGD mencoba memperbarui parameter model lebih sering. Dalam hal ini, parameter model diubah setelah perhitungan kerugian pada setiap contoh pelatihan. Jadi, jika dataset berisi 1000 baris SGD akan memperbarui parameter model 1000 kali dalam satu siklus dataset, bukan satu kali seperti pada Gradient Descent. SGD memiliki algoritma sebagai berikut:

$$\theta = \theta - a \,.\nabla J\left(\theta; x(i); y(i)\right) \qquad \textbf{(2.11)}$$

$$dimana\ \{x(i), y(i)\}\ adalah\ data\ sampel\ training$$

Karena parameter model sering diperbarui, parameter memiliki varians tinggi dan fluktuasi fungsi kerugian pada intensitas yang berbeda.

Kelebihan:

1. Pembaruan parameter model yang sering menyebabkan konvergen dalam waktu yang lebih singkat.
2. Membutuhkan lebih sedikit memori karena tidak perlu menyimpan nilai fungsi kerugian.
3. Mungkin mendapatkan minima baru.

Kekurangan:

1. Varians tinggi dalam parameter model.
2. Dapat menembak bahkan setelah mencapai minimum global.
3. Untuk mendapatkan konvergensi yang sama dengan gradient descent perlu dilakukan penurunan nilai learning rate secara perlahan.

2.15.3 Adam

Adam (Estimasi Momen Adaptif) bekerja dengan momentum orde pertama dan kedua. Intuisi di balik Adam adalah bahwa kita tidak ingin berguling begitu cepat hanya karena kita dapat melompati minimum, kita ingin sedikit mengurangi kecepatan untuk pencarian yang cermat. Selain menyimpan rata-rata penurunan eksponensial dari gradien kuadrat masa lalu seperti AdaDelta, Adam juga menyimpan rata-rata penurunan eksponensial dari gradien masa lalu M(t).

M(t) dan V(t) adalah nilai momen pertama yang merupakan Mean dan momen kedua yang masing-masing merupakan varians tidak terpusat dari gradien.

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \quad \textbf{(2.12)}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t} \quad \textbf{(2.13)}$$

Di sini, kita mengambil mean dari M(t) dan V(t) sehingga E[m(t)] dapat sama dengan E[g(t)] di mana, E[f(x)] adalah nilai harapan dari f(x). Untuk mempebaharui parameter, berikut bentuk formulanya:

$$\theta_{t+1} = \theta_t \frac{v_t}{\sqrt{\widehat{v_t}} - \epsilon}\widehat{m_t} \quad \textbf{(2.14)}$$

Nilai untuk 1 adalah 0,9, 0,999 untuk 2, dan (10 x exp(-8)) untuk ‘ϵ’.

Kelebihan:

1. Metode ini terlalu cepat dan konvergen dengan cepat.
2. Memperbaiki kecepatan belajar yang hilang, varians tinggi.

Kekurangan:

1. Computationally costly atau proses komputasi yang besar

2.16 Thresholding

Thresholding adalah jenis segmentasi gambar, di mana kita mengubah piksel gambar untuk membuat gambar lebih mudah dianalisis. Dalam thresholding,

programmer dapat mengubah gambar dari warna atau skala abu-abu menjadi gambar biner, yaitu gambar yang hanya hitam dan putih. Umumnya dapat menggunakan ambang batas sebagai cara untuk memilih area yang menarik dari suatu gambar, sementara mengabaikan bagian yang tidak kami pedulikan. Pada percobaan sebelumnya, thresholding telah dilakukan untuk menetapkan beberapa ambang batas sederhana, terutama pada bagian manipulasi piksel. Dalam hal ini, programmer dapat menggunakan manipulasi array NumPy sederhana untuk memisahkan piksel milik sistem dari latar belakang yang umumnya berwarna monokrom atau hitam.

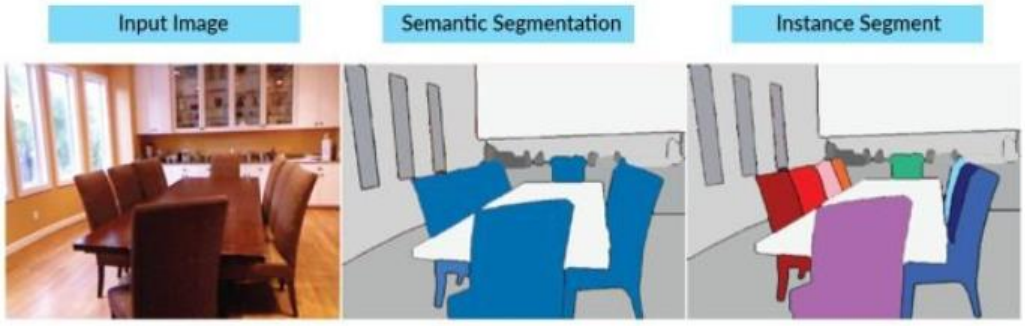
Prosesnya bekerja seperti ini. Pertama, kita akan memuat gambar asli, mengubahnya menjadi skala abu-abu, dan menyamarkan teksturnya dengan salah satu metode dari episode Blurring. Kemudian, dapat menggunakan operator > untuk menerapkan ambang batas t, angka dalam kisaran tertutup [0.0, 1.0]. Piksel dengan nilai warna di satu sisi t akan diaktifkan, sedangkan piksel dengan nilai warna di sisi lain akan dimatikan. Untuk menggunakan fungsi ini, kita harus menentukan nilai t yang baik. Bagaimana kita bisa melakukan itu? Salah satu caranya adalah dengan melihat histogram grayscale dari citra tersebut.

2.17 Segmentasi Gambar

Segmentasi gambar atau image segmentation adalah proses membagi gambar ke dalam wilayah yang berbeda berdasarkan karakteristik piksel untuk mengidentifikasi objek atau batas untuk menyederhanakan gambar dan menganalisisnya dengan lebih efisien. Segmentasi berdampak pada sejumlah domain, mulai dari industri pembuatan film hingga bidang kedokteran. Misalnya, perangkat lunak di belakang layar hijau mengimplementasikan segmentasi gambar untuk memangkas latar depan dan menempatkannya di latar belakang untuk pemandangan yang tidak dapat dipotret atau akan berbahaya untuk dipotret dalam kehidupan nyata. Segmentasi gambar juga digunakan untuk melacak objek dalam urutan gambar dan untuk mengklasifikasikan medan, seperti cadangan minyak bumi, dalam gambar satelit. Beberapa aplikasi segmentasi medis termasuk identifikasi otot yang cedera, pengukuran tulang dan jaringan, dan deteksi struktur

yang mencurigakan untuk membantu ahli radiologi (Computer Aided Diagnosis, atau CAD) [AI Stanford Edu].

Segmentasi gambar adalah proses dimana citra digital dipartisi menjadi berbagai subkelompok (piksel) yang disebut Objek Citra, yang dapat mengurangi kompleksitas citra, dan dengan demikian menganalisis citra menjadi lebih sederhana, terdapat 2 macam segmentasi yang umum digunakan, yakni semantic segmentation dan instance segmentation seperti yang ditunjukan pada gambar 2.19.



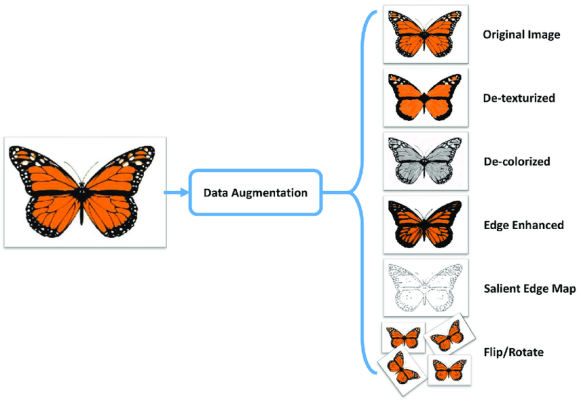
Gambar 2.19 Macam Segmentasi pada Gambar

Peneliti dan programmer pada umumnya menggunakan berbagai algoritma segmentasi gambar untuk membagi dan mengelompokkan sekumpulan piksel tertentu bersama-sama dari gambar. Dengan melakukan itu, peneliti dapat menetapkan label ke piksel dan piksel dengan label yang sama termasuk dalam kategori di mana mereka memiliki beberapa atau hal lain yang sama di dalamnya.

Dengan menggunakan label ini, kita dapat menentukan batas, menggambar garis, dan memisahkan objek yang paling dibutuhkan dalam gambar dari objek lain yang tidak terlalu penting. Pada contoh di bawah ini, dari gambar utama di sebelah kiri, kami mencoba untuk mendapatkan komponen utama, mis. kursi, meja dll dan karenanya semua kursi berwarna seragam. Di tab berikutnya, kami telah mendeteksi instance, yang berbicara tentang objek individual, dan karenanya semua kursi memiliki warna yang berbeda.

2.18 Augmentasi Gambar

Augmentasi data digunakan selama proses pelatihan CNN. Terutama ketika himpunan citra latih relatif sedikit. Augmentasi data ini dapat dipandang seperti ensemble learning, di mana sejumlah keluaran CNN dapat dilakukan voting (jika CNN berupa model klasifikasi) atau dirata rata kan (Jika CNN berupa model regesi) untuk menghasailkan keluaran akhir CNN tersebut (Lee et al.2014) Teknik augmentasi data dapat digunakan dengan sejumlah cara seperti pada ilustrasi gambar 2.20,



Gambar 2.20 Macam Fitur Augmentasi Data Gambar

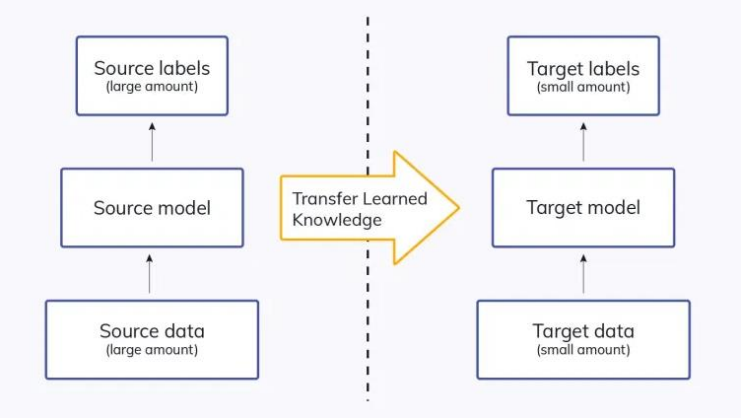
Diantaranya adalah flip, random croping, color jitter dan random combination. Flip atau pembalikan citra dapat dilakukan secara horizontal dan vertikal. Misalnya citra wajah yang menghadap ke kiri dapat dibalik menjadi ke kanan Random Croping untuk membuat variasi data dari berbagai sudut pandang. Seolah kita menguji sebauh masukan dengan banyak model klasifikasi. Color jitter untuk mendapatkan variasi warna yang acak.

Cara paling sederhana dalam teknik ini adalah dengan melakukan variasi kontras secara acak. Cara lain yang lebih kompleks adalah dengan mengaplikasikan principal component analysis (PCA) terhadap semua komponen warna RGB piksel dalam citra latih. Kemudian, lakukan penyampelan" himpunan warna" sepanjang arah principal component. Terakhir, tambahkan offset ke dalam semua piksel pada suatu citra latih. Random Combination untuk sejumlah teknik

pengolahan citra seperti translation atau pembesaran/pengecilan, rotation atau pemutaran dengan derajat tertentu, stretching atau peregangan/pengerutan, shearing, dan lens distortions. Pada saat pelatihan dapat dilakukan augmentasi data setiap data hasil augmentasi dilatihkan ke CNN dengan label yang sama dengan citra latih aslinya. Citra dilatih secara berulang ulang sampai nilai loss yang relatif kecil, yang berarti CNN sudah bisa membedakan kedua citra tersebut secara akurat. Setelah proses pelatihan selesai, CNN diuji. Citra diuji dengan augmentasi dengan sejumlah teknik seperti pada saat pelatihan. Setiap citra hasil augmentasi dima- sukkan ke CNN sedemikian hinggga dihasilkan sejumlah label hasil pengenalan. Selanjutnya, kita melakukan voting untuk menentukan label keputusan final CNN.

2.19 Transfer Learning

Transfer learning adalah sebuah metode yang memanfaatkan representasi fitur dari model yang telah dilatih sebelumnya, sehingga tidak perlu melatih model baru dari awal. Model pra-pelatihan pada umumnya dilatih pada kumpulan data besar yang merupakan tolok ukur standar dalam batas visi komputer. Bobot yang diperoleh dari model dapat digunakan kembali dalam tugas visi komputer lainnya seperti yang digambarkan pada gambar 2.21.

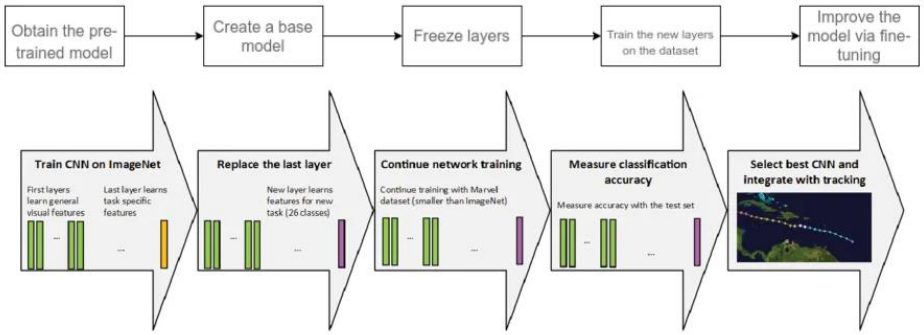


Gambar 2.21 Ilustrasi Transfer Learning

Model-model ini dapat digunakan secara langsung dalam membuat prediksi pada tugas-tugas baru atau diintegrasikan ke dalam proses pelatihan model baru.

Memasukkan model pra-pelatihan ke dalam model baru menghasilkan waktu pelatihan yang lebih rendah dan kesalahan generalisasi yang lebih rendah. Transfer learning sangat berguna dalam penelitian yang memiliki kumpulan data pelatihan kecil. Dalam hal ini, peneliti dapat menggunakan bobot dari model yang telah dilatih sebelumnya untuk menginisialisasi bobot model baru.

Pada gambar 2.22 terdapat 5 langkah dalam implementasi transfer learning:



Gambar 2.22 Langkah Implementasi Transfer Learning

1. Menentukan pre-trained model

   Langkah pertama adalah mendapatkan model pra-terlatih yang ingin digunakan untuk masalah tertentu.

2. Membuat model dasar

   Langkah kedua membuat instance model dasar menggunakan salah satu arsitektur seperti ResNet atau Xception, pada penulisan ini penulis menggunakan MobileNetV2. Pada proses ini peneliti dapat mengunduh beban pra-latihan secara opsional. Jika tidak mengunduh bobot, peneliti harus menggunakan arsitektur untuk melatih model dari awal. Terdapat catatan yang perlu diingat, bahwa pada umumnya model dasar akan memiliki lebih banyak unit di lapisan keluaran akhir daripada yang butuhkan. Saat membuat model dasar, diharuskan untuk menghapus lapisan keluaran akhir. Selanjutnya adalah menambahkan lapisan keluaran akhir yang kompatibel dengan masalah tertentu.

3. Membekukan lapisan (freeze layer)

Langkah ketiga adalah membekukan lapisan dari model yang telah dilatih sebelumnya. Hal ini bermanfaat jika tidak ingin bobot di lapisan tersebut diinisialisasi ulang. Jika ya, maka akan kehilangan semua pembelajaran yang telah terjadi.

4. Melatih layer baru pada dataset

   Langkah keempat adalah menambahkan lapisan baru yang dapat dilatih yang akan mengubah fitur lama menjadi prediksi pada kumpulan data baru. Hal ini dapat dipertimbangkan karena model pra-pelatihan (pre-trained) dimuat tanpa lapisan keluaran akhir. Pada pembahasan sebelumnya telah dituliskan bahwa keluaran akhir model pra-latihan kemungkinan besar akan berbeda dari keluaran yang dinginkan untuk model target. Sebagai contoh, model pra-pelatihan yang dilatih pada dataset ImageNet akan menghasilkan 1000 kelas. Namun, model milik peneliti mungkin hanya memiliki dua kelas. Dalam hal ini, peneliti harus melatih model dengan lapisan keluaran baru di tempatnya. Oleh karena itu, peneliti harus menambahkan beberapa lapisan padat baru. Lapisan padat akhir harus disesuaikan dengan unit yang sesuai dengan jumlah output yang diharapkan oleh model target tersebut.

5. Meningkatkan model melalui metode fine-tuning

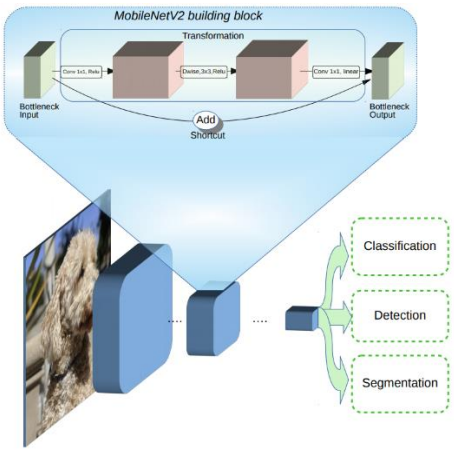
   Langkah terakhir, peneliti dapat meningkatkan kinerjanya melalui fine-tuning. Penyesuaian halus dilakukan dengan mencairkan model dasar atau sebagian darinya akan melatih seluruh model lagi di seluruh kumpulan data dengan kecepatan pembelajaran yang sangat rendah. Tingkat pembelajaran yang rendah akan meningkatkan kinerja model pada dataset baru sekaligus mencegah overfitting. Learning rate harus rendah karena modelnya cukup besar sedangkan datasetnya kecil. Hal tersebut merupakan solusi untuk overfitting, karenanya tingkat pembelajarannya rendah. Lakukan kompilasi ulang model setelah membuat perubahan sehingga dapat diterapkan. Hal tersebut terjadi karena perilaku model dibekukan setiap kali memanggil fungsi kompilasi. Peneliti harus memanggil fungsi kompilasi lagi setiap kali ingin mengubah perilaku

model. Langkah selanjutnya adalah melatih model lagi sambil memantaunya melalui callback untuk memastikannya tidak terlalu overfit.

2.20 MobileNetV2

Pada 24 Juni 2017, Tim Google AI mengumumkan perilisan MobileNetV2 untuk mendukung aplikasi mobile vision generasi berikutnya. MobileNetV2 adalah peningkatan yang signifikan atas MobileNetV1 dan mendorong berbagai kasus pengenalan visual mobile termasuk klasifikasi, deteksi objek dan segmentasi semantik. MobileNetV2 dirilis sebagai bagian dari TensorFlow-Slim Image Classification Library. MobileNetV2 juga tersedia sebagai modul di TF-Hub, dan pre-trained checkpoints sebelumnya dapat ditemukan di github.

