

**APLIKASI WEB PENDETEKSI JERAWAT PADA WAJAH
MENGUNAKAN MODEL *DEEP LEARNING*
DENGAN TENSORFLOW**



Disusun Oleh:

N a m a : July Arifianto

NIM : 17523216

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**APLIKASI WEB PENDETEKSI JERAWAT PADA WAJAH
MENGUNAKAN MODEL *DEEP LEARNING*
DENGAN TENSORFLOW**

TUGAS AKHIR



(Izzati Muhiimah, S.T., M.Sc., Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

**APLIKASI WEB PENDETEKSI JERAWAT PADA WAJAH
MENGUNAKAN MODEL *DEEP LEARNING*
DENGAN TENSORFLOW**

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 6 Januari 2022

Tim Penguji

Izzati Muhimmah, S.T., M.Sc., Ph.D.



Anggota 1

Elyza Gustri Wahyuni, S.T., M.Cs.



Anggota 2

Rian Adam Rajagede, S.Kom., M.Cs.



الجمعة الاستاذة الأستاذة
Mengetahui,

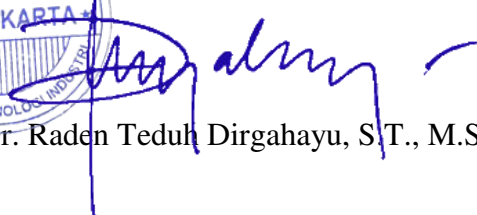
Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia



(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)



HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : July Arifianto

NIM : 17523216

Tugas akhir dengan judul:

APLIKASI WEB PENDETEKSI JERAWAT PADA WAJAH MENGUNAKAN MODEL *DEEP LEARNING* DENGAN TENSORFLOW

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 6 Januari 2022



(July Arifianto)

HALAMAN PERSEMBAHAN

Pertama saya persembahkan tugas akhir ini kepada kedua orang tua tercinta, ibu Sri Sunarti dan bapak Sukirno. Berkat doa dan dukungan ibu dan bapak lah pengerjaan tugas akhir ini dapat selesai dengan lancar. Pertanyaan yang selalu ibu tanyakan, “*piye le skirpsine wes rampung durung?*” selalu menjadi dorongan untuk segera menyelesaikan tugas akhir ini. Saya ingin membuat ibu dan bapak bangga melihat anakmu ini bisa lulus dengan gelar S.Kom.

Kedua kepada kedua nenek saya, semoga nenek selalu diberikan kesehatan dan panjang umur. Saya tahu doa nenek selalu mendampingi cucumu ini. Setiap saya berkunjung, nenek pasti selalu membacakan doa agar cucumu ini sukses dan berpesan agar jangan meninggalkan sholat lima waktu. *Insyallah* cucumu ini bisa membuat nenek bangga.

Ketiga untuk kedua adik saya yang saya sayangi, Isnawati dan Anis Nurhidayah. Terima kasih atas doa dan semangatnya, walaupun terkadang kakakmu ini sering jail dan sering merepotkan. Aku juga mendoakan semoga sekolah kalian berjalan lancar dan nantinya dapat diterima di perguruan tinggi yang diimpikan.

Keempat kepada keluarga dan teman-teman dekat yang selalu bertanya “kapan sidang?”, “*wes lulus durung?*”. Ini, *alhamdulillah* tugas akhirnya sudah selesai. Bagi kalian yang sudah lulus dan bekerja, tunggu, aku segera menyusul kalian. Maaf kalau ada yang menawarkan projek selama ini aku tolak dengan alasan sedang sibuk mengerjakan skripsi. Setelah ini, *insyaallah* sudah siap untuk diajak proyekan lagi.

HALAMAN MOTO

*"If something's important enough, you should try.
Even if you the probable outcome is failure."*

- Elon Musk



KATA PENGANTAR

Assalamu'alaikum warohmatullohi wabarokatuh

Alhamdulillah puji syukur senantiasa penulis panjatkan atas karunia serta hidayah yang Allah SWT telah berikan sehingga penulis dapat menyelesaikan tugas akhir yang berjudul “Aplikasi Web Pendeteksian Jerawat Pada Wajah Menggunakan Model *Deep Learning* Dengan TensorFlow”. Tugas Akhir ini sebagai adalah salah satu persyaratan yang harus dipenuhi dalam menyelesaikan jenjang sarjana di Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Shalawat serta salam semoga selalu tercurahkan kepada Nabi Muhammad SAW serta para sahabat dan pengikutnya sampai akhir zaman.

Proses penyelesaian tugas akhir ini tidak terlepas dari bantuan, dan bimbingan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Bapak Hari Purnomo, Prof., Dr., Ir., M.T., IPU selaku Dekan Fakultas Teknologi Industri, Universitas Islam Indonesia.
2. Bapak Hendrik, S.T., M.Eng selaku Ketua Jurusan Informatika, Universitas Islam Indonesia.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Jurusan Informatika Program Sarjana, Universitas Islam Indonesia.
4. Bapak Rian Adam Rajagede, S.Kom., M.Cs. dan Ibu Elyza Gustri Wahyuni, S.T., M.Cs. selaku dosen penguji yang telah memberikan kritik dan sarannya.
5. Ibu Izzati Muhimmah, S.T., M.Sc., Ph.D. selaku Dosen Pembimbing yang telah banyak meluangkan waktu, memberikan saran, dan arahan dalam proses pengerjaan tugas akhir ini.
6. Seluruh dosen dan staf pengajar Jurusan Informatika UII yang telah memberikan ilmunya kepada penulis sehingga dapat menjadi bekal dalam mengerjakan tugas akhir ini.
7. Kedua orang tua penulis Bapak Sukirno dan Ibu Sri Sunarti yang selalu mendoakan dan memberikan dukungan baik secara materi maupun moril.
8. Adik penulis yaitu Isnawati dan Anis Nurhidayah yang telah memberikan semangat dan motivasi kepada penulis.
9. Teman-teman satu satu bimbingan tugas akhir yang senantiasa berbagi ilmu dan informasi yang bermanfaat sampai tugas akhir ini selesai.

10. Teman-teman UKM Robotic yang selalu memberikan semangat dan motivasi.
11. Teman-teman Jurusan Informatika Angkatan 2017 yang sedang berjuang bersama untuk meraih gelar S.Kom dan Toga UII, terima kasih atas pengalaman berharga selama menjadi mahasiswa Informatika UII
12. Semua pihak yang tidak dapat disebut satu per satu, terima kasih.

Semoga segala bantuan, bimbingan, doa, dan ilmu yang telah diberikan kepada penulis mendapatkan imbalan dari Allah SWT. Penulis memohon maaf apabila dalam proses penyusunan tugas akhir ini terdapat kekhilafan dan kesalahan. Penulis menyadari sepenuhnya bahwa masih memiliki banyak keterbatasan kemampuan dalam penulisan tugas akhir ini. Oleh karena itu penulis mengharap adanya kritik dan saran yang membangun demi kesempurnaan penyusunan dan penulisan tugas akhir ini. Semoga tugas akhir ini dapat bermanfaat bagi semua yang telah membaca dan membutuhkan.

Wassalamu 'alaikum warohmatullohi wabarokatuh

Yogyakarta, 6 Januari 2022

Penulis

الجامعة الإسلامية
الاستاذة الأندونيسية

SARI

Jerawat merupakan permasalahan kulit yang umum. Jerawat muncul di bidang dermatologi untuk usia 15 - 40 tahun, 85% pada kelompok usia remaja dengan rentang usia 12 - 24 tahun. Jerawat paling sering terjadi pada wajah terutama area dahi, pipi, dan dagu karena berbagai faktor etiologi dari jerawat. Asesmen jerawat merupakan prosedur awal yang secara umum dilakukan oleh dokter kulit sebelum memberikan tindakan perawatan terhadap kulit jerawat. Asesmen pada jerawat tidak ada prosedur penilaian yang dianggap sebagai standar global yang memunculkan masalah antar dan intra pengamat. Selain itu, dokter kulit tidak memiliki cukup waktu untuk menghitung lesi jerawat dengan benar karena mereka harus menemui banyak pasien dalam waktu yang terbatas. Bagi sebagian dokter kulit, menghitung lesi jerawat itu membosankan karena berbagai jenis dan jumlah lesi yang ada. Pengembangan *artificial intelligence* saat ini digunakan untuk membantu pekerjaan manusia untuk menjalankan tugas di berbagai bidang, salah satunya adalah pada bidang *computer vision*. *Computer vision* merupakan disiplin ilmu yang mengembangkan kemampuan komputer untuk mendapatkan informasi dari citra digital yang diamati. Proses asesmen jerawat dapat dibantu dengan menggunakan *computer vision* ini. Pengembangan *computer vision* untuk pendeteksian objek paling banyak menggunakan model *deep learning*. Pendeteksian dilakukan dengan menggunakan model *deep learning* yang dilatih untuk mengenali objek jerawat. Foto-foto jerawat yang muncul di area wajah merupakan dataset yang digunakan untuk melatih model. Model pendeteksian jerawat dibuat dengan menggunakan metode *transfer learning* dari model SSD Resnet50 V1 COCO. Proses pelatihan model dilakukan dengan menggunakan TensorFlow di Google Colab. Hasil pendeteksian jerawat menggunakan model *deep learning* dinilai dapat berjalan dengan baik. Model pendeteksian jerawat nilai *total loss* sebesar 0,135, nilai mAP sebesar 42.1%, dan nilai AR sebesar 32%. Model pendeteksian dijalankan menggunakan TensorFlow Serving yang membutuhkan memori dan waktu *inference* lebih sedikit. Model hasil pelatihan kemudian diimplementasikan kedalam aplikasi berbasis web. Pengguna cukup mengunggah foto dan mendapatkan hasilnya. Proses deteksi jerawat pada aplikasi rata-rata memerlukan waktu 2,77 detik per gambar.

Kata kunci: jerawat, *object detection*, *deep learning*, aplikasi web.

GLOSARIUM

Anotasi	informasi tambahan yang terkait dengan titik tertentu dalam berkas.
API	sebuah <i>software</i> yang menyediakan layanan spesifik yang dapat diakses melalui Web oleh berbagai jenis <i>software</i> lain.
Backend	bagian dari aplikasi yang bertanggung jawab untuk menyediakan kebutuhan yang tak terlihat oleh pengguna.
Blob	data biner besar yang disimpan dalam satu kesatuan.
Compile	proses untuk mengubah berkas kode program dengan berkas lain yang terkait menjadi berkas yang siap untuk dieksekusi oleh sistem operasi secara langsung.
CSV	format file di mana setiap data dalam baris dipisahkan oleh tanda koma atau titik koma.
Dataset	istilah informal untuk kumpulan data dan menyangkut suatu topik tertentu.
Endpoint	semacam titik masuk ke aplikasi yang memungkinkan pengguna berkomunikasi melalui protokol HTTP.
Frontend	bagian dari aplikasi yang terlihat dan digunakan langsung oleh pengguna.
GPU	prosesor yang terdiri dari banyak inti yang lebih kecil dan lebih khusus.
JSON	suatu format ringkas pertukaran data komputer
Landmark	lokalisasi titik-titik yang menonjol
Library	koleksi dari rutin-rutin program untuk membangun dan mengembangkan perangkat lunak
Tensor	struktur data berupa matriks atau <i>array</i> multidimensi
XML	bahasa markup untuk keperluan umum yang disarankan oleh W3C untuk membuat dokumen <i>markup</i> untuk keperluan pertukaran data antar sistem yang beraneka ragam.
Foto close up	foto yang diambil dengan jarak yang sangat dekat sehingga objek foto terlihat jelas.

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR.....	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR.....	vii
SARI	ix
GLOSARIUM	x
DAFTAR ISI	xi
DAFTAR TABEL	xiv
DAFTAR GAMBAR.....	xv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	3
BAB II TINJAUAN PUSTAKA	5
BAB III LANDASAN TEORI.....	9
3.1 Pengetahuan Dasar Tentang Jerawat (<i>acne vulgaris</i>)	9
3.2 Keparahan Jerawat	10
3.3 Pengolahan Citra Digital	11
3.3.1 Definisi Citra Digital	12
3.3.2 Jenis Citra Digital	12
3.3.3 <i>Feature Extraction</i>	13
3.4 <i>Computer Vision</i>	13
3.4.1 <i>Image Classification</i>	13
3.4.2 <i>Semantic Segmentation</i>	14
3.4.3 <i>Object Detection</i>	14
3.4.4 <i>Instance Segmentation</i>	15
3.5 <i>Machine Learning</i>	16
3.5.1 <i>Supervised Learning</i>	16
3.5.2 <i>Unsupervised Learning</i>	16
3.5.3 <i>Semi-Supervised Learning</i>	17
3.5.4 <i>Reinforcement Learning</i>	17
3.6 <i>Deep Learning</i>	17
3.6.1 <i>Artificial Neural Network (ANN)</i>	17
3.6.2 <i>Forward Propagation</i>	18
3.6.3 <i>Back Propagation</i>	18
3.6.4 <i>Activation Function</i>	19
3.7 <i>Convolutional Neural Network</i>	20
3.7.1 <i>Convolution Layer (Conv. Layer)</i>	20
3.7.2 <i>Pooling Layer</i>	21

3.7.3	<i>Fully Connected Layer</i>	21
3.8	<i>Object Detection</i>	22
3.9	<i>Image Augmentation</i>	22
3.10	Anotasi Citra.....	23
3.11	<i>Single Shot Multibox Detector (SSD)</i>	24
3.12	<i>Python</i>	26
3.13	Web API.....	26
3.14	Flask	27
3.15	VueJS.....	27
3.16	<i>Evaluation Matrix</i>	28
3.17	TensorFlow	29
3.18	TensorFlow Serving	29
3.19	Google Colaboratory	30
BAB IV METODOLOGI PENELITIAN.....		32
4.1	Populasi dan Sampel Penelitian	32
4.2	Variabel dan Definisi Operasional Penelitian	32
4.3	Jenis dan Sumber Data	32
4.4	Metode Analisis Data.....	32
4.5	Tahapan Penelitian	32
4.6	Evaluasi Hasil	33
BAB V ANALISIS DAN PEMBAHASAN.....		34
5.1	Pengumpulan Dataset.....	34
5.2	<i>Preprocessing</i> citra	34
5.2.1	Segmentasi Wajah	35
5.2.2	<i>Resize</i> Citra.....	35
5.2.3	Pelabelan Citra	36
5.2.4	Pembagian Citra	37
5.2.5	Augmentasi Citra.....	37
5.3	Konfigurasi Google Colab	38
5.3.1	Mengunggah Dataset ke Google Drive	38
5.3.2	Menghubungkan Google Colab dan Google Drive	39
5.3.3	Instalasi <i>Library</i> Python	39
5.4	Membuat TensorFlow Record	39
5.4.1	Membuat <i>Label Map</i>	39
5.4.2	Mengubah Berkas XML ke CSV	40
5.4.3	TFRecord Data <i>Train</i>	41
5.4.4	TFRecord Data <i>Test</i>	41
5.5	Pelatihan Model	42
5.5.1	Konfigurasi Pipeline Pelatihan Deteksi Objek.....	42
5.5.2	Pelatihan Model Deteksi Objek.....	43
5.5.3	Ekspor Model	44
5.6	Model Hasil Pelatihan.....	44
5.6.1	<i>Learning Rate</i>	44
5.6.2	<i>Total Loss</i>	45
5.6.3	Model.....	46
5.6.4	Pengujian Model.....	46
5.7	Desain Aplikasi	47
5.7.1	Alur data	47
5.7.2	Desain Antarmuka Aplikasi	48
5.8	Pengembangan Aplikasi.....	49