MobileNetV2 dibangun di atas ide-ide dari MobileNetV1, menggunakan konvolusi yang dapat dipisahkan secara mendalam sebagai blok bangunan yang efisien. Namun, V2 memperkenalkan dua fitur baru ke arsitektur yakni linear bottleneck atau kemacetan linier antara lapisan, dan koneksi pintasan antara bottleneck. Struktur dasar ditunjukkan pada gambar 2.23 di bawah ini:

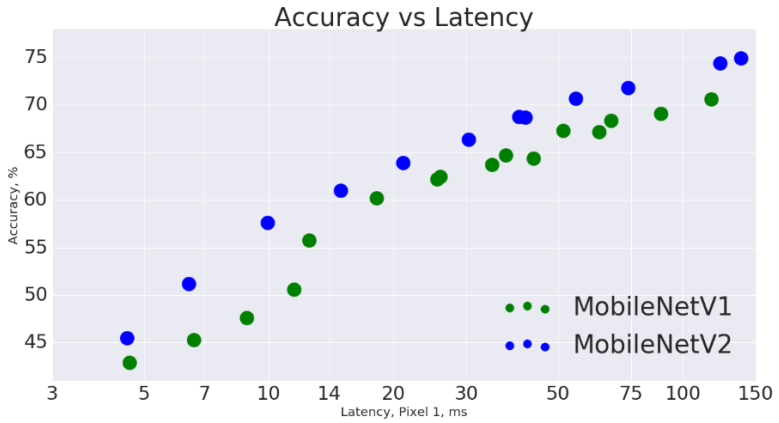


Gambar 2.23 Ilustrasi Building Block pada MobileNetV2

MobileNetV2 memiliki ciri khas yaitu bottleneck akan menyandikan input dan output perantara model sementara lapisan dalam (inner layer) akan mencerna

kemampuan model untuk mengubahnya dari konsep tingkat rendah seperti piksel ke deskriptor tingkat tinggi seperti kategori gambar. MobileNetV2 terdiri dari koneksi residual tradisional yang memiliki shortcuts atau pintasan sehingga memungkinkan proses pelatihan yang lebih cepat dengan akurasi yang lebih baik.

Dalam fase percobaan, arsitektur MobileNetV2 memiliki performa yang lebih cepat dengan mendapatkan akurasi yang maksimal di seluruh spektrum latensi. MobileNetV2 menggunakan operasi 2x lebih sedikit, membutuhkan parameter 30% lebih sedikit, dan sekitar 30-40% lebih cepat ketika dilakukan uji coba pada ponsel Google Pixel daripada model MobileNetV1 seperti pada gambar 2.24.



Gambar 2.24 Perbandingan Latensi MobileNetV1 dan MobileNetV2

MobileNetV2 adalah ekstraktor fitur yang sangat efektif untuk deteksi dan segmentasi objek. Misalnya, untuk deteksi saat dipasangkan dengan SSDLite, MobileNetV2 unggul sekitar 35% lebih cepat dengan memperoleh nilai akurasi yang sama dari MobileNetV1 seperti yang dijelaskan pada tabel 2.3.

Tabel 2.3 Perbandingan Dua Versi MobileNet + SSDLite

Model	**Params**	**Multiply-Adds**	**mAP**	**Mobile CPU**
MobileNetV1 + SSDLite	5.1 M	1.3B	22.2%	270ms
MobileNetV2 + SSDLite	4.3M	0.8B	22.1%	200ms

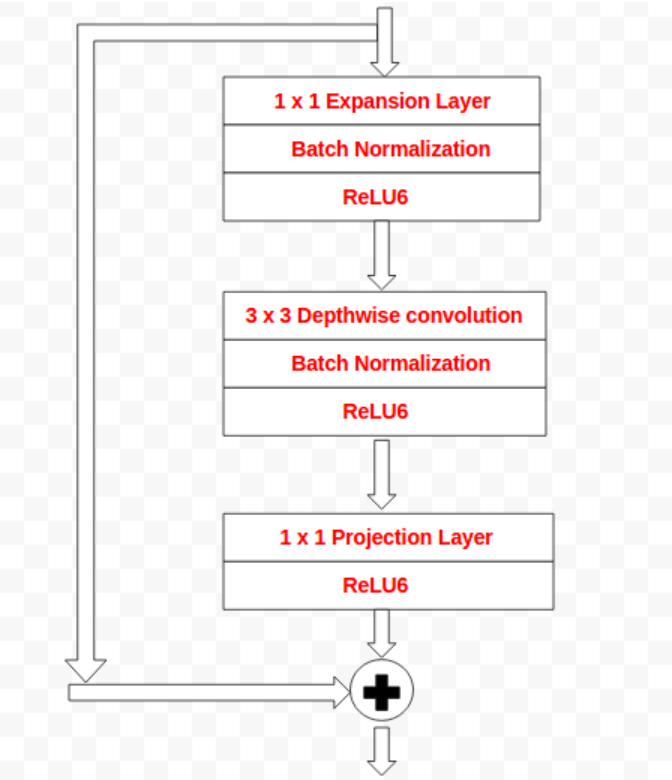
Untuk mengaktifkan segmentasi semantik pada perangkat, tim Google AI menggunakan MobileNetV2 sebagai ekstraktor fitur dalam DeepLabv3. Pada benchmark segmentasi semantik, PASCAL VOC 2012, sebagai pengekstrak fitur, MobileNetV2 menghasilkan kinerja yang sama dengan MobileNetV1. Pada percobaan ini, MobileNetV2 lebih unggul karena hanya membutuhkan parameter 5,3 kali lebih sedikit dan operasi 5,2 kali lebih sedikit dalam hal Multiply-Adds dibandingkan MobileNetV1 seperti pada tabel 2.4.

Tabel 2.4 Perbandingan Dua Versi MobileNet + DeepLabV3

Model	**Params**	**Multiply-Adds**	**mIOU**
MobileNetV1 + DeepLabV3	11.15M	14.25B	75.29%
MobileNetV2 + DeepLabV3	2.11M	2.75B	75.32%

2.21 Arsitektur MobileNetV2

Ide dasar di balik MobileNetV1 adalah mengganti konvolusi dengan proses yang mahal dengan yang lebih murah. Dan kehadiran V1 mendapatkan kesuksesan. Perubahan utama dalam arsitektur V2 adalah penggunaan blok bottleneck terbalik (inverted bottleneck) dan koneksi residual seperti pada gambar 2.25.



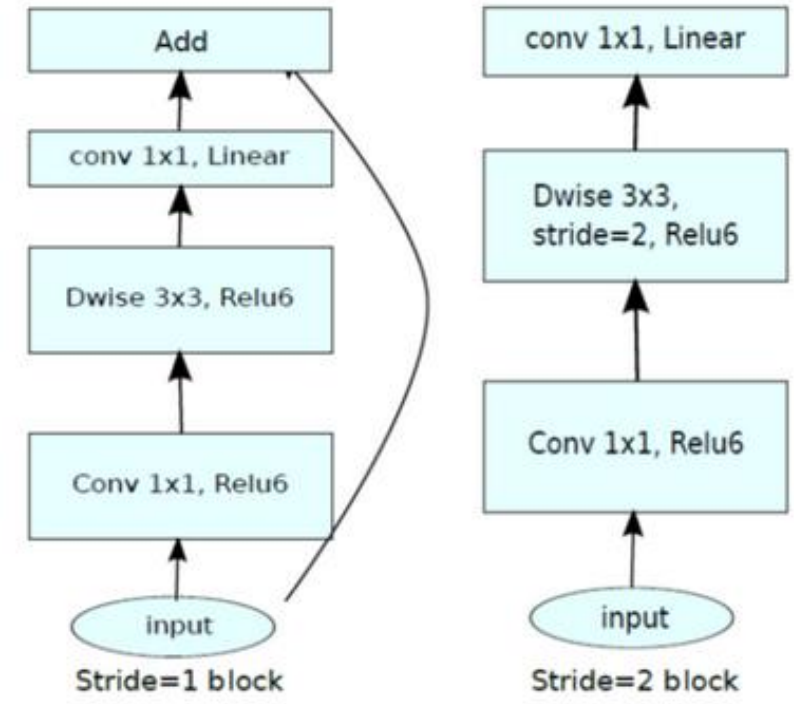
Gambar 2.25 Arsitektur Inverted Bottleneck pada MobileNetV2

MobileNetV2 masih menggunakan konvolusi yang dapat dipisahkan. Tetapi sekarang V2 memiliki blok residu bottleneck, bukan hanya blok konvolusi yang dapat dipisahkan. Ada 3 lapisan konvolusi di blok residu bottleneck. 2 lapisan terakhir telah hadir di MobileNetV1. Mereka adalah lapisan depth wise convolutional layer yang diikuti oleh lapisan konvolusi 1 x 1 titik ke titik.

Di MobileNetV1 konvolusi pointwise menjaga jumlah channel (saluran) tetap sama atau menggandakannya. Tapi di sini 1 x 1 konvolusi membuat jumlah channel lebih kecil. Ini dikenal sebagai lapisan proyeksi. Lapisan ini juga disebut lapisan bottleneck karena mengurangi jumlah aliran data yang melaluinya.

Lapisan pertama adalah lapisan ekspansi 1 x 1. Lapisan tersebut memperluas data (menambah jumlah saluran) yang mengalir melaluinya. Lapisan ekspansi merupakan kebalikan dari lapisan proyeksi yang mana data akan diperluas berdasarkan faktor ekspansi. Ini adalah hyperparameter yang dapat ditemukan dari

pertukaran arsitektur yang berbeda. Faktor ekspansi memiliki nilai default yakni 6.



Gambar 2.26 Perbedaan Kinerja MobileNetV2 Berdasarkan Jumlah Stride

Gambar 2.26 adalah Lapisan kedua yang merupakan depth wise convolution layer. Hal penting lainnya dalam blok residu bottleneck adalah koneksi residu. Pada arsitektur ini, ReLU6 digunakan sebagai fungsi aktivasi yang memiliki formula min(max(x, 0), 6). Setiap lapisan memiliki lapisan normalisasi batch dan fungsi aktivasi (ReLU6), kecuali lapisan proyeksi. Lapisan proyeksi hanya memiliki normalisasi batch karena output dari lapisan proyeksi berdimensi rendah. Jika depth wise convolution layer menggunakan stride=2, blok inverted bottleneck tidak akan memiliki koneksi residual.

Blok bangunan dasar atau *basic building block* adalah konvolusi yang dapat dipisahkan kedalamannya dengan residu. Struktur rinci dari blok ini ditunjukkan pada tabel berikut.

Tabel 2.5 Tabel Struktur Rinci pada Building Block MobileNetV2

Input	Operator	Output
$h * w * k$	1x1 conv2d, ReLU6	$h * w * (tk)$
$h * w * tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} * \frac{w}{s} * tk$
$\frac{h}{s} * \frac{w}{s} * tk$	Linear 1x1 conv2d	$\frac{h}{s} * \frac{w}{s} * k'$

Arsitektur MobileNetV2 berisi initial fully convolution layer dengan 32 filter, diikuti oleh 19 lapisan bottleneck sisa yang dijelaskan dalam Tabel 2.5 Google AI menggunakan ReLU6 sebagai non-linier karena ketahanannya ketika digunakan dengan perhitungan presisi rendah. Google AI juga menggunakan ukuran kernel 3 × 3 sebagai standar untuk jaringan modern, dan menggunakan normalisasi dropout serta batch selama pelatihan.

Google AI menggunakan konstanta tingkat ekspansi di seluruh jaringan. Dalam fase percobaan, Google AI menemukan bahwa tingkat ekspansi antara 5 dan 10 menghasilkan kurva kinerja yang hampir identik, jaringan yang lebih kecil membuat kinerja menjadi lebih baik dengan tingkat ekspansi yang sedikit lebih kecil dan jaringan yang lebih besar memiliki kinerja yang sedikit lebih baik dengan tingkat ekspansi yang lebih besar.

Dalam eksperimen utama, Google AI menggunakan faktor ekspansi bernilai 6 yang diterapkan pada ukuran tensor input. Sebagai contoh, untuk lapisan bottleneck yang menggunakan tensor input 64 channel akan menghasilkan tensor dengan 128 saluran, lapisan ekspansi menengah kemudian 64 * 6 = 384 saluran.

Seperti pada kami menyesuaikan arsitektur ke titik kinerja yang berbeda, dengan menggunakan resolusi gambar input dan pengganda lebar sebagai hyperparameter yang dapat diatur, yang dapat disesuaikan tergantung pada akurasi / kinerja trade-off yang diinginkan. Jaringan utama kami (pengganda lebar 1,224 × 224), memiliki, biaya komputasi 300 juta multiply-adds dan menggunakan 3,4

juta parameter. Google AI mengeksplorasi kinerja trade off, untuk resolusi input dari 96 hingga 224, dan pengganda lebar 0,35 hingga 1,4. Biaya komputasi jaringan berkisar dari 7 kalikan hingga 585M MAdds, sedangkan ukuran model bervariasi antara 1,7M dan 6,9 juta parameter. Pada tabel 2.6 dijelaskan bahwa terdapat perbedaan kecil pada implementasi, yaitu untuk pengganda kurang dari satu, Google AI menerapkan pengganda lebar ke semua lapisan kecuali lapisan konvolusi terakhir. Kiat tersebut dapat meningkatkan kinerja untuk model yang lebih kecil.

Tabel 2.6 Struktur Lapisan pada Arsitektur MobileNetV2

Input	**Operator**	**t**	**c**	**n**	**s**
$224^2 \ x \ 3$	Conv2d	-	32	1	2
$112^2 \ x \ 32$	bottleneck	6	16	1	1
$112^2 \ x \ 16$	bottleneck	6	24	2	2
$56^2 \ x \ 24$	bottleneck	6	32	3	2
$28^2 \ x \ 32$	bottleneck	6	64	4	2
$14^2 \ x \ 64$	bottleneck	6	96	3	1
$14^2 \ x \ 96$	bottleneck	6	160	3	2
$7^2 \ x \ 160$	bottleneck	6	320	1	1
$7^2 \ x \ 320$	Conv2d 1x1	-	1280	1	1
$7^2 \ x \ 1280$	Avgpool 7x7	-	-	1	-
$1 \ x \ 1 \ x \ 1280$	Conv2d 1x1	-	k	-	

Pada MobileNetV2, setiap baris menggambarkan urutan dari 1 atau lebih lapisan identik (modulo stride) dan berulang sebanyak n kali. Semua lapisan dalam urutan yang sama memiliki nomor c dari saluran output. Lapisan pertama dari masing-masing urutan memiliki langkah s dan menggunakan stride=1. Semua konvolusi spasial menggunakan kernel berukuran 3 × 3. Ekspansi faktor t selalu diterapkan ke ukuran input seperti yang dijelaskan dalam tabel 2.7.

Tabel 2.7 Perbandingan Jumlah Maksimum Memori

Size	MobileNetV1	MobileNetV2	ShuffleNet (2x.g=3)
$112\ x\ 112$	64/1600	16/400	32/800
$56\ x\ 56$	128/800	32/200	48/300
$28\ x28$	256/400	64/100	400/600K
$14\ x14$	512/200	160/62	800/310
$7\ x7$	1024/199	320/32	1600/156
$1\ x\ 1$	1024/2	1280/2	1600/3
max	1600K	400K	600K

Jumlah maksimum channel atau memori (dalam Kb) perlu diwujudkan pada setiap resolusi spasial untuk arsitektur yang berbeda. Google AI mengasumsikan 16-bit float (tipe data) untuk aktivasi. Untuk ShuffleNet, tim Google AI menggunakan 2x, g = 3 yang cocok dengan kinerja MobileNetV1 dan MobileNetV2. Untuk lapisan pertama MobileNetV2 dan ShuffleNet dapat menggunakan trik yang dapat mengurangi kebutuhan memori. Meskipun ShuffleNet menggunakan bottleneck di tempat lain, tensor nonbottleneck masih perlu diimplementasikan karena adanya kehadiran shortcuts (pintasan) antara tensor non-bottleneck.

2.22 Confusion Matrix

Confusion matrix merupakan salah satu teknik yang dapat digunakan untuk mengukur kinerja suatu model khususnya klasifikasi (supervised learning) pada machine learning. Confusion matrix disebut juga sebagai error matrix. Pada dasarnya confusion matrix memberikan informasi perbandingan hasil klasifikasi yang dilakukan oleh sistem (model) dengan hasil klasifikasi sebenarnya. Confusion matrix berbentuk tabel matrix yang menggambarkan kinerja model klasifikasi pada serangkaian data uji yang nilai sebenarnya diketahui. Gambar berikut merupakan confusion matrix dengan 4 kombinassi nilai prediksi dan nilai aktual yang berbeda.

Terdapat 4 istilah yang merepresentasikan hasil proses klasifikasi confusion matrix, yaitu True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN). Supaya lebih mudah dipahami, penulis mengimplementasikan dengan contoh kasus sederhana untuk memprediksi seseorang menderita acne inflammatory atau acne comedonal.

1. True Positive (TP) : Jumlah observasi positif yang tepat prediksi
2. True Negative (TN) : Jumlah observasi negatif yang tepat prediksi
3. False Positive (FP) : Jumlah observasi positif yang salah diprediksi sebagai negatif
4. False Negative (FN) : Jumlah observasi negative yang salah diprediksi sebagai positif

Metode ini sangat berguna karena confusion matrix akan memberi tahu seberapa baik model yang kita buat, Secara khusus confusion matrix juga memberikan informasi tentang TP, FP, TN, dan FN. Hal ini berguna karena hasil dari klasifikasi umumnya tidak dapat diekspresikan dengan baik dalam satu angka saja. Berikut adalah beberapa manfaat dari confusion matrix:

1. Menunjukkan bagaimana model ketika membuat prediksi
2. Tidak hanya memberi informasi tentang kesalahan yang dibuat oleh model tetapi juga jenis kesalahan yang dibuat
3. Setiap kolom dari confusion matrix merepresentasikan instance dari kelas prediksi. Setiap baris dari confusion matrix mewakili instance dari kelas aktual

2.22.1 Accuracy

Accuracy menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar. Accuracy dapat dikatakan sebagai rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan demikian, accuracy merupakan tingkat kedekatan nilai prediksi dengan nilai aktual (sebenarnya). Nilai accuracy dapat diperoleh dengan persamaan berikut.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(2.15)}$$

2.22.2 Precision (Nilai Prediksi Positif)

Precision menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Maka precision merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif, Dari semua kelas positif yang telah diprediksi dengan benar, berapa banyak data yang benar-benar positif. Nilai precision dapat diperoleh dengan persamaan berikut.

$$Precision = \frac{TP}{TP + FP} \quad \textbf{(2.16)}$$

2.22.3 Recall (Tingkat Positif Kebenaran)

Recall menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Maka, recall merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif, Nilai recall dapat diperoleh dengan persamaan berikut

$$Recall = \frac{TP}{TP + FN} \quad \textbf{(2.17)}$$

2.22.4 F1-Score

Matriks ini mewakili rata-rata harmonic antara nilai recall dan precision. Nilai F1-Score dapat diperoleh dengan persamaan berikut

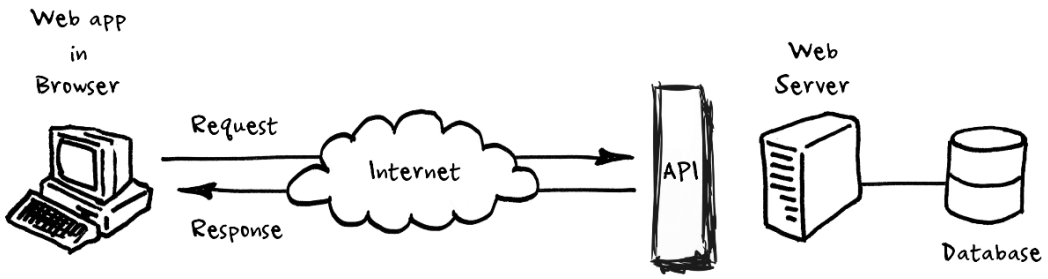
Keterangan pada persamaan F1-Score:

$$F1 - Score = \frac{2 * p * r}{p + r} \quad \textbf{(2.18)}$$

1. P = Precision
2. R = Recall

2.23 API (Application Programming Interface)

API adalah singkatan dari Application Programming Interface yang digunakan perangkat lunak untuk mengakses data, perangkat lunak server atau aplikasi lain dan telah ada selama beberapa waktu.



Gambar 2.27 Ilustrasi Alur Kerja Application Programming Interface

Dalam istilah awam, ini adalah perantara perangkat lunak yang memungkinkan dua aplikasi untuk berbicara satu sama lain seperti pada gambar 2.27. Tugas pokok API adalah sebagai penerjemah antara dua orang yang tidak berbicara bahasa yang sama tetapi dapat berkomunikasi menggunakan perantara. API sangat serbaguna dan dapat digunakan pada sistem berbasis web, sistem operasi, sistem basis data, dan perangkat keras komputer. Pengembang menggunakan API untuk membuat pekerjaan mereka lebih efisien dengan menggunakan kembali kode dari sebelumnya dan hanya mengubah bagian yang relevan dengan proses yang ingin mereka tingkatkan. API yang baik memudahkan pembuatan program karena blok penyusunnya sudah ada. API menggunakan protokol yang ditentukan untuk memungkinkan pengembang membangun, menghubungkan, dan mengintegrasikan aplikasi dengan cepat dan dalam skala besar.

2.24 Flask

Flask adalah kerangka kerja (framework) web berbasis Python. Flask memberi pengguna alat, pustaka (library), dan teknologi yang memungkinkan pengguna membangun aplikasi web dan membuat API.

Flask adalah bagian dari kategori kerangka mikro. Kerangka kerja mikro biasanya kerangka kerja dengan sedikit atau tanpa ketergantungan ke perpustakaan eksternal. Ini memiliki pro dan kontra. Kelebihannya adalah kerangka kerjanya ringan, ada sedikit ketergantungan untuk memperbarui dan mengawasi bug keamanan, kekurangannya adalah terkadang pengguna harus melakukan lebih banyak pekerjaan sendiri atau menambah sendiri daftar dependensi dengan menambahkan plugin.

2.25 Postman

Postman ini merupakan tool wajib bagi para developer yang berkutat pada pembuatan API, fungsi utama postman ini adalah sebagai GUI API Caller namun sekarang postman juga menyadiakan fitur lain yaitu Sharing Collection API for Documentation (free), Testing API (free), Realtime Collaboration Team (paid)

2.26 Flowchart

Flowchart atau diagram alur adalah representasi grafis dari langkah-langkah. Sejarah flowchart berasal dari pengembangan ilmu komputer sebagai alat untuk mewakili algoritma dan logika pemrograman tetapi telah diperluas untuk digunakan dalam semua jenis proses lainnya. Saat ini, diagram alur memainkan peran yang sangat penting dalam menampilkan informasi dan membantu penalaran. Flowchart membantu pengembang dalam memvisualisasikan proses yang kompleks, atau membuat eksplisit struktur masalah dan tugas. Flowchart juga dapat digunakan untuk menentukan proses atau proyek yang akan dilaksanakan. Adapun macam macam flowchart sebagai berikut :

1. Flowchart Sistem

   Flowchart sistem menampilkan bagaimana data mengalir di dalam sistem dan eksekusi keputusan untuk mengontrol peristiwa. Flowchart sistem

biasanya menggunakan simbol yang terhubung untuk menggambarkan alur yang terjadi pada berbagai titik data di dalam suatu sistem.

2. Flowchart Dokumen

   Flowchart dokumen menampilkan dokumen elektronik di antara berbagai unit bisnis. Flowchart dokumen berperan sebagai alat yang membantu analis untuk memahami, menganalisis, mendokumentasikan, dan meningkatkan berbagai proses kerja di dalam suatu organisasi. Flowchart dokumen dibaca dari kiri ke kanan karena menunjukkan aliran dokumen melalui berbagai unit bisnis.

3. Flowchart Data

   Flowchart data adalah langkah awal yang membantu dalam membuat ikhtisar sistem tanpa mengungkapkan detail grafis apa pun. Flowchart data menyoroti saluran yang dilalui data dalam melakukan transmisi di dalam sebuah rangkaian sistem.

4. Flowchart Program

   Flowchart program dibuat dengan simbol grafis standar yang mewakili urutan instruksi kode. Flowchart program digunakan untuk menggambarkan cara kerja internal sistem komputerisasi modern. Empat simbol dasar digunakan untuk membangun diagram alur program modern. Ini termasuk awal, proses, keputusan, dan akhir. Flowchart program membantu pengembang perangkat lunak dan arsitek untuk menemukan bug di dalam potongan kode komputer. Perangkat ini membantu meningkatkan efisiensi pengkodean dan mengomunikasikan logika penting sistem kepada peninjau dan pengembang.

BAB 3

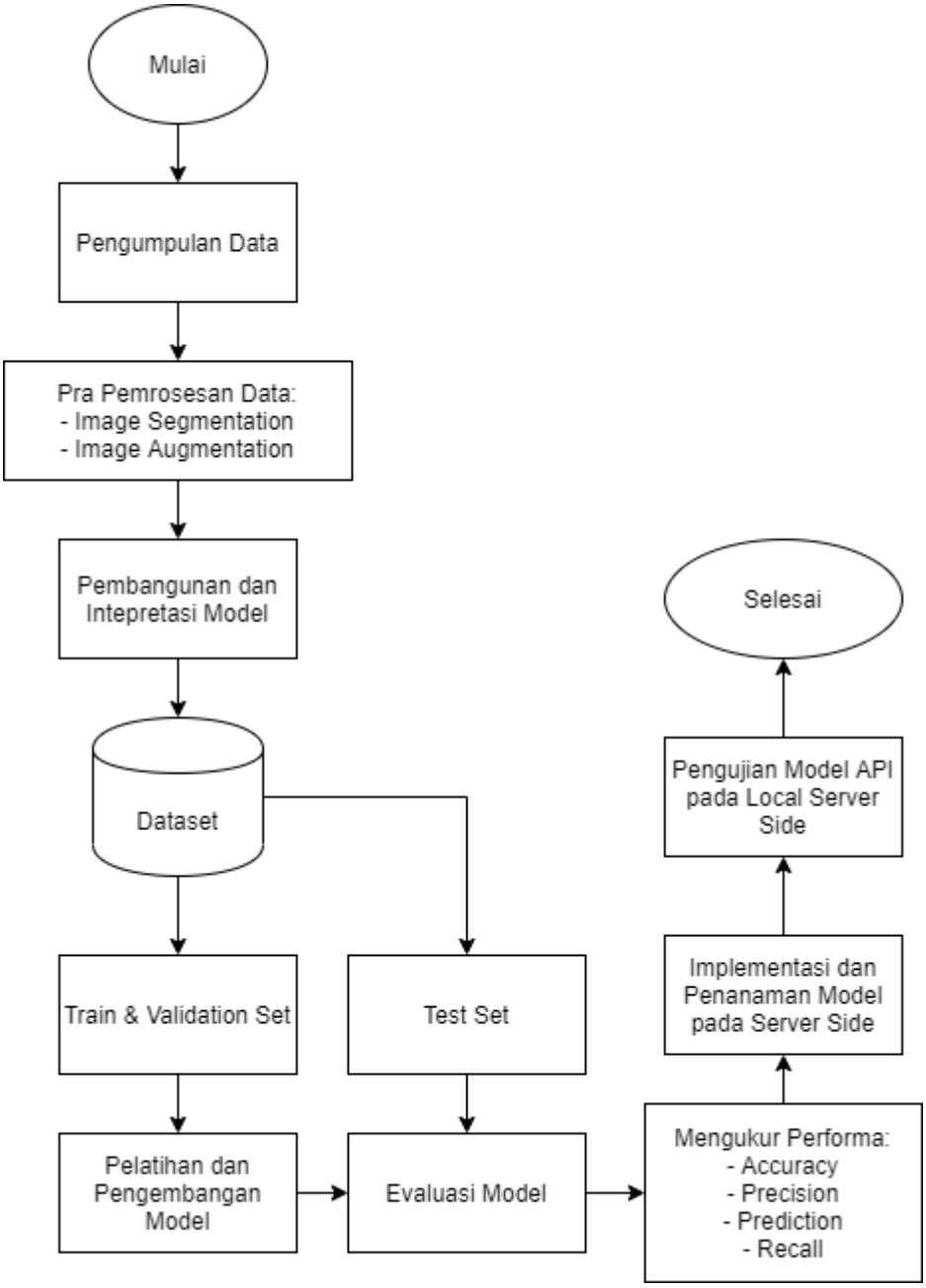
PERANCANGAN DAN IMPLEMENTASI

Pada bab 3 yakni Perancangan dan Implementasi terdiri atas beberapa tahapan proses. Tahapan awal dimulai dari gambaran umum yang menjelaskan perihal skema alur perancangan dan pelatihan model yang berguna sebagai tolak ukur pemrogram dalam mengimplementasikan kode secara baik dan terstruktur. Tahap kedua membahas pengumpulan data Acne yang dikumpulkan dari beberapa sumber penelitian kedokteran khususnya di bidang spesialisasi dermatologi, diantara lain Dermnet. Set data terdiri dari 700 data gambar jerawat comedonal dan inflammatory yang kemudian dibagi dua kembali dengan jumlah yang sama rata sebanyak 350 data per jenis jerawat.

Setelah mengumpulkan set data, tahap ketiga adalah pemrosesan data dengan memanfaatkan proses image segmentation dan image augmentation. Pada tahap keempat dilakukan pelatihan data secara berulang menggunakan arsitektur dan pre-trained model MobileNetV2. Pengujian sample akan menampilkan hasil model data gambar apakah terklasifikasi sebagai jerawat comedonal atau jerawat inflammatory sehingga pada hasil pengujian akan dilakukan analisis dan evaluasi akurasi model. Dan langkah terakhir adalah pengimplementasian model pada server side menggunakan microframework flask. Flask akan menerjemahkan input berupa gambar menjadi prediksi yang diwakilkan oleh besaran persentase terhadap jenis jerawat tertentu. Pembuatan server side digunakan untuk memproduksi model API yang kedepannya mampu digunakan untuk segala platform, baik web, desktop, maupun mobile.

3.1 Gambaran Umum

Secara garis besar, model klasifikasi jenis jerawat melakukan sejumlah tahapan utama seperti pada bagan berikut:



Gambar 3.1 Tahapan Pengembangan Model Klasifikasi Jenis Jerawat

Tahapan awal yang dilakukan penulis dalam penelitian ini adalah mengumpulkan data gambar jerawat inflammatory dan jerawat comedonal dari internet, seperti dermnet.com dan google image. Setiap citra yang telah dikumpulkan akan dikategorikan menjadi beberapa bagian, yaitu data latih, data validasi, dan data uji. Langkah berikutnya yang dilakukan penulis adalah

melakukan preprocessing data gambar menggunakan teknik Image Augmentation dan Image Segmentation.

Teknik Image Augmentation digunakan untuk memperbanyak jumlah data khususnya pada data latih, pada proses ini juga dilakukan normalisasi data pixel gambar, dan melakukan resize gambar menjadi ukuran 224 x 224 x 3 pixels. Sedangkan teknik Image Segmentation dilakukan untuk mengurangi kompleksitas citra sehingga mempermudah model menganalisis citra tersebut. Pembuatan model machine learning akan dibuat menggunakan framework Tensorflow 2.0 yang akan mengimplementasikan salah satu jenis arsitektur Convolutional Neural Network (CNN) yaitu MobileNetV2. Dalam tahapan ini juga akan memanfaatkan teknik transfer learning menggunakan pre-train weight dari imagenet dataset agar proses pelatihan model machine learning tidak memakan waktu lama.

Evaluasi adalah tahapan ketika model diuji dengan data yang belum dikenali, dalam penelitian ini adalah data uji yang telah disiapkan sebelumnya. Pada proses evaluasi akan dilihat performa dari model apakah sudah optimal atau belum, jika belum maka akan dilakukan proses pelatihan ulang hingga hasil prediksi dari model cukup optimal. Selanjutnya ketika model machine learning sudah dianggap optimal maka model akan diekspor dengan ekstensi .h5 untuk memudahkan tahap *deployment*.

Implementasi dan penanaman model akan mengonsumsi model machine learning dalam file dengan format .h5 yang kemudian akan dilakukan pemrosesan ulang dalam microframework flask yakni sebuah kerangka perangkat lunak berbasis bahasa pemrograman python. Pemrosesan data akan dilakukan dari mulai memasukan data gambar hingga mendapatkan hasil berupa persentase dari jenis jerawat yang berhasil di prediksi.

3.2 Perancangan

Tahap ini merupakan tahap dimana penulis melakukan pemilihan modul dan library yang akan digunakan, pengumpulan dataset, membuat rancangan alur preprocessing data, dan skema pelatihan model machine learning.

3.2.1 Spesifikasi Perangkat

Spesifikasi komputer yang digunakan penulis untuk menjalankan program ini adalah sebagai berikut :

Kebutuhan Perangkat Keras (Hardware) :

1. Laptop Asus Vivobook S14 S410UN.
2. Intel® Core™ i5-8250U.
3. 16 GB RAM, 1 TB HDD, 128 GB SSD.
4. VGA Nvidia GeForce MX150 4 GB RAM.

Kebutuhan Perangkat Lunak (Software):

1. PyCharm Community Edition 2017.1.5.
2. Windows 10 64-bit.

Persyaratan Minimum Perangkat Platform (Android):

1. Android 7.0 Nougat
2. 1.5 GB RAM
3. Processor Snapdragon 450
4. Tersisa 100 MB penyimpanan internal

3.2.2 Pemilihan Library

Pada tahap ini penulis memanfaatkan sejumlah library atau modul yang disediakan oleh python.

```
from google.colab import drive
import os
```

Gambar 3.2 Import Library Google Colab dan OS

Library google.colab digunakan untuk menghubungkan google colab dengan google drive. Pada penelitian ini seluruh dataset disimpan di dalam google drive. Selain itu google.colab berfungsi untuk melakukan pengunduhan dataset dari google drive ke google colab sehingga data dapat dilatih pada notebook

colab. Library os digunakan untuk berinteraksi dengan sistem operasi. OS berada di bawah modul utilitas standar Python. Library ini menyediakan portabilitas untuk seluruh fungsionalitas yang bergantung pada sistem operasi.

```
import glob as gb
```

Gambar 3.3 Import Library Glob

Glob pada gambar 3.3 adalah istilah umum yang digunakan untuk mendefinisikan teknik untuk mencocokkan pola yang ditentukan menurut aturan shell Unix. Glob (kependekan dari global) digunakan untuk mengembalikan semua jalur file yang cocok dengan pola tertentu. Kita dapat menggunakan glob untuk mencari pola file tertentu. Dalam penelitian ini glob digunakan untuk mencari file yang nama filenya cocok dengan pola tertentu dengan menggunakan karakter wildcard. Pola glob menentukan set nama file dengan karakter wildcard. Misalnya, perintah shell Unix Bash mv *.jpg jointphotographicgroup/moves (mv) semua file dengan nama berakhiran .jpg akan cocok dengan semua file berkestensi .jpg (data gambar)

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import seaborn as sns
```

Gambar 3.4 Import Library Matplotlib untuk Visualisasi Data

Pada gambar 3.4 terdapat beberapa library matplotlib beserta sub-library nya yang digunakan untuk proses visualisasi data. Library pertama adalah matplotlib.image yaitu sebuah modul gambar yang mendukung pemuatan, penskalaan, dan pengoperasian tampilan gambar. Library kedua adalah matplotlib.pyplot yaitu kumpulan modul yang dapat membuat area pemetaan grafik dan menghiasi grafik dengan label sehingga lebih mudah dipahami dalam segi bisnis. Pada matplotlib.pyplot terdapat beberapa jenis pemetaan grafik yang dapat dipilih, diantaranya grafik garis, grafik pie, grafik batang, dan grafik

sebaran. Sedangkan seaborn adalah library yang sebagian besar digunakan untuk membuat grafik statistik. Seaborn adalah modul visualisasi data yang dibangun di atas matplotlib dan terintegrasi dengan struktur data pandas di Python. Visualisasi adalah bagian sentral dari Seaborn yang membantu dalam eksplorasi dan pemahaman data.

```
import numpy as np
import pandas as pd
import datetime
```

Gambar 3.5 Import Library Numpy, Pandas, dan Datetime

Pada gambar 3.5 terdapat library yang menangani pengolahan angka. Library pertama adalah numpy yaitu library yang dapat digunakan untuk melakukan berbagai macam operasi matematika pada array. Numpy menyediakan struktur data berbasis python yang menjamin perhitungan array dan matriks yang efisien dan numpy termasuk salah satu library dengan kemampuan fungsi matematika tingkat tinggi. Library kedua adalah pandas yaitu library yang digunakan untuk analisis data. Pandas dapat mengimpor data dari berbagai format file seperti nilai yang dipisahkan koma, JSON, SQL, Microsoft Excel. Pandas memungkinkan berbagai operasi manipulasi data seperti penggabungan, pembentukan kembali (reshaping), pemilihan, serta pembersihan data (cleaning), dan fitur data wrangling untuk mengurangi perselisihan pada data. Modul datetime memasok kelas untuk memanipulasi tanggal dan waktu. Fokus implementasinya adalah pada ekstraksi atribut yang efisien untuk pemformatan dan manipulasi keluaran yang berhubungan dengan kalender.