5.8.1	Menjalankan model pendeteksian	49
5.8.2	Pembuatan Web API	50
5.8.3	Pembuatan Antarmuka Web	51
5.9	Evaluasi Hasil	54
BAB VI KESIMPULAN DAN SARAN.....		57
6.1	Kesimpulan	57
6.2	Saran.....	57
DAFTAR PUSTAKA.....		59
LAMPIRAN		61



DAFTAR TABEL

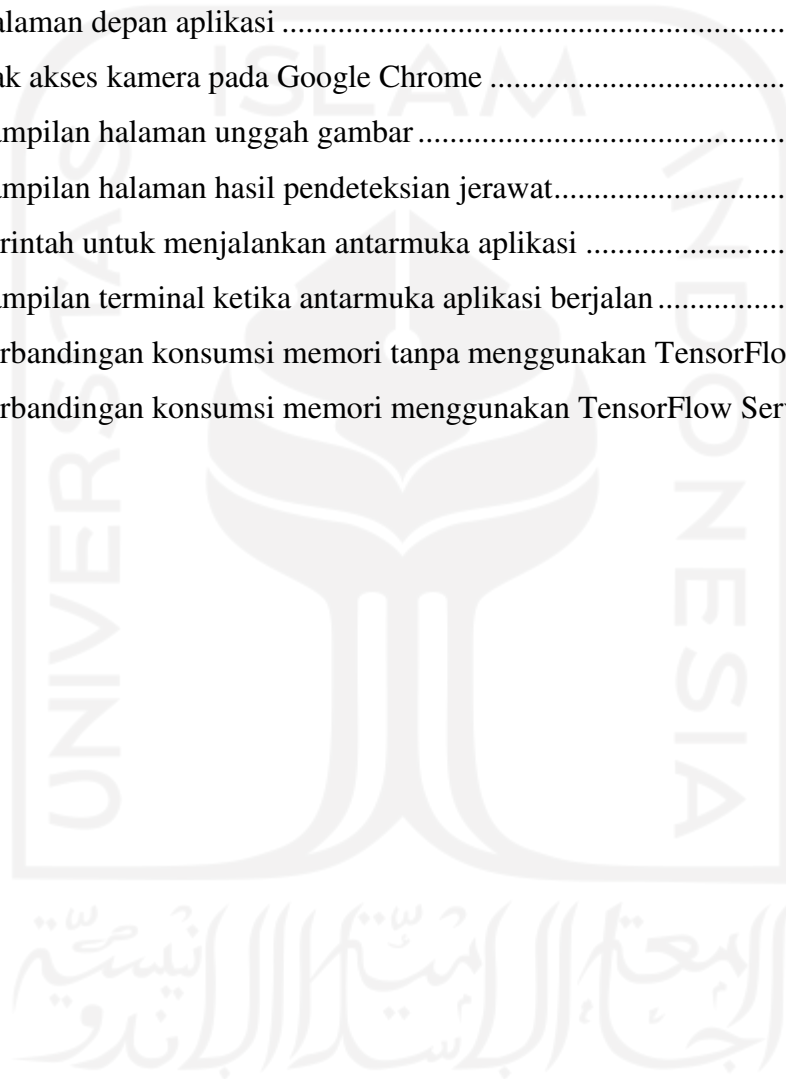
Tabel 2.1 Perbandingan dengan Penelitian Terdahulu	7
Tabel 3.1 Jenis jerawat pada wajah.....	10
Tabel 3.2 Penentuan keparahan jerawat berdasarkan jumlah lesi.....	11
Tabel 5.1 Tabel jenis augmentasi citra	37
Tabel 5.2 Daftar <i>library</i> Python yang digunakan	39
Tabel 5.3 Format isi berkas CSV	41
Tabel 5.4 Konfigurasi <i>pipeline</i> pelatihan model model.....	42
Tabel 5.5 Hasil percobaan pelatihan dengan jumlah <i>step</i> yang berbeda	43



DAFTAR GAMBAR

Gambar 3.1 Contoh jerawat pada wajah	9
Gambar 3.2 Bentuk gambar yang dibaca komputer.....	12
Gambar 3.3 Contoh <i>image classification</i>	14
Gambar 3.4 Contoh <i>semantic segmentation</i>	14
Gambar 3.5 Contoh <i>object detection</i>	15
Gambar 3.6 Contoh <i>instance segmentation</i>	16
Gambar 3.7 Ilustrasi saraf otak biologis (neuron) dan model matematikanya	18
Gambar 3.8 Satu lapisan konvolusi	20
Gambar 3.9 Lapisan konvolusi dengan lapisan <i>fully connected</i>	22
Gambar 3.10 Contoh pelabelan objek dengan labelImg.....	23
Gambar 3.11 Arsitektur jaringan SSD.....	25
Gambar 3.12 Penggunaan banyak <i>bounding box</i> lokasi sampel.....	25
Gambar 3.13 Ilustrasi perhitungan <i>intersection</i> dan <i>union</i>	28
Gambar 3.14 Perintah instalasi TensorFlow Serving pada Ubuntu.....	30
Gambar 3.15 Perintah instalasi TensorFlow Serving API.....	30
Gambar 3.16 Spesifikasi <i>server</i> Google Colab.....	31
Gambar 4.1 Tahapan penelitian	33
Gambar 5.1 Sampel dataset.....	34
Gambar 5.2 Proses segmentasi wajah dengan landmark wajah.....	35
Gambar 5.3 Contoh resolusi gambar jerawat setelah <i>resize</i>	36
Gambar 5.4 Contoh pelabelan objek jerawat dengan labelImg	36
Gambar 5.5 Isi dari <i>label map</i>	40
Gambar 5.6 Perintah untuk mengubah XML ke CSV data <i>train</i>	40
Gambar 5.7 Perintah untuk mengubah XML ke CSV data <i>test</i>	40
Gambar 5.8 Perintah untuk membuat TFRecord data <i>train</i>	41
Gambar 5.9 Perintah untuk membuat TFRecord data <i>test</i>	42
Gambar 5.10 Perintah untuk menjalankan pelatihan model	43
Gambar 5.11 Perintah untuk ekspor model.....	44
Gambar 5.12 Grafik <i>learning rate</i> pelatihan model.....	45
Gambar 5.13 Grafik <i>total loss</i> pelatihan model	45
Gambar 5.14 Struktur model <i>object detection</i>	46
Gambar 5.15 Contoh hasil pendeteksian jerawat.....	47

Gambar 5.16 Alur data aplikasi	47
Gambar 5.17 Rancangan antarmuka <i>upload</i> gambar	48
Gambar 5.18 Rancangan antarmuka hasil pendeteksian objek.....	49
Gambar 5.19 Perintah untuk menjalankan model dengan TensorFlow Serving	49
Gambar 5.20 Hasil pengecekan <i>port</i> yang terbuka dengan netstat	50
Gambar 5.21 Perintah untuk menjalankan web API.....	51
Gambar 5.22 Tampilan pada terminal ketika web API berjalan.....	51
Gambar 5.23 Halaman depan aplikasi	52
Gambar 5.24 Hak akses kamera pada Google Chrome	52
Gambar 5.25 Tampilan halaman unggah gambar	53
Gambar 5.26 Tampilan halaman hasil pendeteksian jerawat.....	53
Gambar 5.27 Perintah untuk menjalankan antarmuka aplikasi	54
Gambar 5.28 Tampilan terminal ketika antarmuka aplikasi berjalan	54
Gambar 5.29 Perbandingan konsumsi memori tanpa menggunakan TensorFlow Serving	55
Gambar 5.30 Perbandingan konsumsi memori menggunakan TensorFlow Serving.....	56



BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada saat ini atau di jaman sekarang memiliki kulit bagus menjadi keinginan banyak orang dan berpengaruh di pergaulan masyarakat. Seiring dengan meningkatnya klinik dermatologi, persaingan dalam metode dan teknologi untuk menyembuhkan kulit semakin kuat. Namun salah satu perawatan masalah kulit yang paling sering muncul saat ini adalah perawatan wajah berjerawat. Sebagian besar remaja dan bahkan beberapa orang dewasa mengalami masalah ini.