```
from sklearn.metrics import classification_report, confusion_matrix
```

Gambar 3.6 Import Library Sklearn untuk Evaluasi Model

Pada gambar 3.6 merupakan library sklearn.metrics yang berfungsi untuk mengevaluasi kinerja klasifikasi pada model yang telah dilatih. Terdapat dua

fungsi evaluasi yang digunakan oleh penulis yakni classification_report dan confusion_matrix. Classification_report digunakan untuk mengukur kualitas prediksi dari algoritma klasifikasi. Laporan ini menunjukkan presisi, recall, dan f1-score klasifikasi utama berdasarkan per-kelas. Metrik dihitung dengan menggunakan positif benar dan salah, negatif benar dan salah. Sedangkan fungsi confusion_matrix digunakan untuk mengevaluasi keakuratan klasifikasi.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import Callback, ReduceLROnPlateau, TensorBoard, EarlyStopping, ModelCheckpoint
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, GlobalAveragePooling2D
```

Gambar 3.7 Import Library Tensorflow untuk Penugasan *Deep Learning*

Pada gambar 3.7 terdapat sejumlah kelas yang merupakan bagian dari library tensorflow dan keras untuk pengoperasian dan pelatihan model pembelajaran mendalam atau deep learning. Library pertama adalah ImageDataGenerator yaitu sebuah kelas dari Keras yang menyediakan cara cepat dan mudah untuk menambah jumlah gambar. ImageDataGenerator menyediakan sejumlah teknik augmentasi yang berbeda seperti standardisasi, rotasi, shift, flips, perubahan kecerahan, dan lain-lain. Kelas ImageDataGenerator memastikan bahwa model menerima variasi baru dari gambar di setiap epoch (perulangan). Tetapi kelas ini hanya mengembalikan gambar yang diubah dan tidak menambahkannya ke kumpulan gambar asli.

Kelas kedua adalah Callback yang merupakan seperangkat fungsi yang diterapkan pada tahapan tertentu dari prosedur pelatihan. Fungsi callback diantara lain adalah menghentikan pelatihan saat proses telah mencapai skor akurasi/kerugian tertentu, menyimpan model sebagai *checkpoint* atau pos pemeriksaan setelah setiap epoch yang berhasil, menyesuaikan tingkat pembelajaran dari waktu ke waktu, dan lain sebagainya. Sedangkan kelas ketiga yaitu earlystopping digunakan untuk menentukan ukuran kinerja yang akan dipantau, pemicunya, dan sekali dipicu, earlystopping akan menghentikan proses pelatihan. Callback EarlyStopping dikonfigurasi ketika dipakai melalui argumen.

Kelas keempat yaitu ReduceLROnPlateau digunakan untuk menyesuaikan kecepatan pembelajaran ketika kinerja model yang stabil terdeteksi, misalnya. tidak ada perubahan untuk sejumlah periode pelatihan tertentu. Callback ini dirancang untuk mengurangi kecepatan pembelajaran setelah model berhenti meningkat dengan harapan untuk menyempurnakan bobot model. Kelas kelima adalah TensorBoard yang merupakan alat visualisasi yang disediakan satu paket bersama TensorFlow. Callback ini mencatat beberapa peristiwa termasuk grafik ringkasan metrik dan visualisasi grafik pelatihan. Sedangkan kelas keenam adalah ModelCheckpoint yang merupakan callback untuk menyimpan bobot model atau seluruh model pada frekuensi tertentu. ModelCheckpoint memungkinkan kita untuk menentukan kuantitas yang akan dipantau, seperti memantau nilai loss atau accuracy pada set data pelatihan atau validasi.

Kelas ketujuh adalah MobileNetV2. MobileNetV2 sangat mirip dengan MobileNet versi perdana, yang membedakan yakni MobileNetV2 menggunakan inverted residual blok dengan fitur bottlenecking. V2 memiliki jumlah parameter yang jauh lebih rendah daripada MobileNet asli. MobileNets mendukung ukuran input apa pun bahkan yang lebih besar dari 32 x 32. Dengan demikian ukuran gambar yang lebih besar akan menghasilkan kinerja yang lebih presisi dan detail.

Kelas kedelapan adalah Models. Kelas ini merupakan turunan dari library tensorflow.keras.models yang menyediakan model Sequential. Model Sequential merupakan pilihan yang tepat untuk tumpukan lapisan (layer) dasar di mana setiap lapisan memiliki tepat satu tensor input dan satu tensor output. Namun model Sequential tidak sesuai jika model memiliki banyak input atau beberapa output.

Kelas kesembilan adalah Optimizer Adam. Adam adalah algoritma pengoptimalan alternatif dari Stochastic Gradient Descent untuk melatih model pembelajaran mendalam. Adam menggabungkan properti terbaik dari algoritma AdaGrad dan RMSProp untuk menyediakan algoritma optimasi yang dapat menangani sparse gradients pada masalah noise. Spares gradients menunjukkan sebuah kondisi dimana jaringan tidak menerima sinyal yang cukup kuat untuk menyesuaikan bobotnya.

Kelas kesepuluh adalah Dense Layer. Dense layer adalah lapisan jaringan saraf yang terhubung secara mendalam. Lapisan padat atau dense layer adalah lapisan yang paling umum dan sering digunakan karena sering digunakan untuk melakukan menangani operasi pada input dan hingga mengembalikan output. Kelas kesebelas adalah Dropout layer. Dropout layer secara acak menetapkan unit input ke 0 dengan frekuensi laju pada setiap langkah selama waktu pelatihan yang membantu mencegah overfitting. Lapisan Dropout hanya berlaku ketika pelatihan diatur ke True sehingga tidak ada nilai yang dibuang selama inferensi. Kelas keduabelas adalah BatchNormalization layer. Lapisan ini akan mengubah input menjadi standar, yang berarti lapisan akan memiliki rata-rata nol dan standar deviasi bernilai satu. Selama pelatihan, lapisan akan melacak statistik untuk setiap variabel input dan menggunakannya untuk standarisasi data. Kelas ketigabelas adalah GlobalAveragePooling2D yakni operasi yang menghitung output rata-rata dari setiap peta fitur di lapisan sebelumnya. Operasi yang cukup sederhana ini mengurangi data secara signifikan dan mempersiapkan model untuk lapisan klasifikasi akhir.

3.2.3 Pengumpulan Data

Dataset yang digunakan dalam penelitian ini adalah citra jerawat komedonal dan citra jerawat inflammatory yang didapat dari internet, seperti Dermnet. Penelitian ini menggunakan data sekunder disebabkan keterbatasan waktu dan kondisi pandemi saat ini yang tidak memungkinkan untuk mengumpulkan orang dalam jumlah banyak secara tatap muka langsung. Jumlah citra yang terdapat pada dataset adalah 700 gambar dengan format .jpg. Jumlah citra jerawat inflammatory adalah 350 gambar dengan ukuran yang bervariasi, begitu juga dengan jumlah citra jerawat komedonal yang memiliki jumlah yang sama.

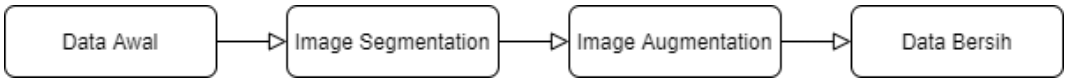
a. Acne Inflammatory b. Acne Comedonal

Gambar 3.8 Dataset Jenis Jerawat

Pada gambar dataset 3.8, Terdapat perbedaan signifikan antara citra jenis jerawat inflammatory dengan citra jenis jerawat komedonal. Jenis jerawat inflammatory memiliki ciri utama yaitu cenderung berwarna merah dengan ukuran jerawat yang tampak besar. Berbeda dengan jenis jerawat inflammatory, jenis jerawat komedonal memiliki ukuran yang lebih kecil dan berwarna hitam atau putih. Hal ini dikarenakan perbedaan penyebab timbulnya jerawat sehingga bentuk jerawatnya pun berbeda. Dataset yang telah dikumpulkan lalu dibagi menjadi beberapa kategori, yaitu data latih, data validasi, dan data uji. Pembagian dataset ini dilakukan dengan rasio 60:20:20, maka total data latih berjumlah 420 citra, data validasi berjumlah 140 citra, dan data uji berjumlah 140 citra untuk 2 jenis kategori jerawat.

3.2.4 Preprocessing Data

Preprocessing data merupakan sebuah tahapan dimana dilakukan suatu pemrosesan pada dataset sebelum data tersebut memasuki proses pelatihan model. Hal ini dilakukan untuk mempermudah proses pelatihan model machine learning. Pada tahapan ini, penulis melakukan beberapa metode pre-processing data yang biasa digunakan untuk data citra.



Gambar 3.9 Alur Preprocessing Data Gambar

Gambar 3.9. merupakan alur secara singkat bagaimana tahapan preprocessing dilakukan pada penelitian ini. Tahapan pertama yaitu dataset akan dilakukan preprocessing menggunakan teknik image segmentation berbasis metode K-means clustering. K-Means Clustering dipilih karena metode ini sederhana dan efisien. Metode ini akan mengelompokan jenis data menjadi sejumlah klaster yang telah didefinisikan. Untuk metode image segmentation, kluster merujuk pada jumlah warna citra yang berbeda. Pada penelitian ini, jumlah klaster warna yang akan digunakan berjumlah 4 klaster.

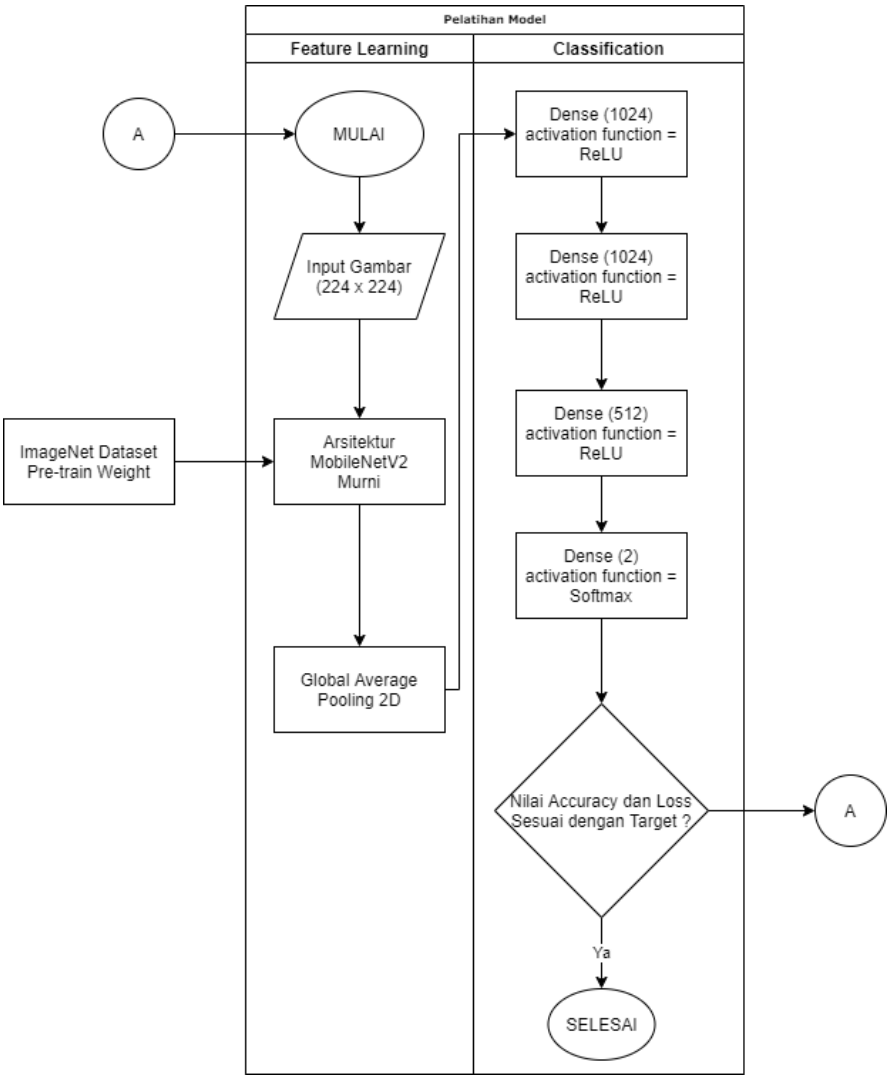
Setelah dilakukan image segmentation menggunakan metode K-means clustering, data gambar yang telah diolah akan dilakukan metode preprocessing selanjutnya yaitu image augmentation. Image augmentation adalah sebuah teknik yang dapat meningkatkan jumlah data latih secara artifisial dengan menghasilkan banyak varian realistis dari setiap contoh data latih. Pada metode ini, teknik yang digunakan adalah rescale, rotate, shift, shear, zoom, dan flip. Data citra latih akan dilakukan rescale dengan cara membagi tiap-tiap pixel dengan angka 255. Selanjutnya data latih akan dilakukan rotate sebesar 25° . Setelah dilakukan rotate, data tersebut akan dilakukan shift sebesar 0.15 pixel. Selanjutnya data juga akan dilakukan shear sebanyak 0.15° dan dilakukan zooming sebesar 0.5 s.d 1.1 kali. Terakhir, data akan dilakukan flip secara vertikal dan horizontal.

3.2.5 Skema Pelatihan Model

Pelatihan sebuah model Classification Neural Network umumnya terbagi menjadi 2 bagian, yaitu feature learning dan classification. Input citra pada penelitian kali ini berupa citra yang berukuran 224 x 224 x 3. Angka 3 pada ukuran gambar merepresentasikan bahwa citra tersebut memiliki 3 channel warna, yaitu red, green, dan blue. Pada tahapan feature learning dalam penelitian ini akan memanfaatkan teknik transfer learning menggunakan pre-train weight dari ImageNet dataset.

Transfer learning adalah suatu teknik atau metode yang memanfaatkan model yang telah dilatih terhadap suatu dataset untuk menyelesaikan permasalahan lain yang serupa dengan cara menggunakannya sebagai titik awal,

memodifikasi, dan memperbaharui parameter dari suatu model sesuai dengan dataset baru, dalam hal ini dataset yang telah dikumpulkan sebelumnya. Metode ini digunakan dikarenakan jumlah citra dalam dataset yang digunakan dalam penelitian ini tergolong kecil, yaitu hanya 700 citra untuk 2 jenis jerawat. Model yang dipilih dalam penelitian ini adalah model dengan arsitektur MobileNetV2 yang cocok digunakan pada mobile device dikarenakan ukuran model yang tidak terlalu besar tanpa mengurangi performa model. Proses alur mengenai bagaimana tahapan feature learning dalam penelitian ini bekerja dapat dilihat pada Gambar 3.10



Gambar 3.10 Flowchart Pelatihan Model

Tahapan classification adalah sebuah tahap dimana hasil dari feature learning akan digunakan untuk proses klasifikasi citra jerawat. Pada tahapan ini dirancang sebuah custom fully connected layer yang sesuai dengan kebutuhan dataset. Fully connected layer pada tahapan ini dimulai dengan overlapping layer Dense dengan jumlah parameter sebanyak 1024 dengan activation function ReLU. Dilanjutkan dengan layer Dense dengan jumlah parameter sebanyak 512 dengan activation function ReLU dan diakhiri dengan sebuah output layer bertipe Dense dengan 2 parameter dengan activation function softmax. Fungsi softmax dipilih agar output prediksi dari model ini berupa persentase ketepatan prediksi untuk setiap jenis jerawat. Tahapan pelatihan ini akan dilakukan secara berulang kali dan akan dihentikan ketika akurasi dan loss dari model sudah optimal. Target minimum akurasi dari model adalah 85% dengan loss kurang dari 0.5.

3.3 Implementasi

Implementasi adalah suatu tahapan dimana penulis merealisasikan rancangan yang telah dibuat sebelumnya ke dalam bentuk kode menggunakan bahasa pemrograman Python dan dibantu dengan *library* Tensorflow. Bahasa pemrograman Python dipilih karena merupakan bahasa pemrograman *high level* dan cenderung mudah dimengerti oleh manusia. Bahasa ini juga dipilih karena termasuk salah satu bahasa yang populer digunakan dalam mengimplementasikan algoritma deep learning dengan bantuan *library* Tensorflow. Tensorflow merupakan salah satu *library* deep learning yang populer saat ini. Hal ini dikarenakan library ini memudahkan implementasi algoritma deep learning. Dalam tahapan ini akan dibagi menjadi beberapa sub bab yaitu preprocessing, pembuatan model, dan pelatihan model.

3.3.1 Preprocessing

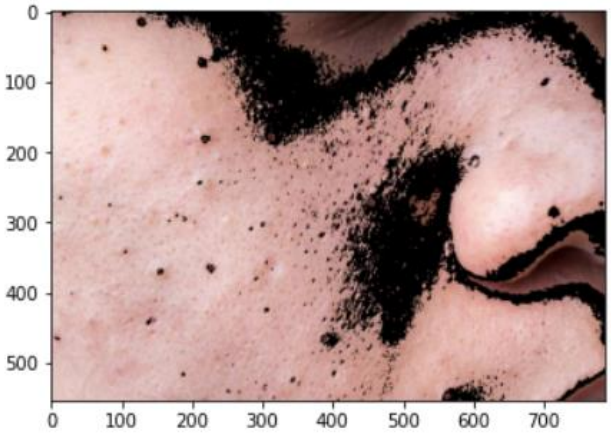
Implementasi tahapan preprocessing akan dilakukan sesuai dengan rancangan yang telah dibuat pada tahapan perancangan. Tahapan ini akan mengimplementasikan teknik image segmentation dan image augmentation ke dalam bentuk pengkodean python. Teknik image segmentation berbasis metode K-means Clustering diimplementasikan ke dalam bentuk kode dengan bantuan

library OpenCV. Hal ini dilakukan karena *library* OpenCV telah menyediakan fungsi untuk menjalankan algoritma K-means Clustering. Implementasi kode dari teknik image segmentation yang digunakan dalam penelitian ini dapat dilihat pada Gambar 3.11

```
def img_segmentation(img, k=4):
  pixel_values = img.reshape((-1, 3))
  pixel_values = np.float32(pixel_values)

  criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

  _, labels, (centers) = cv2.kmeans(pixel_values, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
  centers=np.uint8(centers)
  labels = labels.flatten()
  segmented_img = centers[labels.flatten()]
  segmented_img = segmented_img.reshape(img.shape)
  masked_img =  np.copy(img)
  masked_img = masked_img.reshape((-1, 3))
  cluster = 2
  masked_img[labels==cluster] = [0, 0, 0]
  masked_img = masked_img.reshape(img.shape)

  return masked_img
```

Gambar 3.11 Kode Teknik Image Segmentation

Pada Gambar 3.11. merupakan potongan kode implementasi teknik image segmentation pada penelitian ini. Fungsi img_segmentation() memiliki 2 buah parameter yaitu img dan k. Parameter img dalam fungsi tersebut adalah parameter yang merujuk pada citra yang ingin diproses. Sedangkan parameter k dalam fungsi tersebut menyatakan berapa banyak klaster warna yang ingin diproses. Dalam penelitian ini, nilai k yang dipakai adalah 4. Contoh citra hasil proses image segmentation dapat dilihat pada Gambar 3.12.