Jerawat merupakan permasalahan kulit yang umum. Jerawat muncul di bidang dermatologi untuk usia 15 - 40 tahun, 85% pada kelompok usia remaja dengan rentang usia 12 - 24 tahun (Masterson, 2018). Masalah jerawat muncul pada area wajah, dada dan punggung yang masing-masing memiliki prevalensi dan keparahan sebesar 92%, 45% dan 61% (Ramli et al., 2012). Jerawat paling sering terjadi pada wajah terutama area dahi, pipi, dan dagu karena berbagai faktor etiologi dari jerawat.

Asesmen jerawat merupakan prosedur awal yang secara umum dilakukan oleh dokter kulit sebelum memberikan tindakan perawatan terhadap kulit jerawat. Asesmen pada jerawat tidak ada prosedur penilaian yang dianggap sebagai standar global yang memunculkan masalah antar dan intra pengamat. Penilaian oleh dokter kulit yang berbeda menghasilkan penilaian yang berbeda untuk pasien yang sama dan terkadang dokter kulit yang sama juga memberikan penilaian yang berbeda untuk pasien yang sama pada hari yang berbeda. Selain itu, dokter kulit tidak memiliki cukup waktu untuk menghitung lesi jerawat dengan benar karena mereka harus menemui banyak pasien dalam waktu yang terbatas. Bagi sebagian dokter kulit, menghitung lesi jerawat itu membosankan karena berbagai jenis dan jumlah lesi yang ada. Oleh karena itu, mereka hanya memberikan perkiraan jumlah yang mengarah ke perkiraan tingkat keparahan jerawat (Ramli et al., 2012).

Sekarang ini teknologi digital sudah menjadi kebutuhan sehari-hari. Era dimana hampir semua aspek kehidupan manusia berhubungan erat dengan komputer. Perkembangan teknologi digital juga semakin pesat dengan bertumbuhnya kemampuan komputer melakukan komputasi. Salah satu pemanfaatannya adalah pengembangan kecerdasan buatan atau yang lebih dikenal dengan sebutan *Artificial Intelligence* (AI).

Pegembangan *artificial intelligence* salah satunya adalah pada bidang *computer vision*. *Computer vision* merupakan disiplin ilmu yang mengembangkan kemampuan komputer untuk mendapatkan informasi dari citra digital yang diamati. Di dalam *computer vision* terdapat beberapa bidang lagi yaitu *image classification*, *image segmentation*, dan *object detection*. *Object detection* (pendeteksian objek) merupakan salah satu bidang yang paling menarik untuk dikembangkan. Pendeteksian objek adalah kemampuan komputer untuk mendeteksi lokasi dan mengenali objek yang ada di dalam citra digital. Pendeteksian objek digunakan di berbagai bidang seperti kesehatan, agrikultur, *autonomous driving*, dan lain-lain. Implementasi *computer vision* ini sudah bisa ditemukan di beberapa aplikasi *desktop*, *mobile*, *embedded system*, dan web.

Aplikasi berbasis web mulai populer seiring dengan meningkatnya jumlah pengguna internet aktif di seluruh dunia. Selain memberikan kemudahan untuk mengaksesnya, aplikasi berbasis web juga tidak membutuhkan sumber daya yang besar dari sisi perangkat lunak maupun perangkat keras. Aplikasi berbasis web dapat diakses dengan mudah melalui *browser* dan koneksi internet atau intranet sehingga pengguna dapat mengakses melalui berbagai jenis perangkat komputer. Berbagai macam instansi menggunakan aplikasi berbasis web menjadi salah satu solusi teknologi informasi mereka. Perkembangan web *browser* sekarang ini sudah mendukung aplikasi web untuk terhubung dengan piranti perangkat keras secara langsung diantaranya kamera, mikrofon, USB, printer, dan lain-lain.

Berdasarkan uraian di atas, dalam penelitian ini akan dibuat sebuah aplikasi berbasis web untuk mendeteksi dan menghitung lesi jerawat pada wajah yang mengarah pada tingkat keparahan jerawat pada suatu citra digital. Adapun model pendeteksian yang digunakan pada aplikasi adalah model *deep learning* yaitu *Single Shot Detector* (SSD). Oleh karena itu, peneliti membuat penelitian yang berjudul “**Aplikasi Web Pendeteksi Jerawat Pada Wajah Menggunakan Model *Deep Learning* Dengan TensorFlow**”. Harapannya dengan penelitian ini mampu mengembangkan aplikasi berbasis web yang dapat digunakan untuk melakukan asesmen perawatan kulit di klinik dermatologi, dalam kasus ini adalah asesmen wajah berjerawat.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana model pendeteksian jerawat pada wajah menggunakan *deep learning*?

2. Bagaimana hasil model pendeteksian jerawat pada wajah menggunakan *deep learning*?
3. Bagaimana implementasi model pendeteksian jerawat pada wajah menggunakan *deep learning* pada aplikasi web?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini sebagai berikut:

1. Bahasa pemrograman yang digunakan adalah Python dengan *framework* Tensorflow.
2. Data yang digunakan dalam penelitian ini merupakan citra wajah yang memiliki masalah jerawat dengan resolusi gambar > 350px dan foto *close up* pada area wajah.
3. Dataset dikumpulkan dari *Google Image*, DermNet NZ, dan mengambil dari penelitian (Darmawan et al., 2020).
4. Metode yang digunakan adalah *deep learning* dengan model SSD Resnet50 V1 COCO.
5. Jumlah dataset yang digunakan berjumlah 200 gambar yang terdiri dari:
 - a. Data *Training*: data gambar yang digunakan untuk proses *training* berjumlah 160.
 - b. Data *Testing*: data gambar yang digunakan untuk proses *testing* berjumlah 40 gambar.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui model pendeteksian jerawat pada wajah menggunakan *deep learning* dengan model SSD ResNet50 V1 FPN 640x640.
2. Mengetahui hasil model pendeteksian jerawat pada wajah dengan menggunakan *deep learning* dengan model SSD ResNet50 V1 FPN 640x640.
3. Mengetahui cara implementasi model pendeteksian objek pada aplikasi web.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Dapat memudahkan suatu pihak dalam menentukan tingkat keparahan jerawat berdasarkan jumlah lesi jerawat yang terdeteksi pada wajah dengan bantuan teknologi, melalui model hasil pengujian penelitian ini.
2. Dengan diketahui cara mendeteksi jerawat pada wajah diharapkan dapat membuat perkembangan permasalahan *computer vision* lainnya.
3. Hasil penelitian ini dapat dijadikan acuan untuk penelitian lebih lanjut yang berbasis pada pendeteksian objek pada citra digital dan terkait penggunaan *deep learning*.



BAB II

TINJAUAN PUSTAKA

Sehubungan dengan penelitian yang dilakukan penulis, referensi dari penelitian terdahulu sangat penting untuk dilakukan agar terhindar dari plagiarisme atau duplikasi dari penelitian terdahulu, hal ini bertujuan juga sebagai bahan untuk kontribusi penelitian bagi penulis agar penelitian tentang tema ini terus berkembang. Berikut beberapa ulasan tentang penelitian terdahulu yang pernah dilakukan sebelumnya berkenaan dengan data dan metode yang digunakan.

Pada penelitian (Chantharaphaichit et al., 2015), pendeteksian jerawat dilakukan dengan *image processing*, *binary thresholding*. Gambar RGB diubah ke *grayscale* untuk dilakukan normalisasi dan ke HSV untuk mendapatkan nilai *brightness* (V). Kedua nilai tersebut digunakan untuk mendapatkan *region of interest* yang kemudian dilakukan *binary thresholding*. Hasil dari penelitian ini mendapatkan nilai sensitivitas 86.37%, presisi 80.00% dan akurasi 70.00% dari 10 gambar yang diuji.

Pada penelitian (Alamdari et al., 2016) menggunakan beberapa algoritma pendeteksian *K-means clustering*, *texture analysis* dan *color-base segmentation* sedangkan untuk klasifikasi menggunakan *Support Vector Machine* (SVM) dan FCM. Data pada penelitian ini hanya disebutkan menggunakan 35 gambar untuk melakukan *testing* atau validasi. Hasil dari segmentasi mendapatkan nilai akurasi lebih dari 70% dan pada klasifikasi mendapatkan nilai tertinggi sebesar 80%.

Pada penelitian (Prasetyo et al., 2018) menggunakan metode *connected-component labeling*, filter *Laplacian of Gaussian*, seleksi ciri luas area, *eccentricity*, *meanintensity* serta standar deviasi. Data yang digunakan berjumlah 38 gambar, dimana 2 gambar untuk mendapatkan karakteristik citra jerawat dan 36 gambar untuk pengujian. Hasil dari penelitian ini didapatkan nilai *sensitivity* sebesar 46.4%, dan *specificity* 97%.

Dalam penelitian (Patnaik et al., 2018) menggunakan beberapa algoritma *deep learning* Inception v2, Inception v3, dan MobileNet untuk mengklasifikasi 20 jenis masalah kulit. Pada penelitian tidak disebutkan jumlah data yang digunakan namun menggunakan perbandingan 90% data untuk *training* dan 10% untuk *testing* dan validasi. Hasil dari penelitian ini mendapatkan nilai akurasi tertinggi sebesar 88%.

Pada penelitian (Junayed et al., 2019) menggunakan algoritma CNN berbasis Deep Residual Neural Network untuk membuat model klasifikasi jenis lesi jerawat. Penelitian ini

menggunakan dataset 5 kelas penyakit jerawat dari dermnet.com dengan jumlah 1800 gambar, 360 untuk masing-masing kelas dan menggunakan perbandingan 80% untuk *training* serta 20% untuk *testing*. Hasil dari penelitian ini dapat menghasilkan akurasi klasifikasi sebesar 95.89%.

Pada penelitian (Zhao et al., 2019), peneliti bekerja sama dengan Nestlé Skin Health SHIELD untuk membuat model *deep learning* yang dapat mengetahui keparahan jerawat dari foto *selfie*. Pada penelitian ini disebutkan menggunakan 4.700 gambar sudah diberi label oleh 11 dokter kulit internal untuk 5 kategori keparahan, dari 1 (bersih) ke 5 (berat) dan menggunakan 230 gambar untuk *testing* untuk menghitung RMSE. Penelitian ini menggunakan model *deep learning* ResNet 152 untuk melakukan *transfer learning* dan OpenCV untuk mengekstrak area-area dari wajah dengan tujuan mengurangi *background* yang tidak relevan. Hasil dari penelitian ini mendapatkan nilai RMSE sebesar 0.482 dan paling baik untuk mendeteksi keparahan jerawat ringan.

Penelitian mengenai penggunaan *deep learning* untuk pendeteksian jerawat juga dilakukan oleh (Rashataprucksa et al., 2020). Tujuan dari penelitian ini adalah untuk mengatasi performa dari model pendeteksian jerawat tradisional. Penelitian ini menggunakan model Faster-RCNN dan R-FCN. Data berjumlah 871 gambar yang sudah diberi label dengan 4 kategori jerawat, 783 gambar untuk *training* dan 88 gambar untuk *testing*. Hasil dari penelitian ini didapatkan nilai mAP untuk Faster R-CNN sebesar 23.3% dan R-FCN sebesar 28.3%.

Tabel 22.1 Perbandingan dengan Penelitian Terdahulu

No.	Penulis	Judul Penelitian	Metode Penelitian	Hasil
1	Chantharaphaichit et al. (2015)	<i>Automatic Acne Detection for Medical Treatment</i>	<i>binary thresholding</i>	Gambar RGB diubah ke <i>grayscale</i> untuk dilakukan normalisasi dan ke HSV untuk mendapatkan nilai <i>brightness</i> (V). Kedua nilai tersebut digunakan untuk mendapatkan <i>region of interest</i> yang kemudian dilakukan <i>binary thresholding</i> . Hasil dari penelitian ini mendapatkan nilai sensitivitas 86.37%, presisi 80.00% dan akurasi 70.00% dari 10 gambar yang diuji.
2	Alamdari et al. (2016)	<i>Detection and Classification of Acne Lesions in Acne Patients: A Mobile Application</i>	<i>K-means clustering, texture analysis dan color-base segmentation, SVM dan FCM</i>	Data pada penelitian ini hanya disebutkan menggunakan 35 gambar untuk melakukan <i>testing</i> atau validasi. Hasil dari segmentasi mendapatkan nilai akurasi lebih dari 70% dan pada klasifikasi mendapatkan nilai tertinggi sebesar 80%.
3	Prasetyo et al. (2018)	Aplikasi Pendeteksi Jerawat Di Wajah Dengan Menggunakan Teknik Pengolahan Citra Pada Foto	<i>connected-component labeling, filter Laplacian of Gaussian, seleksi ciri luas area, eccentricity, meanintensity dan standar deviasi</i>	Data yang digunakan berjumlah 38 gambar, dimana 2 gambar untuk mendapatkan karakteristik citra jerawat dan 36 gambar untuk pengujian. Hasil dari penelitian ini didapatkan nilai <i>sensitivity</i> sebesar 46.4%, dan <i>specificity</i> 97%.
4	Patnaik et al. (2018)	<i>Automated Skin Disease Identification using Deep Learning Algorithm</i>	Model <i>deep learning</i> Inception v2, Inception v3, dan MobileNet	Pada penelitian tidak disebutkan jumlah data yang digunakan namun menggunakan perbandingan 90% data untuk <i>training</i> dan 10% untuk <i>testing</i> dan validasi. Hasil

				dari penelitian ini mendapatkan nilai akurasi tertinggi sebesar 88%.
5	Junayed et al. (2019)	<i>AcneNet – A Deep CNN Based Classification Approach for Acne Classes</i>	<i>CNN berbasis Deep Residual Neural Network</i>	Penelitian ini menggunakan dataset 5 jenis penyakit jerawat dari dermnet.com dengan jumlah 1800 gambar, 360 gambar untuk masing-masing kelas dan menggunakan perbandingan 80% untuk <i>training</i> serta 20% untuk <i>testing</i> . Hasil dari penelitian ini dapat menghasilkan akurasi klasifikasi sebesar 95.89%.
6	Zhao et al. (2019)	<i>A Computer Vision Application for Assessing Facial Acne Severity from Selfie Image</i>	<i>Facial Landmark detection, rolling skin patches, transfer learning model deep learning ResNet-152</i>	Penelitian ini menggunakan 4.700 gambar sudah diberi label oleh 11 dokter dengan 5 kategori keparahan, dari 1 (bersih) ke 5 (berat) dan menggunakan 230 gambar untuk <i>testing</i> . Penelitian ini mendapatkan nilai RMSE sebesar 0.482 dan paling baik untuk mendeteksi keparahan jerawat ringan.
7	Rashataprucksa et al. (2020)	<i>Acne Detection with Deep Neural Networks</i>	<i>Deep Learning menggunakan Deep Neural Networks dengan model Faster R-CNN dan R-FCN</i>	Hasil dari penelitian ini didapatkan nilai mAP untuk model Faster R-CNN sebesar 23.3% dan model R-FCN sebesar 28.3%

BAB III

LANDASAN TEORI

3.1 Pengetahuan Dasar Tentang Jerawat (*acne vulgaris*)

Jerawat terjadi ketika folikel rambut atau sering disebut pori-pori kulit tersumbat. Kelenjar minyak kecil yang disebut kelenjar *sebaceous* yang terletak di sekitar folikel rambut, menghasilkan zat berminyak yang disebut sebum. Sebum, yang biasanya mengalir ke permukaan melalui folikel rambut, terperangkap di dalam pori-pori kulit ketika folikel rambut tersumbat (Ramli et al., 2012). Dari sini, bakteri yang dikenal sebagai *Propionibacterium acnes* akhirnya menyerang sebum, sehingga menghasilkan peradangan kulit dan jerawat.