Gambar 3.12 Output Citra dari Implementasi Image Segmentation

Teknik Image Augmentation diimplementasikan ke dalam bentuk kode dengan bantuan kelas ImageDataGenerator yang telah disediakan oleh library Tensorflow. Seperti yang telah dijelaskan pada tahap perancangan, terdapat beberapa teknik Image Augmentation yang digunakan dalam penelitian ini seperti rescale, rotate, shift, shear, zoom, dan flip. Implementasi kode dari teknik ini dapat dilihat pada Gambar 3.13.

```
# Datagen untuk validation set dan train set
datagen = ImageDataGenerator(preprocessing_function=img_segmentation,
                             rescale = 1./255,
                             rotation_range = 25,
                             zoom_range = [0.5, 1.1],
                             width_shift_range=0.15,
                             shear_range= 0.15,
                             height_shift_range=0.15,
                             horizontal_flip=True,
                             vertical_flip= True,
                             fill_mode='nearest',
                             )

# ImageDataGenerator untuk test set
val_datagen = ImageDataGenerator(preprocessing_function=img_segmentation,
    rescale=1./255)

test_datagen = ImageDataGenerator(rescale=1./255)
```

Gambar 3.13 Kode Teknik Image Augmentation

Pada Gambar 3.13. di atas terdapat beberapa parameter dari kelas ImageDataGenerator yang digunakan dalam penelitian ini. Parameter ini merujuk pada teknik Image Augmentation yang digunakan dalam penelitian ini. Parameter preprocessing_function=img_segmentation bertujuan untuk agar kelas ImageDataGenerator menjalankan teknik image segmentation yang telah dibuat fungsinya seperti pada Gambar 3.10. Parameter rescale = 1./255 bertujuan untuk menormalisasi setiap pixel yang terdapat dalam seluruh citra pada data latih menjadi rentang 0 s.d 1. Rotation_range = 25 bertujuan untuk merotasi citra secara acak. Zoom_range = [0.5, 1.1] bertujuan untuk memperbesar citra sebesar 0,5 kali dan 1,1 kali secara acak. Width_shift_range = 0.15 bertujuan untuk menggeser citra secara horizontal sebesar 0.15. Height_shift_range = 0.15 bertujuan untuk menggeser citra secara vertikal sebesar 0.15. Shear_range = 0.15 bertujuan untuk membuat gambar terdistorsi sepanjang sumbu x sebesar 0.15

derajat. Horizontal_flip=True bertujuan untuk memutar gambar secara horizontal. Vertical_flip=True bertujuan untuk memutar gambar secara vertikal.

3.3.2 Pengembangan Model

Tahapan ini adalah tahapan dimana penulis mengimplementasikan rancangan model Convolutional Neural Network(CNN) berbasis arsitektur MobileNetV2 menggunakan bahasa pemrograman python dengan bantuan library Tensorflow. Dalam tahapan pembuatan model terbagi menjadi 2 tahapan, yaitu pembuatan fungsi callback dan pembuatan model. Pembuatan fungsi callback ditujukan untuk mengatasi permasalahan overfit. Overfit adalah sebuah kondisi ketika model memiliki prediksi yang baik untuk data latih, namun memeliki prediksi yang buruk dalam data uji atau pada data yang belum pernah dipelajari sebelumnya. Terdapat beberapa fungsi callback bawaan dari tensorflow.keras yang dipakai dalam penelitian ini seperti ReduceLROnPlateau, ModelCheckpoint, dan EarlyStopping.

ReduceLROnPlateau adalah sebuah fungsi callback yang berfungsi untuk mengurangi learning rate pada saat training ketika target metric berhenti meningkat, dalam penelitian ini metric yang dipantau adalah val_loss(nilai loss pada data validasi). Nilai learning rate baru akan didapatkan dengan cara nilai learning rate lama dikali dengan nilai faktor. Nilai faktor yang digunakan dalam penelitian ini adalah 0.01. Dalam penelitian ini juga diatur nilai minimal learning rate pada fungsi ReduceLROnPlateu sebesar $1\,x\,10^{-6}$. Implementasi kode dari penggunaan beberapa fungsi callback ini dapat dilihat pada Gambar 3.14.

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.01, min_lr=1e-6)
checkpoint_cb = ModelCheckpoint("/content/gdrive/MyDrive/model/MobileNetV2/current_best_model.h5", saved_best_only=True)
early_stop_cb = EarlyStopping(patience=10,monitor='val_loss', restore_best_weights= True)

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)

# Menghapus logs dari proses run sebelumnya
!rm -rf ./logs/

callbacks =[reduce_lr, checkpoint_cb, early_stop_cb, tensorboard_callback]
```

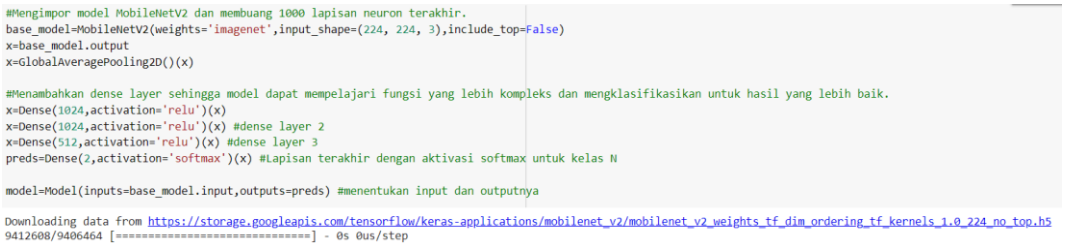
Gambar 3.14 Kode Pembuatan Callbacks

ModelCheckPoint adalah sebuah fungsi callback yang berfungsi untuk menyimpan model yang telah dilatih setiap batchnya dalam suatu direktori checkpoint. Model akan disimpan dalam format .h5, yakni format untuk menyimpan dalam bentuk multidimensional array. Dalam penelitian ini peneliti mengatur parameter saved_only_best=True yang berada dalam fungsi callback ini. Hal ini bertujuan agar fungsi callback ini hanya menyimpan model dengan nilai weight yang terbaik.

EarlyStopping adalah sebuah fungsi callback yang berfungsi untuk menghentikan proses pelatihan model ketika nilai metric yang dipantau berhenti meningkat. Metric yang dipantau pada fungsi ini sama dengan metric yang dipantau pada fungsi ReduceLROnPlateu, yakni val_loss metric. Parameter patience bertujuan mengatur batas maksimal toleransi epoch yang tidak memiliki peningkatan. Parameter restore_best_weights=True bertujuan ketika pelatihan model dihentikan oleh fungsi callback ini, nilai weight terbaik akan dikembalikan ke dalam model terbaru.

Pembuatan model adalah tahap dimana penulis mengimplementasikan rancangan model machine learning ke dalam bahasa pemrograman. Dalam mengimplementasikan rancangan arsitektur MobileNetV2, penulis menggunakan bantuan library Tensorflow.keras yang telah menyediakan kelas MobileNetV2.

MobileNetV2 adalah sebuah kelas yang berisi model rancangan machine learning berbasis arsitektur MobileNetV2. Kelas ini juga memudahkan peneliti mengimplementasikan metode transfer learning. Hal ini dikarenakan model tersebut telah dilatih sebelumnya pada dataset besar, seperti imagenet. Dalam penelitian ini ukuran citra yang dapat diterima oleh model adalah citra yang berukuran 224 x 224 x 3. Implementasi kode model machine learning berbasis arsitektur MobileNetV2 dapat dilihat pada Gambar 3.15



```
#Mengimpor model MobileNetV2 dan membuang 1000 lapisan neuron terakhir.
base_model=MobileNetV2(weights='imagenet',input_shape=(224, 224, 3),include_top=False)
x=base_model.output
x=GlobalAveragePooling2D()(x)

#Menambahkan dense layer sehingga model dapat mempelajari fungsi yang lebih kompleks dan mengklasifikasikan untuk hasil yang lebih baik.
x=Dense(1024,activation='relu')(x)
x=Dense(1024,activation='relu')(x) #dense layer 2
x=Dense(512,activation='relu')(x) #dense layer 3
preds=Dense(2,activation='softmax')(x) #Lapisan terakhir dengan aktivasi softmax untuk kelas N

model=Model(inputs=base_model.input,outputs=preds) #menentukan input dan outputnya
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9412608/9406464 [==============================] - 0s 0us/step
```

Gambar 3.15 Kode Pembuatan Model

Parameter weights=’imagenet’ pada kelas MobileNetV2 yang berada pada Gambar 3.15. bertujuan untuk memuat pretrain weight yang telah dilatih sebelumnya, yakni pretrain weight dari ImageNet dataset. Parameter input_shape=(224, 224, 3) bertujuan untuk mengatur ukuran citra yang dapat diterima oleh model tersebut. Parameter include_top = False bertujuan agar model tersebut tidak menggunakan fully connected layer yang telah disediakan oleh kelas MobileNetV2. Hal ini dikarenakan peneliti ingin menggunakan custom fully connected layer yang telah disesuaikan dengan dataset dan sesuai dengan kebutuhan penelitian.

GlobalAveragePooling2D adalah sebuah kelas yang berfungsi untuk melakukan operasi pooling dengan cara menghitung jumlah rata-rata dari hasil output model MobileNetV2 dan akan mengembalikan tensor 1-D yang dapat diolah oleh fully connected layer. Output dari layer ini akan dimasukan ke dalam layer berikutnya yaitu overlapping Dense layer dengan nilai parameter units=1024 dan activation function ReLU. Selanjutnya output dari overlapping Dense layer akan dimasukkan ke dalam layer Dense dengan nilai parameter units=512 dan activation function ReLU. Setelah itu, output dari layer Dense(512) akan dimasukkan ke dalam output layer yang berupa Dense layer dengan nilai units= 2 sesuai dengan jumlah jenis kelas jerawat dan activation function softmax. Output dari model ini berupa nilai probabilitas untuk setiap jenis jerawat

3.3.3 Pelatihan Model

Pelatihan Model adalah tahapan dimana penulis melakukan pelatihan terhadap model yang telah dibuat sebelumnya. Sebelum melakukan pelatihan

model, seluruh layer yang terdapat dalam kelas MobileNetV2 tidak akan dimasukan ke dalam proses pelatihan. Hal ini bertujuan agar weight yang ada pada tiap layer tersebut tidak tergantikan dengan weight baru pada saat pelatihan model. Implementasi kode dari proses ini dapat dilihat dalam Gambar 3.16.

```
for layer in base_model.layers:
  layer.trainable = False
optimizer=Adam()
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
print(model.summary())
```

Gambar 3.16 Kode Proses Kompilasi Model

Model yang telah dibuat sebelumnya akan dilakukan kompilasi seperti pada Gambar 3.16 di atas. Pada proses kompilasi dilakukan pendefinisian loss function dan optimizers yang digunakan. loss function yang digunakan adalah categorical cross-entropy sedangkan optimizer yang digunakan adalah Adam. Categorical cross-entropy dipilih dikarenakan output dari model tersebut berupa nilai probabilitas untuk setiap kemungkinan jenis penyakit jerawat. Optimizer Adam dipilih dikarenakan optimizer ini dapat mendapatkan hasil yang bagus dengan waktu komputasi yang relatif cepat. Model selanjutnya dikompilasi dengan loss function dan optimizer yang telah didefinisikan dengan metrics Accuracy. Cuplikan arsitektur model dan informasi total parameter yang akan dilatih dapat dilihat pada Gambar 3.17.

```
block_16_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_16_expand_BN[0][0]
__________________________________________________________________________________________________
block_16_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_16_expand_relu[0][0]
__________________________________________________________________________________________________
block_16_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_16_depthwise[0][0]
__________________________________________________________________________________________________
block_16_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_16_depthwise_BN[0][0]
__________________________________________________________________________________________________
block_16_project (Conv2D)       (None, 7, 7, 320)    307200      block_16_depthwise_relu[0][0]
__________________________________________________________________________________________________
block_16_project_BN (BatchNorma (None, 7, 7, 320)    1280        block_16_project[0][0]
__________________________________________________________________________________________________
Conv_1 (Conv2D)                 (None, 7, 7, 1280)   409600      block_16_project_BN[0][0]
__________________________________________________________________________________________________
Conv_1_bn (BatchNormalization)  (None, 7, 7, 1280)   5120        Conv_1[0][0]
__________________________________________________________________________________________________
out_relu (ReLU)                 (None, 7, 7, 1280)   0           Conv_1_bn[0][0]
__________________________________________________________________________________________________
global_average_pooling2d (Globa (None, 1280)         0           out_relu[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 1024)         1311744     global_average_pooling2d[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 1024)         1049600     dense[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 512)          524800      dense_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 2)            1026        dense_2[0][0]
==========================================================================================
Total params: 5,145,154
Trainable params: 2,887,170
Non-trainable params: 2,257,984
__________________________________________________________________________________________________
None
```

Gambar 3.17 Rangkuman Arsitektur Model MobileNetV2

Gambar 3.17. menunjukan jumlah nilai total parameter yang terdapat dalam model yang telah dibuat, yakni senilai 5,145,154 parameter. Dari semua total parameter tersebut yang ikut dalam proses pelatihan model hanya berjumlah s2,887,170 parameter. Jumlah parameter tersebut merujuk pada jumlah parameter yang terdapat pada custom fully connected layer yang dibuat sebelumnya. Pada Gambar 3.17. juga dapat dilihat informasi mengenai jumlah parameter yang tidak ikut pada proses pelatihan, yakni sejumlah 2,257,984. Hal ini dikarenakan peneliti tidak mengikutsertakan parameter yang terdapat dalam seluruh layer kelas MobileNetV2.

```
# Trains for 75 epochs
hist = model.fit(train_set, epochs=75,
                 validation_data = val_set, verbose = 1,
                 callbacks=callbacks)
```

Gambar 3.18 Kode Pelatihan Model dengan Fungsi Fit

Gambar 3.18. adalah gambar implementasi kode untuk proses fit machine learning model ke dalam dataset yang telah dikategorikan sebelumnya menjadi data latih, data validasi, dan data uji. Hal ini bertujuan agar model machine learning yang telah dibuat sebelumnya mengetahui data yang akan digunakan dalam proses pelatihan. Pada proses ini juga dilakukan pengaturan fungsi callback yang akan digunakan dalam proses pelatihan, yakni fungsi callback yang telah didefinisikan pada sub bab sebelumnya. Proses pelatihan model akan dilakukan sebanyak 75 epochs. Satu epoch merupakan kondisi ketika seluruh dataset melewati satu kali proses forward dan backward dalam jaringan saraf. Model akan dilatih dengan data latih dan hasil dari pembelajaran tersebut akan divalidasi dengan data validasi.

```
Epoch 3/75
7/7 [==============================] - 254s 37s/step - loss: 0.4644 - accuracy: 0.7881 - val_loss: 0.3418 - val_accuracy: 0.8571
Epoch 4/75
7/7 [==============================] - 252s 37s/step - loss: 0.4347 - accuracy: 0.8095 - val_loss: 0.3490 - val_accuracy: 0.8500
Epoch 5/75
7/7 [==============================] - 257s 38s/step - loss: 0.5457 - accuracy: 0.7643 - val_loss: 0.3524 - val_accuracy: 0.8500
Epoch 6/75
7/7 [==============================] - 254s 37s/step - loss: 0.5191 - accuracy: 0.7429 - val_loss: 0.4901 - val_accuracy: 0.7143
Epoch 7/75
7/7 [==============================] - 253s 37s/step - loss: 0.4165 - accuracy: 0.8071 - val_loss: 0.3174 - val_accuracy: 0.8786
Epoch 8/75
7/7 [==============================] - 246s 38s/step - loss: 0.3569 - accuracy: 0.8262 - val_loss: 0.3319 - val_accuracy: 0.8571
Epoch 9/75
7/7 [==============================] - 254s 37s/step - loss: 0.3773 - accuracy: 0.8310 - val_loss: 0.3004 - val_accuracy: 0.9000
Epoch 10/75
7/7 [==============================] - 262s 39s/step - loss: 0.3566 - accuracy: 0.8405 - val_loss: 0.3988 - val_accuracy: 0.8357
Epoch 11/75
7/7 [==============================] - 256s 40s/step - loss: 0.3337 - accuracy: 0.8548 - val_loss: 0.3029 - val_accuracy: 0.8857
Epoch 12/75
7/7 [==============================] - 253s 37s/step - loss: 0.2927 - accuracy: 0.8762 - val_loss: 0.3064 - val_accuracy: 0.8500
Epoch 13/75
7/7 [==============================] - 259s 38s/step - loss: 0.3330 - accuracy: 0.8500 - val_loss: 0.3298 - val_accuracy: 0.8857
Epoch 14/75
7/7 [==============================] - 260s 38s/step - loss: 0.3486 - accuracy: 0.8500 - val_loss: 0.3788 - val_accuracy: 0.8357
Epoch 15/75
7/7 [==============================] - 254s 37s/step - loss: 0.2964 - accuracy: 0.8786 - val_loss: 0.3901 - val_accuracy: 0.8286
Epoch 16/75
7/7 [==============================] - 275s 41s/step - loss: 0.2875 - accuracy: 0.8714 - val_loss: 0.3124 - val_accuracy: 0.8429
Epoch 17/75
7/7 [==============================] - 249s 37s/step - loss: 0.3179 - accuracy: 0.8690 - val_loss: 0.3119 - val_accuracy: 0.8643
Epoch 18/75
7/7 [==============================] - 254s 37s/step - loss: 0.2742 - accuracy: 0.8786 - val_loss: 0.3172 - val_accuracy: 0.8643
Epoch 19/75
7/7 [==============================] - 255s 37s/step - loss: 0.2853 - accuracy: 0.8738 - val_loss: 0.3081 - val_accuracy: 0.8714
```

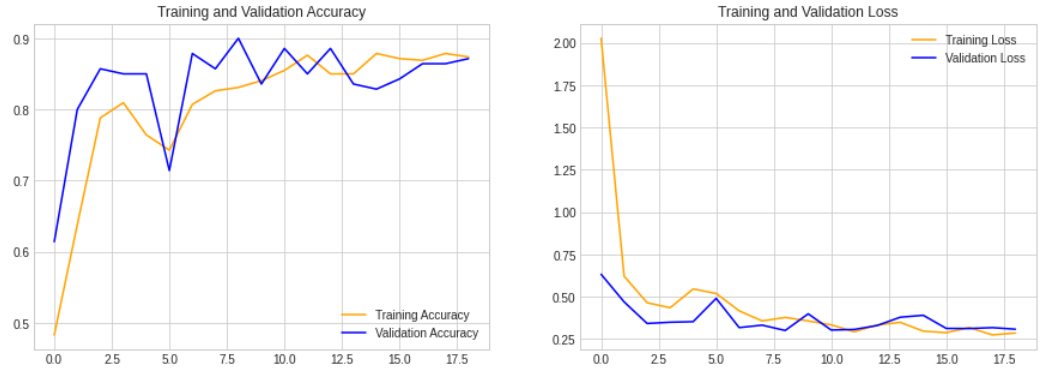
Gambar 3.19 Hasil Pelatihan Model

Gambar 3.19. merupakan gambar hasil akhir pelatihan model machine learning pada epoch ke-19 dengan akurasi 87.38% dan nilai loss 0.2853 pada data latih sedangkan nilai akurasi pada validasi sebesar 87.14% dengan nilai loss 0.3081.

3.4 Pengujian dan Evaluasi Model

Data test adalah data yang telah dipisahkan sebelumnya pada tahap preprocessing. Data yang akan di uji berjumlah 70 data yang tersebar pada setiap kelas. Untuk melakukan klasifikasi, langkah yang harus dilakukan adalah memuat semua data test kemudian menguji data test tersebut menggunakan model yang sudah dilatih.