Menurut (Ramli et al., 2012) mengungkapkan bahwa lesi primer pada jerawat dapat dibagi menjadi dua kategori utama yaitu inflamasi dan non inflamasi. Lesi inflamasi terdiri dari papula, pustula, dan nodul sedangkan non inflamasi terdiri dari komedo terbuka (*whitehead*) dan komedo tertutup (*blackhead*). Papula biasanya muncul sebagai lesi eritema (bercak kemerahan) yang bervariasi dengan ukuran kurang dari 5 mm. Pustula, umumnya adalah papula berisi cairan berwarna putih dengan ukuran 5 mm atau kurang. Nodul adalah lesi yang lebih besar (5 mm atau lebih) juga dengan eritema yang bervariasi. Komedo tertutup memiliki kulit yang tumbuh di atas pori-pori yang tersumbat sehingga terlihat seperti benjolan putih kecil. Komedo yang terbuka, terlihat seperti pori-pori yang membesar dan menghitam. Berikut ini pada Gambar 3.1 merupakan contoh jerawat yang muncul pada wajah.



Gambar 3.1 Contoh jerawat pada wajah

Seperti yang dijelaskan pada paragraf di atas jerawat memiliki beberapa jenis. Berikut ini pada Tabel 3.1 merupakan jenis jerawat yang terdapat pada wajah.

Tabel 3.1 Jenis jerawat pada wajah

No.	Gambar	Keterangan
1		<i>Blackhead</i> (komedo terbuka) Sedikit terangkat dari kulit, tetapi tidak ada peradangan
2		<i>Whitehead</i> (komedo tertutup) Terbentuk ketika pori-pori tersumbat oleh sebum dan sel kulit mati. Pori-pori tidak terbuka di bagian atas.
3		<i>Papules</i> (papula) Papula berwarna merah, benjolan menyakitkan yang disebabkan oleh peradangan pada folikel rambut
4		<i>Pustules</i> (pastula) Pustula terbentuk beberapa hari setelah sel darah putih dalam papula muncul ke permukaan kulit
5		<i>Nodules</i> (nodul) Nodul sering besar, meradang, merah, bengkak dan nyeri saat disentuh.
6		<i>Cyst</i> (kista) jenis jerawat yang parah di mana pori-pori di kulit tersumbat, menyebabkan infeksi dan peradangan

3.2 Keparahan Jerawat

Menurut (Lucky et al., 1996) menilai keparahan lesi jerawat dan mereka menghitungnya. Kemudian mereka mencatat pada *template* wajah dan membaginya menjadi lima segmen wajah, yaitu dahi kanan dan kiri, pipi kanan dan kiri, dan dagu. Setiap jenis lesi seperti komedo terbuka, komedo tertutup, papula, pustula dan nodul dalam setiap segmen *template*

dihitung. Mereka menilai berdasarkan klasifikasi tingkat lima jerawat, sangat ringan untuk pasien yang hanya memiliki sedikit komedo, ringan untuk pasien yang memiliki lebih banyak komedo dan sedikit papula dan pustula, sedang untuk pasien yang memiliki banyak papula dan pustula, parah untuk pasien yang memiliki banyak papula dan pustula. Papula dan pustula dengan beberapa nodul dan sangat parah untuk pasien yang memiliki beberapa komedo, papula, pustula dan nodul.

Penelitian yang dilakukan (Hayashi et al., 2008) menggunakan foto standar dan penghitungan lesi untuk mengklasifikasikan jerawat menjadi empat kelompok. Pertama, mereka mengklasifikasikan jerawat berdasarkan jumlah erupsi inflamasi setengah wajah. Kedua, mereka menghitung lesi dan membagi jumlah total lesi menjadi empat kelompok. Untuk jumlah total lesi <5 , derajat diklasifikasikan sebagai ringan dan 6-20 sebagai sedang. Jika jumlah total lesi antara 21 dan 50, itu termasuk dalam kelompok parah dan lebih dari 50 sangat parah seperti yang ditunjukkan pada Tabel 3.2 berikut.

Tabel 3.2 Penentuan keparahan jerawat berdasarkan jumlah lesi

Jumlah Lesi	Kategori
1-5	Ringan
6-20	sedang
21-50	Parah
>50	Sangat Parah

3.3 Pengolahan Citra Digital

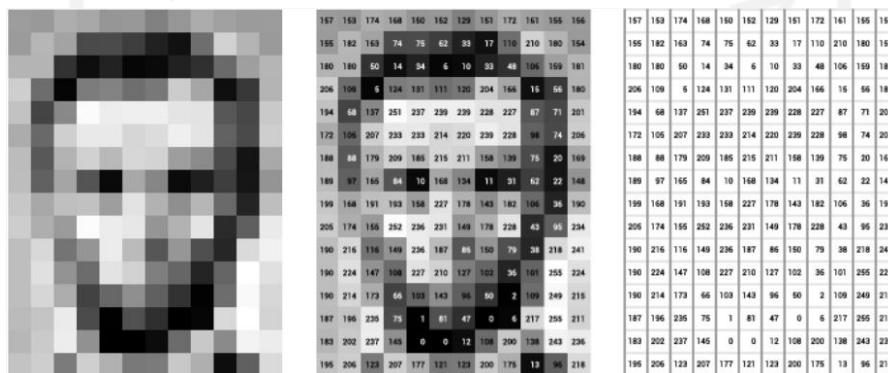
Pengolahan citra merupakan metode untuk melakukan beberapa operasi pada citra, untuk mendapatkan citra yang disempurnakan atau untuk mengekstrak beberapa informasi yang berguna. Menurut (Padilla et al., 2020) menyatakan bahwa pengolahan citra digital merupakan teknik pemrosesan sinyal digital khusus pada gambar. Sebuah citra dapat dianggap sebagai fungsi $f(x, y)$ dari dua variabel kontinu x dan y . Untuk diproses secara digital, citra harus diambil dan diubah menjadi matriks angka.

Pengolahan citra digital dapat dibagi menjadi beberapa kelas: *image enhancement* (peningkatan citra), *image restoration* (restorasi citra), *image analysis* (analisis citra), dan *image compression* (kompresi citra). Informasi pada citra dapat didapatkan dengan menggunakan teknik analisis citra yang memungkinkan sebuah citra diproses sehingga

informasi dapat diekstraksi secara otomatis. Contoh analisis citra adalah segmentasi citra, ekstraksi tepi, dan analisis tekstur dan gerak.

3.3.1 Definisi Citra Digital

Citra digital adalah representasi dari citra nyata sebagai sekumpulan angka yang dapat disimpan dan ditangani oleh komputer. Citra digital dibagi menjadi area kecil yang disebut piksel (elemen gambar). Setiap piksel bersisi angka yang menjelaskan kecerahan (intensitas cahaya) atau warna. Angka-angka tersebut tersusun dalam bentuk matriks berukuran $m \times n$ yang mewakili ukuran lebar dan tinggi dari citra, seperti yang ditunjukkan pada Gambar 3.2.



Gambar 3.2 Bentuk gambar yang dibaca komputer

3.3.2 Jenis Citra Digital

Citra digital pada umumnya dikategorikan dalam tiga jenis, yaitu:

1. Citra Biner

Citra biner atau dalam pengolahan citra sering disebut *binary image* adalah citra yang hanya mempunyai dua nilai yaitu hitam atau putih. Nilai piksel pada gambar hanya berisi angka 1 untuk warna putih dan angka 0 untuk warna hitam.

2. Citra Keabuan

Citra keabuan atau dalam pengolahan citra disebut sebagai citra *grayscale* adalah citra yang hanya mempunyai satu lapisan warna. Setiap piksel pada citra memiliki rentang antara 0 – 255, angka 0 merepresentasikan nilai warna paling gelap atau hitam sedangkan angka 255 merepresentasikan nilai warna paling terang atau putih.