```
plot_hist(hist)
```



a. Grafik Accuracy b. Grafik Loss

Gambar 3.20 Grafik Accuracy dan Loss Function dari Hasil Pelatihan

Gambar 3.20 merupakan bagian dari tahap pengujian dimana penulis menggunakan method plot_hist() yakni salah satu method yang disediakan oleh matplotlib untuk memvisualisasikan data histogram, namun pada penelitian ini, penulis menonaktifkan *bins* (batang) sehingga tren pelatihan model yang digambarkan hanya berupa tepian garis saja. Pembahasan pertama adalah mengenai accuracy terhadap data latih dan data validasi. Tren awal dari accuracy cenderung meningkat tiap epochnya, baik pada data latih maupun data validasi. Tren menengah accuracy cenderung mengalami fluktuasi nilai yang cukup signifikan, pada tahap ini rentan terhadap *overfit*, Namun pada tahap akhir tren yang dihasilkan oleh data latih dan data validasi mulai selaras dan stabil, ditandai oleh garis akhir yang semakin berhimpit.

Pembahasan kedua adalah mengenai loss terhadap data latih dan data validasi. Tren awal pada grafik loss memiliki selisih yang jauh dimana garis loss

pada data validasi (biru) dimulai dari nilai antara 0.50 – 0.75 sedangkan garis loss pada data latih (kuning) dimulai dari nilai 2.00. Selanjutnya pada tren menengah mengalami kemajuan, ditandai dengan grafik loss dari kedua parameter mulai berhimpit dan bertahan hingga tren akhir. Fungsi loss sudah dinilai cukup optimal untuk pengembangan tahap beta.

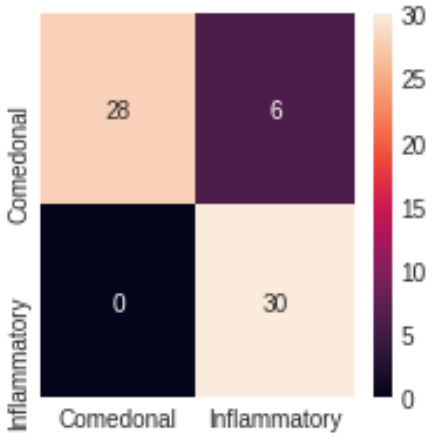
```
evaluate_model(model, X_test, y_test)
```

```
2/2 [==============================] - 1s 54ms/step - loss: 0.3018 - accuracy: 0.9062
Loss of the model is -  0.3018124997615814
2/2 [==============================] - 0s 50ms/step - loss: 0.3018 - accuracy: 0.9062
Accuracy of the model is -  90.625 %
              precision    recall  f1-score   support

   Comedonal       1.00      0.82      0.90        34
Inflammatory       0.83      1.00      0.91        30

    accuracy                           0.91        64
   macro avg       0.92      0.91      0.91        64
weighted avg       0.92      0.91      0.91        64
```

Gambar 3.21 Evaluasi Model Dengan Confusion Matrix

Pada penelitian ini, penulis melakukan proses evaluasi model dengan menggunakan metode Confusion Matrix. Confusion matrix adalah matriks (tabel) yang dapat digunakan untuk mengukur kinerja algoritma machine learning, biasanya dimanfaatkan untuk pengembangan supervised learning. Setiap baris dari matriks konfusi mewakili instance dari kelas aktual dan setiap kolom mewakili instance dari kelas yang diprediksi.

Akurasi model dapat diukur dengan beberapa metode pengukuran confusion matrix diantaranya pengukuran accuracy, pengukuran recall, pengukuran f1-score. Langkah awal untuk melakukan pengukuran adalah melakukan pemetaan hasil klasifikasi data test atau data uji yang terdiri dan tersebar pada 2 kelas.



Gambar 3.22 Confusion Matrix untuk 2 Kelas

3.4.1 Nilai Akurasi

Pengukuran akurasi ini di dasarkan dengan seberapa sering hasil klasifikasi bernilai benar secara kesulurahan. Akurasi dapat dihitung dengan persamaan 3.1.

$$Accuracy = \frac{30 + 28}{30 + 28 + 6 + 0} \quad \textbf{(3.1)}$$

$$Accuracy = \frac{58}{64} = 0.9062 = 90.62\%$$

3.4.2 Error Rate

Pengukuran nilai error atau loss dapat diukur dengan persamaan 3.2.

$$Error\ Rate = \frac{6 + 0}{30 + 28 + 6 + 0} \quad \textbf{(3.2)}$$

$$Error\ Rate = \frac{6}{64} = 0.0937 = 9.37\%$$

3.4.3 Nilai Prediksi Positif (Precision)

Precision adalah nilai dari sebuah model klasifikasi untuk mengembalikan hanya instance yang relevan. Pengukuran nilai presisi dapat diukur dengan persamaan 3.3. Berikut adalah contoh perhitungan nilai precision.

$$Precision\ (Comedonal) = \frac{28}{28+0} = 1 = 100\% \quad \textbf{(3.3)}$$

$$Precision\ (Inflammatory) = \frac{30}{30+6} = 0.8333 = 83.33\%$$

3.4.4 Nilai Kebenaran Positif (Recall)

Recall adalah nilai dari sebuah model klasifikasi untuk mengidentifikasi semua instance yang relevan. Pengukuran recall dapat dihitung menggunakan persamaan 3.4. Berikut adalah contoh pengukuran recall

$$Recall\ (Comedonal) = \frac{28}{28+6} = 0.8235 = 82.35\% \quad \textbf{(3.4)}$$

$$Recall\ (Inflammatory) = \frac{30}{30+0} = 1 = 100\%$$

3.4.5 Nilai F1-Score

F1-Score adalah sebuah metriks yang mengombinasikan recall dan precision menggunakan harmonic mean atau rata-rata harmonik. Rata-rata harmonik (harmonic mean) adalah rata-rata yang dihitung dengan cara mengubah semua data menjadi pecahan, dimana nilai data dijadikan sebagai penyebut dan pembilangnya adalah satu, kemudian semua pecahan tersebut dijumlahkan dan selanjutnya dijadikan sebagai pembagi jumlah data. Rata-rata harmonik sering disebut juga dengan kebalikan dari Rata-rata Hitung (Aritmatik). Berikut rumus harmonic mean pada persamaan 3.5.

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{xi}} \quad \textbf{(3.5)}$$

Persamaan 3.25 Rumus Harmonic Mean

Maka perhitungan F1-Score dapat dimodelkan secara matematis dengan persamaan dibawah.

$$F1 - Score\ (Comedonal) = \frac{2\ (1 * 0.82)}{1 + 0.82} = \frac{1.64}{1.82} = 0.9010 = 90.10\%$$

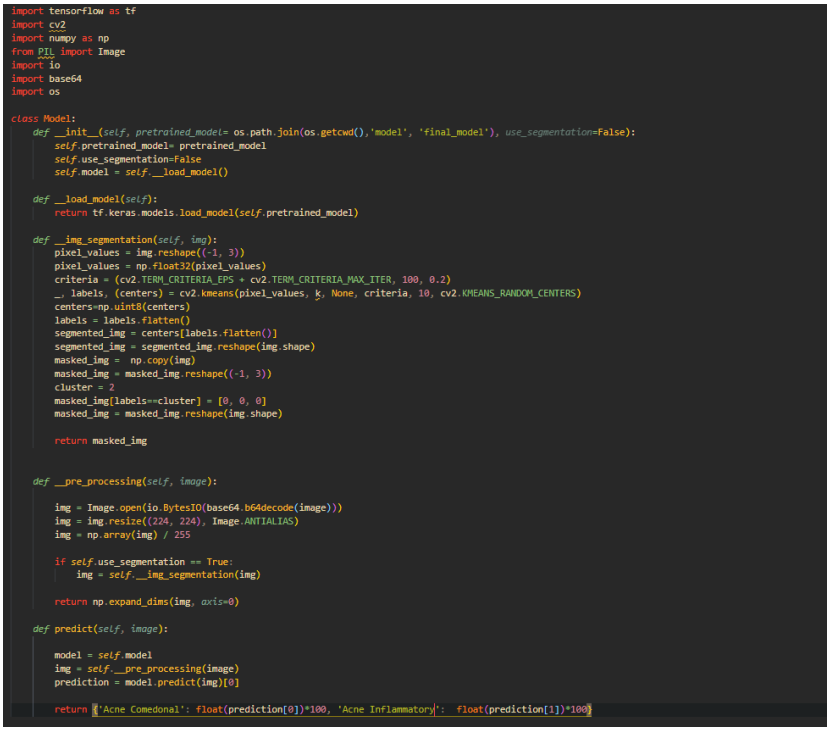
$$F1 - Score\ (Inflammatory) = \frac{2\ (0.83 * 1)}{0.83 + 1} = \frac{1.66}{1.83} = 0.9071 = 90.71\%$$

3.5 Integrasi Server Side

Proses integrasi model klasifikasi jenis jerawat ke dalam server side menggunakan microframework Flask terdiri dari dua tahap, yakni tahap prediksi jenis jerawat dan permodelan REST API.

3.5.1 Prediksi Jenis Jerawat

Pada tahap pertama penulis menginisialisasikan sebuah direktori proyek baru dengan environtment flask. Alasan pemilihan framework flask oleh penulis karena framework ini lebih sederhana dan mudah di konfigurasi. Flask memiliki satu set kecil API yang mudah dipelajari dengan dokumentasi yang lengkap. Flask menyediakan server pengembangan built-in dan proses debugger yang cepat. Fitur utama flask yang dimanfaatkan penulis adalah penanganan RESTful API dan HTTP serta mendukung penulisan API yang koheren dan rapih.



```python
import tensorflow as tf
import cv2
import numpy as np
from PIL import Image
import io
import base64
import os

class Model:
    def __init__(self, pretrained_model= os.path.join(os.getcwd(),'model', 'final_model'), use_segmentation=False):
        self.pretrained_model= pretrained_model
        self.use_segmentation=False
        self.model = self.__load_model()

    def __load_model(self):
        return tf.keras.models.load_model(self.pretrained_model)

    def __img_segmentation(self, img):
        pixel_values = img.reshape((-1, 3))
        pixel_values = np.float32(pixel_values)
        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
        _, labels, (centers) = cv2.kmeans(pixel_values, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
        centers=np.uint8(centers)
        labels = labels.flatten()
        segmented_img = centers[labels.flatten()]
        segmented_img = segmented_img.reshape(img.shape)
        masked_img =  np.copy(img)
        masked_img = masked_img.reshape((-1, 3))
        cluster = 2
        masked_img[labels==cluster] = [0, 0, 0]
        masked_img = masked_img.reshape(img.shape)

        return masked_img


    def __pre_processing(self, image):

        img = Image.open(io.BytesIO(base64.b64decode(image)))
        img = img.resize((224, 224), Image.ANTIALIAS)
        img = np.array(img) / 255

        if self.use_segmentation == True:
            img = self.__img_segmentation(img)

        return np.expand_dims(img, axis=0)

    def predict(self, image):

        model = self.model
        img = self.__pre_processing(image)
        prediction = model.predict(img)[0]

        return {'Acne Comedonal': float(prediction[0])*100, 'Acne Inflammatory':  float(prediction[1])*100}
```

Gambar 3.23 Kode Pemrosesan Citra dan Prediksi Jenis Jerawat

Gambar 3.23 menunjukan potongan kode dari file model.py yang merupakan file penghimpun fungsi pemrosesan gambar dan pelatihan model hingga menjadi sebuah prediksi jenis jerawat berupa besaran presentase atau lebih dikenal dengan sebutan *data pipeline* . Data Pipeline adalah sebuah saluran yang terdiri dari beberapa proses, mulai dari menerima input hingga menjadi sebuah hasil atau output. Langkah pertama adalah membuat sebuah kelas bernama Model yang menyimpan berbagai fungsi.

```python
import tensorflow as tf
import cv2
import numpy as np
from PIL import Image
import io
import base64
import os

class Model:
    def __init__(self, pretrained_model= os.path.join(os.getcwd(),'model', 'final_model'), use_segmentation=False):
        self.pretrained_model= pretrained_model
        self.use_segmentation=False
        self.model = self.__load_model()
```

Gambar 3.24 Kode Fungsi Init

Fungsi yang pertama yang ditunjukkan pada gambar 3.24 adalah fungsi init yaitu sebuah fungsi konstruktor yang akan dipanggil paling pertama setiap

kali program dieksekusi. Fungsi init memiliki 3 parameter yaitu self, pretrained_model, dan use_segmentation. Self adalah sebuah variable untuk merepresentasikan setiap objek yang dibuat. Parameter pretrained_model digunakan untuk mengalamatkan letak pretrained_model disimpan yaitu di dalam sebuah folder bernama model dan file bernama final_model. Parameter use_segmentation=False didefinisikan sebagai nilai default untuk menyatakan bahwa gambar tidak akan melakukan segmentasi pada saat aplikasi pertama kali dijalankan atau proses inisialisasi.

```
def __load_model(self):
    return tf.keras.models.load_model(self.pretrained_model)
```

Gambar 3.25 Kode Fungsi Load Model

Selanjutnya terdapat fungsi load_model pada gambar 3.25. Fungsi ini berfungsi untuk memuat model yang sebelumnya sudah dilatih pada google colab ke tahap production.

```
def __img_segmentation(self, img):
    pixel_values = img.reshape((-1, 3))
    pixel_values = np.float32(pixel_values)
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
    _, labels, (centers) = cv2.kmeans(pixel_values, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
    centers=np.uint8(centers)
    labels = labels.flatten()
    segmented_img = centers[labels.flatten()]
    segmented_img = segmented_img.reshape(img.shape)
    masked_img =  np.copy(img)
    masked_img = masked_img.reshape((-1, 3))
    cluster = 2
    masked_img[labels==cluster] = [0, 0, 0]
    masked_img = masked_img.reshape(img.shape)

    return masked_img
```

Gambar 3.26 Fungsi Image Segmentation

Fungsi ketiga yang ditunjukan pada gambar 3.26 adalah img_segmentation. Fungsi ini menerima 2 parameter yakni self dan img. Langkah pertama pada img_segmentation adalah melakukan reshaping data. Tujuannya adalah untuk memberikan bentuk baru pada array tanpa mengubah data aslinya.

Setelah direshape, gambar akan dikonversi menjadi float32 untuk memudahkan proses kalkulasi matematis.

Langkah kedua menambahkan variable baru yang menyimpan 2 parameter yakni cv2.TERM_CRITERIA_EPS yang berfungsi untuk menghentikan iterasi algoritma yang umumnya berupa nilai epsilon pada saat akurasi telah memenuhi kriteria, dalam hal ini penulis memasang batas akurasi sebesar-besarnya yakni 100%. cv2.TERM_CRITERIA_MAX_ITER berfungsi untuk menghentikan algoritma setelah jumlah iterasinya sudah memenuhi kriteria. Selanjutnya terdapat 3 variable yakni garis bawah (_) yang digunakan untuk mengabaikan nilai tertentu pada proses perulangan, labels yang digunakan sebagai array label yang akan mendefinisikan setiap elemen dengan ‘0’ dan ‘1’, dan centers yang merupakan susuan pusat cluster. Ketiga variable ini adalah bagian dari pemanggilan method cv2.kmeans. Method ini merupakan representasi dari algoritma unsupervised learning yang digunakan untuk melakukan segmentasi gambar. Kmeans akan memprioritaskan struktur dan tepian dari area sebuah objek serta tidak memperdulikan latar belakang dari objek tersebut. Method ini akan mengembalikan label atau nilai target yang menghasilkan kecocokan terbaik berdasarkan nilai *compactness* yaitu jumlah kuadrat jarak dari setiap titik ke pusat yang sesuai atau memenuhi kriteria.

```
def __pre_processing(self, image):

    img = Image.open(io.BytesIO(base64.b64decode(image)))
    img = img.resize((224, 224), Image.ANTIALIAS)
    img = np.array(img) / 255

    if self.use_segmentation == True:
        img = self.__img_segmentation(img)

    return np.expand_dims(img, axis=0)
```

Gambar 3.27 Fungsi Pra Pemrosesan

Pada gambar 3.27 terdapat sebuah fungsi pre_processing yaitu tahap mengubah data mentah atau biasa dikenal dengan raw data yang dikumpulkan dari berbagai sumber menjadi informasi yang lebih bersih dan bisa digunakan untuk pengolahan selanjutnya. Langkah pertama adalah proses manipulasi gambar. Penulis menggunakan method Image yang merupakan bagian dari library PIL, yakni sebuah modul yang menyediakan kemampuan untuk memuat file gambar dan manipulasi gambar. Diawali dengan membuka dan mengidentifikasi file gambar yang diberikan oleh user menggunakan method Image.open kemudian menyimpan data sebagai byte dalam buffer memori dan mengambil nilai dari buffer sebagai string menggunakan io.BytesIO. Server side akan menerima data gambar yang telah di *encoding* dari platform manapun dengan format base64. Base64 adalah sebuah string yang merupakan konversi dari data biner yang berbentuk ASCII. Lalu pada pra pemrosesan ini, penulis membaca kode yang telah di encoding sebelumnya dengan teknik decoding. Penulis menggunakan method base64.b64decode untuk mendekode string Base64 menjadi byte data yang tidak dikodekan kemudian mengubah objek seperti byte menjadi string.

Selanjutnya gambar akan diresize atau diubah lebar dan panjang menjadi 224 x 224. Penulis juga menambahkan parameter Image.ANTIALIAS yang berfungsi untuk mengurangi cacat visual yang terjadi ketika gambar beresolusi tinggi disajikan dalam resolusi yang lebih rendah. Aliasing memanifestasikan dirinya sebagai garis bergerigi atau *jaggies* di tepi objek yang seharusnya halus. Setelah proses antialias, array gambar akan dibagi dengan nilai 255, proses ini juga disebut dengan normalisasi. Normalisasi adalah proses menskalakan data suatu atribut sehingga berada dalam rentang yang lebih kecil, seperti -1,0 hingga 1,0 atau 0,0 hingga 1,0. Hal ini umumnya berguna untuk algoritma klasifikasi. Setelah semua proses pra pemrosesan dipenuhi, gambar akan mengonsumsi fungsi img_segmentation yang sebelumnya dibuat. Dengan method np.expand_dims(), gambar yang berbentuk array ini akan memperluas bentuk array dengan memasukkan sumbu baru pada posisi yang ditentukan.

```python
def predict(self, image):

    model = self.model
    img = self.__pre_processing(image)
    prediction = model.predict(img)[0]

    return {'Acne Comedonal': float(prediction[0])*100, 'Acne Inflammatory':  float(prediction[1])*100}
```

Gambar 3.28 Fungsi Prediksi Gambar

Gambar 3.28 merupakan fungsi prediksi gambar yang selanjutnya akan menghasilkan nilai persentasi dari sebuah jenis jerawat. Pada tahap ini gambar akan menggunakan model dan fungsi pre_processing yang telah dibuat sebelumnya. Apabila hasil prediksi cenderung kepada label 0 maka hasilnya adalah jerawat comedonal dan apabila hasil prediksinya cenderung kepada label 1 maka hasilnya adalah jerawat inflammatory.

3.5.2 Permodelan REST API

```python
from flask import Flask, request, jsonify
from model.model import Model
import os
import json

app = Flask(__name__)
model = Model()

@app.route('/', methods=['GET', 'POST'])
def welcome():
    return "Hello Flask!"

@app.route('/predict', methods=['POST'])
def predict():
    request_json = request.json
    print(request_json)

    prediction= model.predict(request_json.get('image'))

    return jsonify(prediction)

if __name__ == "__main__":
  app.run(debug=True, host='0.0.0.0', port='80')
```

Gambar 3.29 File main.py

Gambar 3.29 merupakan susunan kode untuk permodelan API. Beberapa pendefinisian pada file diatas sudah ada sejak projek flask di inisialisasikan atau

dikenal dengan sebutan boilerplate. Pada file ini, penulis hanya menuliskan bagian kode yang mendefinisikan model, route, serta parameter pada method run. Langkah pertama, penulis mendefinisikan variable yang menyimpan Model. Model tersebut adalah file model.py yang menyimpan serangkaian *data pipeline*. Selanjutnya pendefinisian route dengan endpoint '/' yaitu endpoint yang dibuat penulis dalam rangka uji coba terhadap request dan response yang dilakukan flask. Endpoint '/' mengembalikan method GET dan POST. Metode GET digunakan untuk mengambil data dari server pada sumber daya yang ditentukan. Misalnya, terdapat sebuah API dengan endpoint /users. Membuat permintaan GET ke endpoint itu akan mengembalikan daftar semua pengguna yang tersedia. Pada kasus ini GET pada endpoint '/' akan mengembalikan tulisan "Hello Flask". Sedangkan POST adalah method yang berfungsi untuk membuat sumber daya. Pada hal ini juga mengembalikan nilai "Hello Flask".

Endpoint utama yang digunakan oleh penulis dalam mengklasifikasikan jenis jerawat adalah '/predict'. Endpoint tersebut hanya menggunakan method POST karena tujuan utamanya adalah mengirim data gambar atau membuat resource dengan input data gambar dengan harapan output yang berupa sebuah nilai persentase dari hasil klasifikasi jenis jerawat. Endpoint '/predict' melewati fungsi predict terlebih dahul. Pada fungsi ini terdapat method request.json yang disimpan pada variable request_json. Method ini berfungsi untuk mengonversi objek JSON menjadi data Python. Terdapat hal yang perlu diketahui, bahwa request.json hanya berfungsi ketika input dari sisi client berupa JSON, sehingga perlu adanya komunikasi antara pengembang sisi client dengan pengembang sisi server. Header pada JSON yang diterima oleh sisi server harus memiliki parameter Content-type dengan nilai application/json. Jika permintaan atau request sesuai, maka sisi client dapat menerima body dari sebuah JSON tersebut. Dalam hal ini bagian body JSON adalah persentase hasil prediksi jenis jerawat.

Pada variable prediction, penulis menyematkan model.predict yang memiliki nilai request.json yang sebelumnya telah dibahas, sehingga gambar akan diterima oleh server side dan gambar jerawat tersebut dapat segera melakukan

pelatihan menggunakan model klasifikasi. Nilai dari hasil prediksi model atau variable prediction akan dikembalikan dengan method jsonify. Jsonify membuat serial data ke format JavaScript Object Notation (JSON) dam membungkusnya dalam objek Response dengan header application/json mimetype.

Langkah terakhir adalah mendefinisikan parameter pada method run. Mengatur nilai debug=True. Debug akan menampilkan beberapa informasi tentang aplikasi, seperti peta url, atau nilai konfigurasi sehingga pelacakan kesalahan lebih mudah dideteksi. Parameter kedua adalah host, penulis mendefinisikan alamat '0.0.0.0' agar pada saat aplikasi akan dihosting atau memasuki level production, host akan menyesuaikan alamat secara automatis. Parameter ketiga adalah port, penulis mendefinisikan port 80. Dengan menuliskan perintah python main.py. maka server akan menyala dan API dapat digunakan secara lokal. Pada penulisan ini, penulis menerapkan batas pengembangan pada tahap *development* saja.

3.5.3 Pengujian Model Pada Server

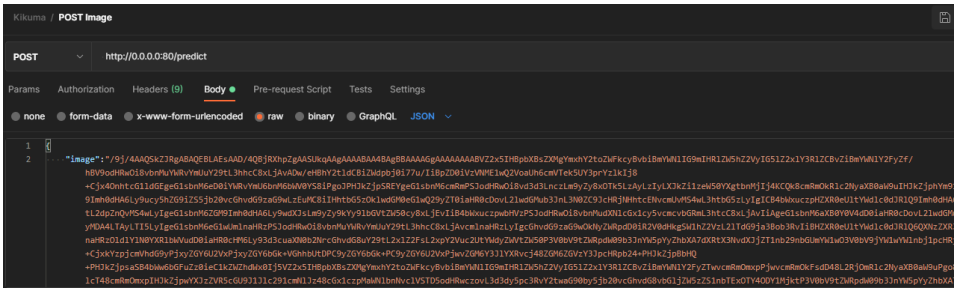
Tahap pengujian model pada sisi server merupakan langkah terakhir dalam rangkaian proses pengembangan. Aplikasi server side yang dibangun dengan flask ini berpotensi untuk diteruskan ke tahap production. Namun ada beberapa hambatan yang tidak memungkinkan untuk sampai ke tahap tersebut, diantaranya karena ukuran model yang besar dan proses pelatihan model secara real-time membutuhkan sumber daya server yang berkapasitas tinggi untuk menghasilkan pelatihan model yang cepat, maka tidak relevan apabila menghosting aplikasi server side ini dengan cara yang umum. Solusinya adalah mendeploy dan menghosting aplikasi server side pada layanan Cloud seperti Google Cloud Platform, Azure, Amazon Web Service, atau produk layanan cloud lainnya yang menyediakan fasilitas khusus untuk pemrosesan model machine learning.

Dengan demikian, penulis hanya mengujinya secara local menggunakan postman. Postman adalah salah satu platform penguji API. Postman juga dapat digunakan untuk mengumpulkan API yang dapat dibuat menjadi sebuah

dokumentasi utuh untuk satu projek tertentu. Jika dokumentasi API dibuat lengkap dengan memanfaatkan Postman akan mempermudah dalam proses pengembangan proyek karena setiap pengembang dapat memiliki acuan yang jelas untuk penggunaan setiap API.

Gambar 3.30 Pemilihan Gambar dalam Folder Local

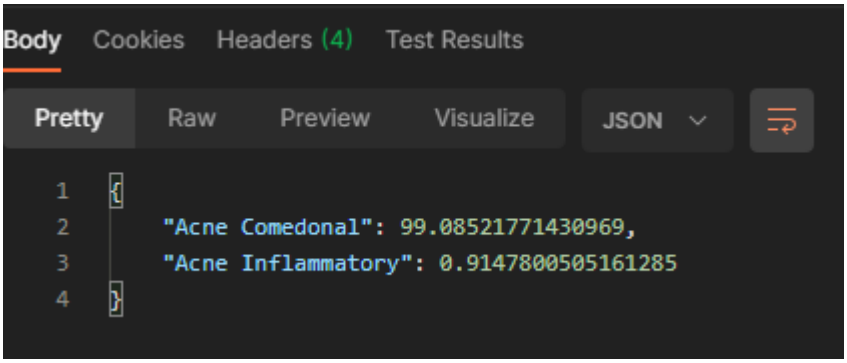
Langkah pertama adalah memilih sampel uji dalam hal ini adalah gambar jerawat comedonal yang digambarkan pada gambar 3.30 yang ingin diketahui jenisnya. Penulis memilih gambar jerawat comedonal sebagai sampel. Dalam pengujiannya, gambar tersebut harus diubah atau di encode kedalam format base64 menggunakan website konversi online sehingga menghasilkan kode base64 seperti pada gambar 3.31



Gambar 3.31 Hasil Encoding Gambar ke Format Base64

Langkah kedua adalah masuk kedalam postman dan atur konfigurasi request API. Pada request ini penulis melakukan permintaan pada endpoint

‘/predict’ menggunakan method POST. Input yang diuji adalah gambar yang sudah di konversi menjadi base64. Atur body request dengan tipe raw dan definisikan alamat base64 pada key image dalam sebuah struktur JSON. Kemudian kirim request POST.



Gambar 3.32 Persentase Klasifikasi Jenis Jerawat

Gambar 3.32 menunjukan respon API dengan method POST pada endpoint ‘/predict’ berhasil melakukan klasifikasi jenis jerawat comedonal secara benar dengan tingkat akurasi 99% sedangkan jerawat inflammatory hanya mendapatkan akurasi sekitar 9%. Dengan demikian, model sudah memenuhi kriteria dalam memprediksi jenis jerawat dengan tepat, salah satu faktor keberhasilannya adalah dengan adanya selisih prediksi yang cukup besar yakni 98%, sehingga model dinilai mampu membedakan jenis jerawat berdasarkan karateristiknya

BAB 4

PENUTUP

4.1 Kesimpulan

Berdasarkan hasil uji coba dengan citra dalam data test yang berjumlah 70 gambar per kelas atau secara keseluruhan berjumlah 140 data gambar. Setelah dilakukan analisis terhadap hasil pengujian, performa model machine learning berbasis arsitektur MobileNetV2 dalam mengklasifikasikan 2 jenis jerawat yakni jerawat comedonal dan jerawat inflammatory dapat dikatakan cukup baik. Hal ini karena akurasi prediksi model machine learning yang mencapai 90.62% dan nilai *loss* 0.3018 pada data test. Hasil pengujian model juga memenuhi target minimum akurasi yaitu 85% dan nilai *loss* kurang dari 0.5 pada data test. Model machine learning yang telah dilatih dapat digunakan pada proses selanjutnya dalam membentuk suatu sistem pendekteksi jenis jerawat berdasarkan karateristiknya. Penelitian ini menemukan solusi alternatif dalam mengoptimasi kecepatan pemrosesan dan pelatihan sebuah API model dari client side ke server side, yakni dengan melakukan encoding gambar yang direquest oleh client ke server dari format jpg, png, jpeg, dan sejenisnya menjadi format base64. Hal tersebut memungkinkan server side merespon API ke client side dengan lebih cepat karena model pada server dapat melakukan proses kalkulasi dan prediksi dengan format base64 yang memiliki bobot data lebih ringan dibandingkan format gambar.

4.2 Saran

Adapun saran yang dapat penulis berikan yaitu menambahkan varian pada data citra untuk setiap kelas, melakukan pelatihan ulang agar mendapatkan hasil model yang optimal dan maksimal, dan pembuatan sebuah aplikasi maupun sistem pendeteksi jenis jerawat berdasarkan karateristiknya yang dapat langsung diimplementasikan pada suatu platform, baik pada aplikasi website maupun aplikasi mobile. Dibutuhkan penambahan fitur threshold untuk memberikan penolakan terhadap input selain gambar jerawat pada tubuh manusia

DAFTAR PUSTAKA

[1] Abadi, Martin dkk. (2016). ”TensorFlow: A system for large-scale machine learning”. USENIX Association. Hal 265-283, Mei 2016. Diakses tanggal 19 Juni 2021 dari https://arxiv.org/pdf/1605.08695v2.pdf.

[2] Cheng, Heng-Tze dkk. (2017). ”TensorFlow Estimators: Managing Simplicity vs. Flexibility in High-Level Machine Learning Frameworks”. Association for Computing Machinery. Hal 1763-1771, Agustus 2017. Diakses tanggal 19 Juni 2021 dari https://arxiv.org/pdf/1708.02637v1.pdf.

[3] Chollet, F. (2017). “Deep Learning with Python”. Manning Publications

[4] DeepAI. (Senin, 21 Juni 2021 22.36 WIB). DeepAI, Rectified Linear Units. Tulisan pada https://deepai.org/machine-learning-glossary-and-terms/rectified-linear-units.

[5] Geron, Aurelien. (2019). “Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.)”. O’Reilly.

[6] Howard, Andrew dkk. (2017). “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. ArXiv. April 2017. Diakses tanggal 19 Juni 2021 dari https://arxiv.org/pdf/1704.04861v1.pdf.

[7] Karimkhani, C. dkk. (2017). “Global Skin Disease Morbidity and Mortality An Update From the Global Burden of Disease Study”. Jama Dermatol. Vol. 153, No. 5, Maret 2017. Diakses tanggal 18 Juni 2021 dari https://jamanetwork.com/journals/jamadermatology/fullarticle/2604831.

[8] Kementrian Kesehatan Republik Indonesia. (2012). “Roadmap Rencana Aksi Penguatan Sistem Informasi Kesehatan Indonesia”. Keputusan Menteri Kesehatan Republik Indonesia Nomor 192/Menkes/SK/VI/2021, No. 1-9, September 2012. Diakses tanggal 18 Juni 2021 dari https://web.rshs.or.id/public_html/wp-content/uploads/2014/04/KMK-No.-192-ttg-Roadmap-Aksi-Penguatan-SIK-Indonesia1.pdf.

[9] Patel, Shailee. (2020). “Types and Variants of Acne”. International Journal Of Scientific Research. Vol. 9, No. 1, Januari 2020. Diakses tanggal 18 Juni 2021 dari https://www.worldwidejournals.com/international-journal-of-scientific-research-(IJSR)/fileview/types-and-variants-of-acne_January_2020_1577860323_7215982.pdf.

[10] Sandler, Mark dkk. (2019). “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. Conference on Computer Vision and Pattern Recognition. Hal 4510-4517, Maret 2019. Diakses tanggal 19 Juni 2021 dari https://arxiv.org/pdf/1801.04381.pdf.

[11] Tensorflow. (Minggu, 20 Juni 2021 15:34 WIB). Tensorflow, Tensorflow API. Tulisan pada https://www.tensorflow.org/api_docs/python/tf?hl=id.

[12] Tschandl, P dkk. (2018). “The HAM10000 Dataset, a Large Collection of Multi-Source Dermatoscopic Images of Common Pigmented Skin Lesions”. Sci Data. Diakses pada tanggal 22 Juni 2021 https://www.nature.com/articles/sdata2018161.

[13] Wikipedia. (Minggu, 20 Juni 2021 23:07 WIB). Multilayer Perceptron. Tulisan pada https://en.wikipedia.org/wiki/Multilayer_perceptron.

[14] Zaenglein, Andrea dkk. (2017). “Guidelines of Care for the Management of Acne Vulgaris”. Journal of the American Academy of Dermatology. Vol. 74, No. 5. Diakses pada tanggal 19 Juni 2021 https://www.jaad.org/action/showPdf?pii=S0190-9622%2815%2902614-6.

[15] Zufar, Muhammad, dan Budi, Setiyono. (2016). ”Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time”. Jurnal Sains dan Seni ITS. Vol. 5, No, 2, Hal 72-77. Diakses tanggal 19 Juni 2021 dari https://www.neliti.com/publications/128862/convolutional-neural-networks-untuk-pengenalan-wajah-secara-real-time#ci.

LAMPIRAN

Lampiran 1 Pernyataan Ujicoba Aplikasi

Saya yang bertandatangan di bawah ini:

Nama : Sinatrio Bimo Wahyudi

NPM : 56418732

Judul PI : Implementasi Convolutional Neural Network Berbasis Tensorflow Pada Klasifikasi Jenis Jerawat Menggunakan Arsitektur MobileNetV2.

Menyatakan bahwa aplikasi dalam penulisan ilmiah ini telah selesai dan diujicobakan. Semua fungsi telah berjalan dengan baik.

Demikian pernyataan ujicoba ini dibuat dengan sebenar-benarnya dan dengan penuh kesadaran.

Depok, 17 - Juli - 2021

Mahasiswa	Pembimbing
	
(Sinatrio Bimo Wahyudi)	(Dr. Yulia Chalri, SKom., MMSI,)

Lampiran 2 Listing Program

1. Impor Library

```
# Library untuk manajerial file di dalam os dan di dalam gdrive
from google.colab import drive
import os

# Library untuk membaca gambar
import glob as gb

# Library untuk melakukan visualisasi data
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import seaborn as sns

# Library untuk melakukan perhitungan komputasi
import numpy as np
import pandas as pd
import datetime

# Library untuk evaluasi model
from sklearn.metrics import classification_report, confusion_matrix

# Library untuk penugasan deep learning
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import Callback, ReduceLROnPlateau,
TensorBoard, EarlyStopping, ModelCheckpoint
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
```

```
from tensorflow.keras.optimizers import Adam

from tensorflow.keras.optimizers import SGD

from tensorflow.keras.layers import Dense, Dropout, BatchNormalization,
GlobalAveragePooling2D

from tensorflow.keras.regularizers import l2
```

2. Mounting Gdrive ke Google Colab

```
from google.colab import drive

drive.mount('/content/gdrive/')
```

3. Peninjauan Dataset

```
DATASET_PATH = '/content/gdrive/MyDrive/Penulisan Ilmiah/jupyter
notebook/Acne'

os.listdir(DATASET_PATH)


TRAIN_DIR = os.path.join(DATASET_PATH, 'train')

TEST_DIR = os.path.join(DATASET_PATH, 'test')

VAL_DIR = os.path.join(DATASET_PATH, 'val')


import glob as gb

def img_count_in_folder(FOLDER_PATH):
  print('-'*70)
  for folder in os.listdir(FOLDER_PATH):
    files=gb.glob(pathname=str(FOLDER_PATH + '/' + folder + '/*.jpg'))
    print(f'{len(files):4} gambar yang ditemukan di dalam folder {folder}')
  print('-'*70)


print("TRAIN DIR")

img_count_in_folder(TRAIN_DIR)

print("VAL DIR")
```

```
img_count_in_folder(VAL_DIR)
print("TEST DIR")
img_count_in_folder(TEST_DIR)
def show_images(img_files):
  plt.figure()
  fig = plt.figure(figsize=(10, 10))
  fig.patch.set_facecolor('xkcd:gray')

  for i in range(len(img_files)):
       plt.subplot(5,5,i+1)    # the number of images in the grid is 5*5 (25)
       img=mpimg.imread(img_files[i])
       plt.imshow(img)
       plt.tight_layout()
       plt.axis('off')
       plt.title(img_files[i].split("/")[7])
    plt.show()

def list_files(dir):
  arr=[]
  for root,dirs, files in os.walk(dir):
    for name in files:
      if name.endswith('.jpg'):
        arr.append(os.path.join(root, name))
        break
  return arr
img_list =[f'{DATASET_PATH}/train/Comedonal/comedonal-1.jpg',
f'{DATASET_PATH}/train/Comedonal/comedonal-2.jpg']
img_list
```

4. Image Segmentation

```
from google.colab import files
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
import numpy as np
from sklearn.cluster import KMeans

filename = '04l5I8TqdzF9WDMJ.png'

img = image.load_img(
   img_list[0],
   target_size=(150, 150))

x = image.img_to_array(img)/255
kmeans = KMeans(10)
kmeans.fit(x.reshape(-1, 3))
x_segment = kmeans.predict(x.reshape(-1, 3)).reshape(x.shape[0], x.shape[1])

plt.imshow(x_segment,'gray')
plt.axis(False)

from google.colab import files
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
import numpy as np
from sklearn.cluster import KMeans