3. Citra Warna

Citra warna merupakan citra yang secara umum memiliki tiga lapisan warna. Sistem pewarnaan citra warna ada beberapa macam seperti RGB, HSV, CMYK, dan lain-

lain. Sebagai contoh citra RGB, terdiri dari tiga lapisan warna *red*, *green*, dan *blue*. Citra warna RGB biasanya digunakan pada monitor komputer dan *scanner* sedangkan CMYK digunakan pada *printer*.

3.3.3 *Feature Extraction*

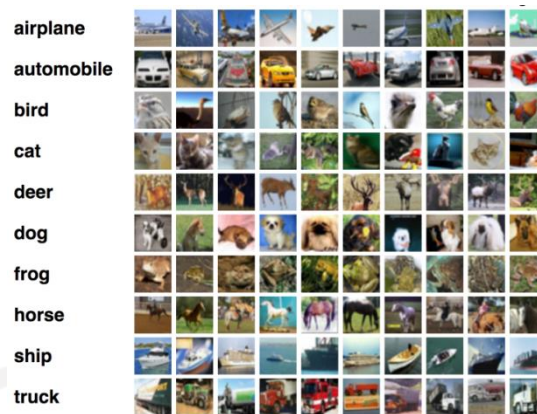
Feature extraction (ekstraksi ciri) merupakan proses untuk mengurangi jumlah sumber daya yang dibutuhkan untuk mendeskripsikan sebuah kumpulan data yang besar. Di dalam *computer vision* dan *image processing*, *feature* (ciri) adalah potongan informasi tentang isi dari sebuah citra, dapat berupa titik, garis, atau objek. Hasil dari ekstraksi ciri dikenal dengan sebutan *feature vector* (vector ciri).

3.4 *Computer Vision*

Computer vision adalah bidang ilmiah yang membahas bagaimana komputer dapat memperoleh pemahaman tingkat tinggi dari gambar atau video digital. Dari perspektif teknik, *computer vision* berusaha untuk memahami dan mengotomatisasi tugas-tugas yang dapat dilakukan oleh sistem visual manusia. Tugas dari *computer vision* mencakup metode untuk memperoleh, memproses, menganalisis dan memahami citra digital, dan mengekstraksi data untuk menghasilkan informasi numerik atau simbolis.

3.4.1 *Image Classification*

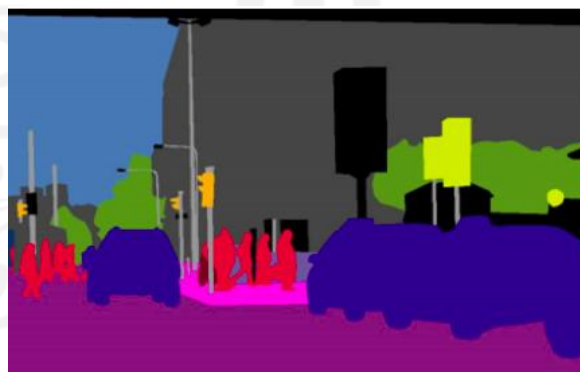
Image classification adalah tugas dasar dari *computer vision* untuk mengkategorikan suatu gambar kedalam suatu kategori dari kumpulan kategori-kategori (misalnya, kucing, anjing, mobil, pesawat), seperti yang ditunjukkan pada Gambar 3.3. Tujuannya adalah agar komputer dapat mengenali objek yang ada pada gambar. Metode yang digunakan untuk tugas klasifikasi diantaranya *Support Vector Machine* (SVM), *Decision Tree*, *K-nearest Neighbor*, *Artificial Neural Network* (ANN) dan *Convolutional Neural Network* (CNN).



Gambar 3.3 Contoh *image classification*

3.4.2 *Semantic Segmentation*

Semantic segmentation mengacu pada proses yang menghubungkan setiap piksel pada gambar ke dalam suatu label kelas yang sesuai, seperti klasifikasi gambar namun pada tingkat piksel. *Semantic segmentation* tidak memisahkan *instance* dari kelas yang sama, ini berarti bahwa jika ada dua objek dari kategori yang sama pada gambar maka peta segmentasi tidak secara inheren membedakannya sebagai objek yang terpisah. Contoh penerapannya dapat dilihat pada Gambar 3.4. *Semantic segmentation* digunakan pada robotika, *autonomous driving*, agrikultur, dan diagnosa citra medis. Beberapa metode *deep learning* yang populer adalah *Fully Convolutional Network* (FCN), U-Net, Mask R-CNN, dan DeepLab.

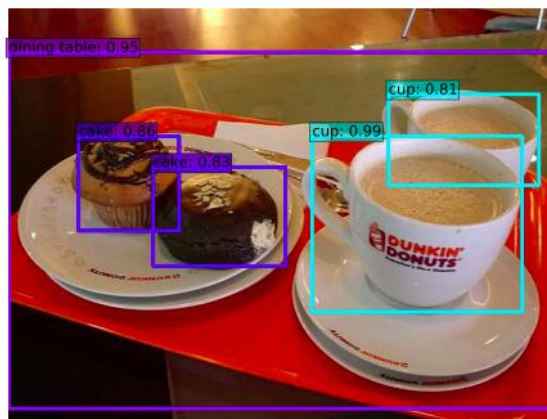


Gambar 3.4 Contoh *semantic segmentation*

3.4.3 *Object Detection*

Object detection (deteksi objek) adalah salah satu tugas penting pada *computer vision* yang berhubungan dengan pendeteksian objek visual dari kelas tertentu (seperti manusia, hewan, atau mobil) dalam citra digital. *Object detection* juga merupakan dasar untuk tugas

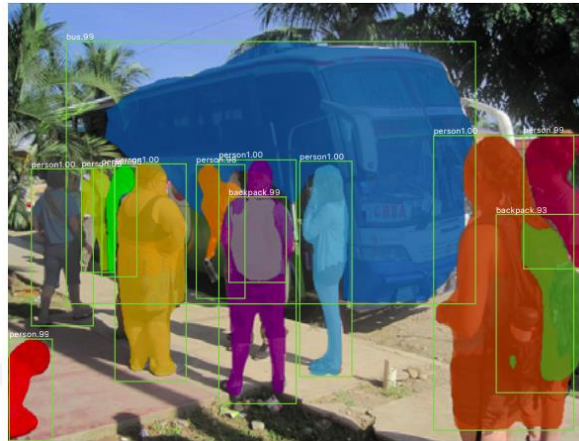
computer vision lainnya, seperti *instance segmentation*, *image captioning*, *object tracking*, dan lain-lain. Kebanyakan algoritma yang dikembangkan sekarang ini adalah deteksi objek berbasis *deep learning*. Jenis pembuatan arsitektur deteksi objek dibagi menjadi dua kategori, *two-stage detector* dan *one-stage detector*. Kerangka *two-stage detector* membagi proses menjadi tahap *region proposal* dan tahap klasifikasi, Faster R-CNN adalah salah satu yang paling terkenal. Di sisi lain, *one-stage detector* dalam melakukan prediksi lokasi dan klasifikasi dilakukan dalam satu tahap sekaligus, YOLO dan SSD menjadi arsitektur yang terkenal (Manuel et al, 2021). Contoh deteksi objek dapat dilihat pada Gambar 3.5.



Gambar 3.5 Contoh *object detection*

3.4.4 *Instance Segmentation*

Instance segmentation merupakan gabungan tugas dari *semantic segmentation* dan *object detection*. Selain mendeteksi objek, *instance segmentation* juga memprediksi setiap piksel yang membentuk objek tersebut. Tujuan dari *instance segmentation* adalah untuk mendapatkan area objek dari kelas yang sama, dibagi menjadi *instance* yang berbeda. Arsitektur model yang populer adalah Mask R-CNN. Contoh penerapannya dapat dilihat pada Gambar 3.6.