# fig = plt.figure(figsize=()
filename = '04l5I8TqdzF9WDMJ.png'
```

```
img = image.load_img(
   img_list[1],
   target_size=(150, 150))

x = image.img_to_array(img)/255
kmeans = KMeans(10)
kmeans.fit(x.reshape(-1, 3))
x_segment = kmeans.predict(x.reshape(-1, 3)).reshape(x.shape[0], x.shape[1])

plt.imshow(x_segment)
plt.axis(False)

from google.colab.patches import cv2_imshow
import cv2

img = cv2.imread(img_list[1])
cv2_imshow(img)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
pixel_values = img.reshape((-1, 3))
pixel_values = np.float32(pixel_values)
print(pixel_values.shape)

criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

k=4
```

```
_, labels, (centers) = cv2.kmeans(pixel_values, k, None, criteria, 10,
cv2.KMEANS_RANDOM_CENTERS)

centers=np.uint8(centers)
labels = labels.flatten()

segmented_img = centers[labels.flatten()]
segmented_img = segmented_img.reshape(img.shape)
plt.imshow(segmented_img)
plt.show

masked_img =  np.copy(img)
masked_img = masked_img.reshape((-1, 3))
cluster = 2
masked_img[labels==cluster] = [0, 0, 0]
masked_img = masked_img.reshape(img.shape)

plt.imshow(masked_img)
plt.show()

def img_segmentation(img, k=4):
  pixel_values = img.reshape((-1, 3))
  pixel_values = np.float32(pixel_values)

  criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

  _, labels, (centers) = cv2.kmeans(pixel_values, k, None, criteria, 10,
cv2.KMEANS_RANDOM_CENTERS)
  centers=np.uint8(centers)
```

```
    labels = labels.flatten()
    segmented_img = centers[labels.flatten()]
    segmented_img = segmented_img.reshape(img.shape)

    masked_img =  np.copy(img)
    masked_img = masked_img.reshape((-1, 3))
    cluster = 4
    masked_img[labels==cluster] = [0, 0, 0]
    masked_img = masked_img.reshape(img.shape)

    return masked_img
```

5. Image Preprocessing

```
# Datagen untuk validation set dan training set
datagen = ImageDataGenerator(preprocessing_function=img_segmentation,
                             rescale = 1./255,
                             rotation_range = 25,
                             zoom_range = [0.5, 1.1],
                             width_shift_range=0.15,
                             shear_range= 0.15,
                             height_shift_range=0.15,
                             horizontal_flip=True,
                             vertical_flip= True,
                             fill_mode='nearest')

# ImageDataGenerator untuk Test Set
val_datagen = ImageDataGenerator(preprocessing_function=img_segmentation,
   rescale=1./255)

test_datagen = ImageDataGenerator(preprocessing_function=img_segmentation,
rescale=1./255)
```

```
IMG_SIZE = (224, 224)

# Pembuatan dataset pelatihan dengan datagen
train_set = datagen.flow_from_directory(TRAIN_DIR,
                                        class_mode='categorical',
                                        batch_size= 64,
                                        target_size=IMG_SIZE,
                                        seed=42)
val_set = val_datagen.flow_from_directory(VAL_DIR,
                                        class_mode='categorical',
                                        batch_size= 64,
                                        target_size=IMG_SIZE,)
test_set = test_datagen.flow_from_directory(TEST_DIR,
                                        class_mode='categorical',
                                        batch_size= 64,
                                        target_size=IMG_SIZE,)
# Pembuatan variabel X_train dan y_train
X_train, y_train = train_set.next()

# Pembuatan variabel X_test dan y_test
X_test, y_test = test_set.next()

# Pembuatan variabel X_val dan y_val
X_val, y_val = val_set.next()
X_train.shape
class_dict = {v : k for k, v in train_set.class_indices.items()}

def plot_augmented_imgs(X, y):
```

```
 labels=[]
 y_label = np.argmax(y, axis=1)

 for label in y_label:
  labels.append(class_dict[label])

 for idx in range(0,10):
    plt.figure(figsize=(5, 5))
    plt.imshow(X[idx])
    plt.title(labels[idx])

plot_augmented_imgs(X_train, y_train)
plot_augmented_imgs(X_val, y_val)
plot_augmented_imgs(X_test, y_test)
```

6. Pengembangan Model

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.01, min_lr=1e-6)
checkpoint_cb = ModelCheckpoint("/content/gdrive/MyDrive/model/MobileNetV
2/current_best_model.h5", saved_best_only=True)
early_stop_cb = EarlyStopping(patience=10,monitor='val_loss', restore_best_wei
ghts= True)

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)

# Menghapus logs dari proses run sebelumnya
!rm -rf ./logs/

callbacks =[reduce_lr, checkpoint_cb, early_stop_cb, tensorboard_callback]
```

```
test_set.class_indices

def evaluate_model(model, X, y):
  print('Loss of the model is - ', model.evaluate(X, y)[0])
  print('Accuracy of the model is - ', model.evaluate(X, y)[1]*100, '%')
  predict = model.predict(X)
  predict = np.argmax(predict, axis=1)
  predict = predict.reshape(1, -1)[0]
  y_label = np.argmax(y, axis=1)
  print(classification_report(y_label, predict, target_names =['Comedonal',
'Inflammatory']))

  conf_matrix = confusion_matrix(y_label, predict)
  df_cm = pd.DataFrame(conf_matrix, index = [i for i in ('Comedonal',
'Inflammatory')],
                columns = [i for i in ('Comedonal', 'Inflammatory')])
  plt.figure(figsize = (3, 3))
  sns.heatmap(df_cm, annot=True)

plt.style.use('seaborn-whitegrid')

def plot_acc(history):
  acc = history.history['accuracy']
  val_acc = history.history['val_accuracy']
  epochs = range(len(acc))
  plt.subplot(1, 2, 1)
  acc_plot, = plt.plot(epochs, acc, 'orange')
  val_acc_plot, = plt.plot(epochs, val_acc, 'blue')
  plt.title('Training and Validation Accuracy')
  plt.legend([acc_plot, val_acc_plot], ['Training Accuracy', 'Validation Accuracy'])
def plot_loss(history):
```

```
  loss = history.history['loss']
  val_loss = history.history['val_loss']
  epochs = range(len(loss))
  plt.subplot(1, 2, 2)
  loss_plot, = plt.plot(epochs, loss, 'orange')
  val_loss_plot, = plt.plot(epochs, val_loss, 'blue')
  plt.title('Training and Validation Loss')
  plt.legend([loss_plot, val_loss_plot], ['Training Loss', 'Validation Loss'])

def plot_hist(history):
  plt.figure(figsize=(15,5))
  plot_acc(history)
  plot_loss(history)
```

7. Pengembangan Arsitektur dan Transfer Learning dengan MobileNetV2

```
# Transfer Learning using MobileNetV2
base_model=MobileNetV2(weights='imagenet',
                       input_shape=(224, 224, 3),
                       include_top=False)
#  Mendapatkan output dari basis model
x=base_model.output
# Membuat custom fully connected layer
x=GlobalAveragePooling2D()(x)
#we add dense layers so that the model can learn more complex functions and classify for better results.
x=Dense(1024,activation='relu')(x)
x=Dense(1024,activation='relu')(x) #dense layer 2
x=Dense(512,activation='relu')(x) #dense layer 3
preds=Dense(2,activation='softmax')(x) #final layer with softmax activation for N classes
```

```
# Mengimplementasikan rancangan ke dalam variabel model
model=Model(inputs=base_model.input,outputs=preds)
for layer in base_model.layers:
  layer.trainable = False
optimizer=Adam()
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
print(model.summary())


# Pelatihan untuk 75 epochs
hist = model.fit(train_set, epochs=75, validation_data = val_set, verbose = 1, callbacks=callbacks)
plot_hist(hist)
evaluate_model(model, X_test, y_test)
evaluate_model(model, X_val, y_val)
evaluate_model(model, X_train, y_train)
```

8. Evaluasi Model

```
evaluate_model(model,X_test, y_test)
```

9. Pengecekan Model Terbaik dari Proses Pelatihan

```
# Cek model terbaik
import tensorflow as tf
best_model = tf.keras.models.load_model('/content/gdrive/MyDrive/model/MobileNetV2/current_best_model.h5')
best_model.summary()


evaluate_model(best_model, X_test, y_test)

best_model.save('/content/gdrive/MyDrive/model/MobileNetV2/final_model')
```

```
best_model.save('/content/gdrive/MyDrive/model/MobileNetV2/final_model/final_model.h5')
```

10. Testing dengan Gambar Acak

```
from google.colab import files
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
import numpy as np

uploads = files.upload()

n_files = len(uploads)
fig = plt.figure(figsize=(2, 2 * n_files))

for i, filename in enumerate(uploads):
  img = image.load_img(
      filename,
      target_size=(224, 224))
  x = image.img_to_array(img)/255
  x = np.expand_dims(x, axis=0)

  predict = model.predict(x)
  class_prediction = np.argmax(predict)
  prediction = class_dict[class_prediction]


  ax = fig.add_subplot(n_files, 1, i+1)
  ax.imshow(img)
  ax.set_title(prediction)
```

```
  ax.axis(False)
  print('percentage:')
  print(predict)
```

11. Model.py

```
import tensorflow as tf
import cv2
import numpy as np
from PIL import Image
import io
import base64
import os


class Model:
    def __init__(self, pretrained_model= os.path.join(os.getcwd(),'model', 'final_m
odel'), use_segmentation=False):
        self.pretrained_model= pretrained_model
        self.use_segmentation=False
        self.model = self.__load_model()


    def __load_model(self):
        return tf.keras.models.load_model(self.pretrained_model)


    def __img_segmentation(self, img):
        pixel_values = img.reshape((-1, 3))
        pixel_values = np.float32(pixel_values)
        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_IT
ER, 100, 0.2)
```

```
        _, labels, (centers) = cv2.kmeans(pixel_values, k, None, criteria, 10, cv2.KM
EANS_RANDOM_CENTERS)
        centers=np.uint8(centers)
        labels = labels.flatten()
        segmented_img = centers[labels.flatten()]
        segmented_img = segmented_img.reshape(img.shape)
        masked_img =  np.copy(img)
        masked_img = masked_img.reshape((-1, 3))
        cluster = 2
        masked_img[labels==cluster] = [0, 0, 0]
        masked_img = masked_img.reshape(img.shape)

        return masked_img

    def __pre_processing(self, image):

        img = Image.open(io.BytesIO(base64.b64decode(image)))
        img = img.resize((224, 224), Image.ANTIALIAS)
        img = np.array(img) / 255

        if self.use_segmentation == True:
            img = self.__img_segmentation(img)
        return np.expand_dims(img, axis=0)

    def predict(self, image):
        model = self.model
        img = self.__pre_processing(image)
        prediction = model.predict(img)[0]
```

```
        return {'Acne Comedonal': float(prediction[0])*100, 'Acne Inflammatory':  fl
oat(prediction[1])*100}
```

12. Main.py

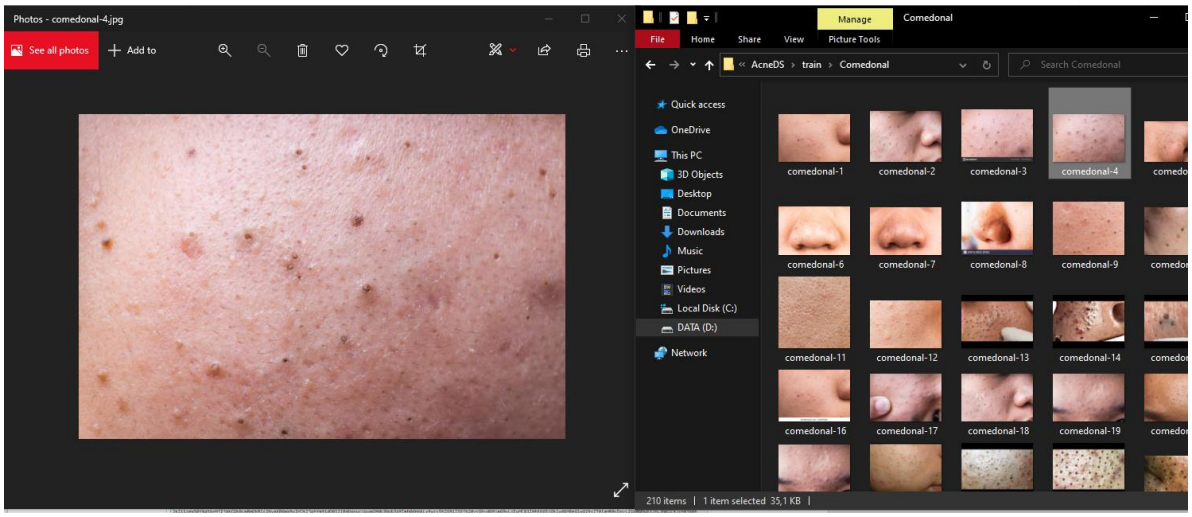
```
from flask import Flask, request, jsonify
from model.model import Model
import os
import json

app = Flask(__name__)
model = Model()

@app.route('/', methods=['GET', 'POST'])
def welcome():
    return "Hello Flask!"

@app.route('/predict', methods=['POST'])
def predict():
    request_json = request.json
    print(request_json)

    prediction= model.predict(request_json.get('image'))

    return jsonify(prediction)

if __name__ == "__main__":
  app.run(debug=True, host='0.0.0.0', port='80')
```

13. Generate_image.py

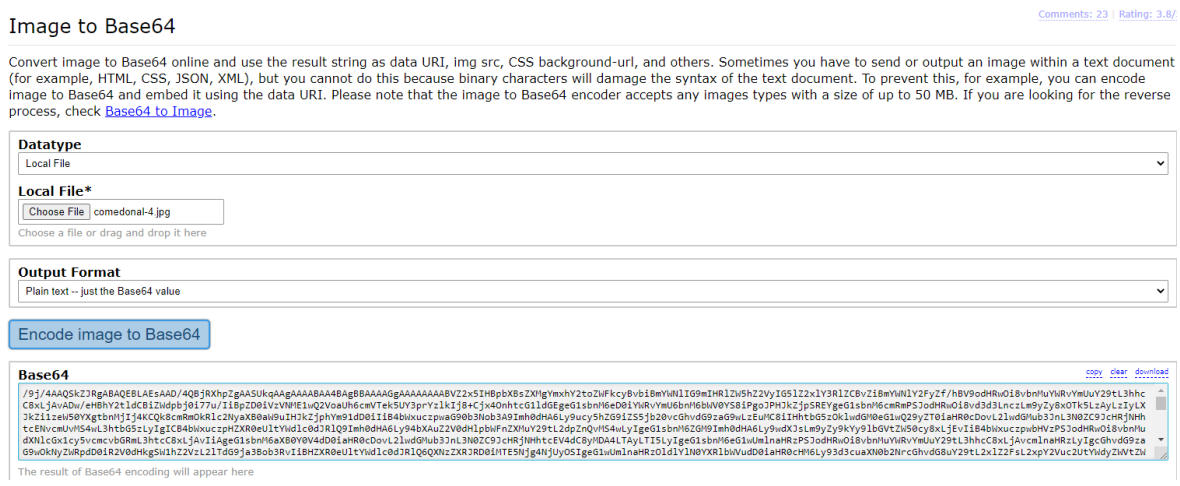
```
from model.model import Model
import os
import base64
import io
from PIL import Image

IMG_PATH = os.path.join(os.getcwd(),'test_images','acne-cystic-11.jpg')
encoded_img =''
with open(IMG_PATH, "rb") as image2string:
    encoded_img = base64.b64encode(image2string.read())
print(encoded_img)
print(type(encoded_img))

model = Model()

prediction= model.predict(encoded_img)
print(prediction)
```

Lampiran 3 Listing Output

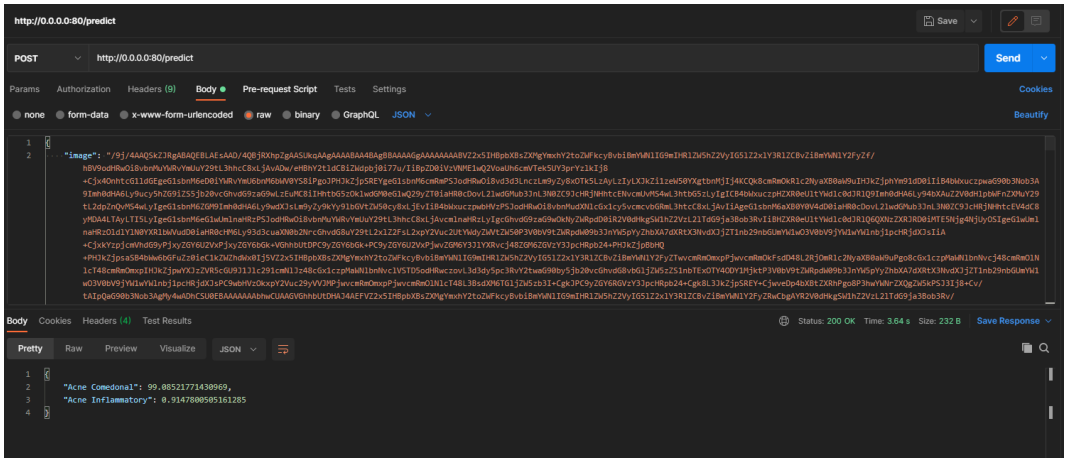
1. Gambar Jerawat Format PNG

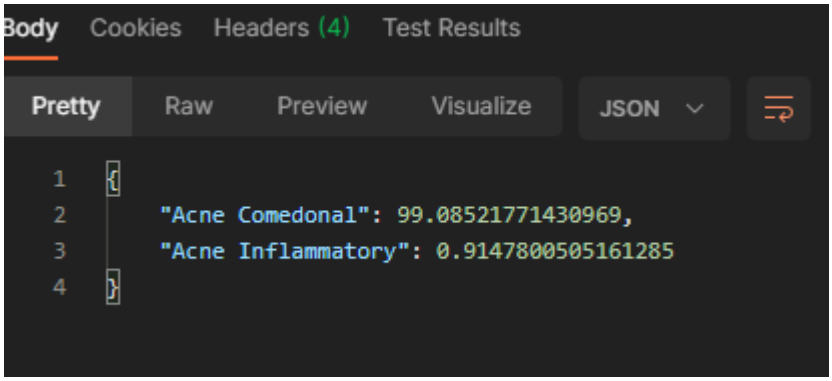


2. Konversi Gambar Format PNG Menjadi Format Base64



3. Testing Klasifikasi Gambar Jerawat Lewat Postman


http://0.0.0.0:80/predict
Save
POST
http://0.0.0.0:80/predict
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
"image": "/9j/4AAQSkZJRgABAQEBLAEsAAD/4QBjRXhpZgAASUkqAAgAAAABAA4BAgB8BAAAGgAAAAAAAABVZ2x5IHBpbXBsZXMgYmxhY2toZWFkcy0vbi8mYWNlIG9mIHRlZW5hZ2VyIG51Z2x1Y3R1ZCBvZi8mYWNlY2FyZf/
"Acne Comedonal": 99.08521771430969,
"Acne Inflammatory": 0.9147800505161285


Body
Cookies
Headers (4)
Test Results
Pretty
Raw
Preview
Visualize
JSON
"Acne Comedonal": 99.08521771430969,
"Acne Inflammatory": 0.9147800505161285