Gambar 3.6 Contoh *instance segmentation*

3.5 Machine Learning

Menurut (Andriy, 2019) *machine learning* adalah sub bidang ilmu komputer yang berkaitan dengan pembuatan algoritma yang mengandalkan kumpulan contoh dari beberapa fenomena. Fenomena dapat berasal dari alam, buatan tangan manusia atau algoritma yang lain. *Machine learning* dapat didefinisikan sebagai proses untuk menyelesaikan sebuah permasalahan dengan mengumpulkan kumpulan data dan secara algoritma membangun model statistik berdasarkan kumpulan data tersebut. Terdapat beberapa istilah pembelajaran yang meliputi *supervised*, *semi-supervised*, *unsupervised*, dan *reinforcement*.

3.5.1 Supervised Learning

Pada tipe *supervised learning* dataset yang digunakan adalah kumpulan data yang telah diberi label. Tujuan dari *supervised learning* adalah menggunakan dataset untuk menghasilkan model yang mengambil vektor fitur sebagai informasi *input* dan *output* yang memungkinkan menyimpulkan label untuk vektor fitur tersebut.

3.5.2 Unsupervised Learning

Pada tipe *unsupervised learning* dataset yang digunakan adalah kumpulan data yang tidak diberi label. Tujuan dari *unsupervised learning* adalah untuk membuat model yang mengambil vektor fitur sebagai *input* dan mengubahnya menjadi vektor atau nilai lain yang dapat digunakan untuk memecahkan masalah praktis.

3.5.3 *Semi-Supervised Learning*

Pada tipe *semi-supervised learning* dataset yang digunakan terdiri dari data yang sudah diberi label dan yang tidak diberi label. Biasanya, jumlah dari data yang tidak diberi label lebih tinggi dibandingkan data yang diberi label. Tujuan *semi-supervised learning* sama dengan tujuan dari tipe *supervised learning*.

3.5.4 *Reinforcement Learning*

Pada *reinforcement learning* adalah subbidang dari *machine learning* di mana mesin “hidup” di dalam sebuah lingkungan dan mampu memahami keadaan lingkungan itu sebagai vektor fitur. Mesin dapat menjalankan aksi di setiap keadaan. Aksi yang berbeda dapat membawa *reward* yang berbeda dan juga dapat memindahkan mesin ke kondisi lingkungan yang lain. Tujuan dari algoritma *reinforcement learning* adalah untuk mempelajari kebijakan.

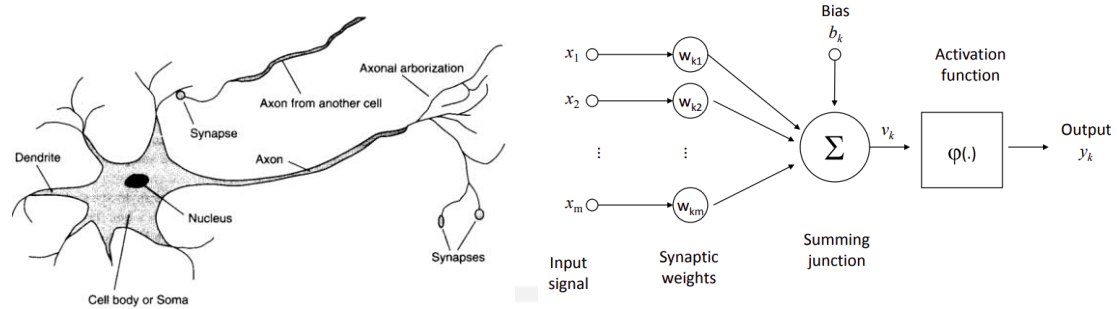
3.6 *Deep Learning*

Deep learning merupakan bagian dari *machine learning* yang meniru cara otak manusia mendapatkan pengetahuan tertentu (*learning*). Otak manusia terdiri dari banyak neuron, begitu juga dengan *deep learning* yang tersusun dari tiga atau lebih lapisan *Artificial Neural Network* (ANN). Menurut (Goodfellow et al., 2015) algoritma *deep learning* tersusun dalam bentuk hierarki untuk meningkatkan kompleksitas dan abstraksi. Masing-masing program dalam hierarki menerapkan transformasi non linear pada setiap *inputnya* dan menggunakannya untuk membuat model statistic sebagai *output*.

3.6.1 *Artificial Neural Network* (ANN)

Artificial Neural Network (ANN) pertama kali dirancang oleh Warren McCulloch dan Walter Pitts pada tahun 1943 yang dikenal dengan McCulloch-Pitts neurons tentang dua neuron aktif secara bersamaan kemudian kekuatan tersebut terkoneksi antara neuron yang seharusnya bertambah. Kemudian pada tahun 1957, Frank Rosenblatt mengenalkan dan mengembangkan sekumpulan besar jaringan saraf tiruan yang disebut *perceptrons*.

Menurut (Kusumadewi, 2003) ANN adalah model komputasi yang meniru jaringan saraf pada otak manusia. Otak manusia tersusun dari sel-sel saraf (neuron) dan penghubung (sinapsis). Neuron bekerja berdasarkan sinyal yang diterima oleh dendrit. Sinyal tersebut kemudian dijumlahkan pada soma (*cell body*), jika jumlah sinyal memenuhi *threshold* maka sinyal diteruskan diteruskan ke neuron lainnya melalui axon.



Gambar 3.7 Ilustrasi saraf otak biologis (neuron) dan model matematikanya

Pada Gambar 3.7 menunjukkan bahwa informasi x_i yang datang dari neuron lain (atau *input*) diterima di dendrit. Secara khusus, informasi tersebut diberi bobot oleh bobot sinaptik dengan menentukan efek *input*. *Input* berbobot yang datang dikumpulkan dalam inti sebagai jumlah bobot dan informasi ini kemudian dikirim untuk diproses lebih lanjut di akson y . Dari situ proses mencapai tujuannya (misalnya, otot) atau dimasukkan ke neuron lain melalui dendritnya.

3.6.2 Forward Propagation

Forward Propagation mengacu pada proses yang terjadi pada *feed forward* network dimana sebuah *input* x dapat menghasilkan *output* y , informasi mengalir ke depan dari *input layer* menuju *output layer*. *Input* x memberikan informasi awal yang kemudian menyebar ke *hidden layer* dan akhirnya menghasilkan *output* y (Goodfellow et al., 2015).

3.6.3 Back Propagation

Back propagation, sering disebut *backprop*, merupakan metode atau algoritma untuk menghitung gradien pada *neural network*. *Backprop* memungkinkan nilai *error output* mengalir dari layer *output* ke layer pertama (arah mundur) untuk meminimalkan fungsi biaya dengan menyesuaikan nilai bobot dan bias *neural network*. Algoritma ini berjalan paling tidak setelah terjadinya satu kali *forward propagation*. Gradien ini dapat diartikan sebagai indikasi bagaimana *output* setiap layer harus berubah untuk mengurangi *error*. Gradien pada bobot dan bias dapat digunakan sebagai bagian dari pembaruan gradien stokastik (melakukan pembaruan tepat setelah gradien dihitung) atau dengan metode optimasi berbasis gradien lainnya (Goodfellow et al., 2015).

3.6.4 Activation Function

Menurut (Zhang et al., 2021) *activation function* atau fungsi aktivasi merupakan fungsi memutuskan apakah suatu neuron harus diaktifkan atau tidak dengan menghitung jumlah bobot dan penambahan bias. Berikut ini beberapa fungsi aktivasi yang sering digunakan pada *deep learning*.

1. ReLU

Rectified Linear Unit (ReLU) merupakan fungsi aktivasi yang paling populer. ReLU menyediakan transformasi non linear yang sederhana. ReLU hanya mempertahankan elemen positif dan membuang semua elemen negatif dengan mengatur aktivasi ke 0. Bila mendapatkan *input* positif, turunannya hanya 1, dengan kata lain, aktivasi hanya *men-threshold* pada nilai nol, seperti pada persamaan (3.1).

$$ReLU(x) = \max(x, 0) \quad (3.1)$$

2. Sigmoid

Sigmoid merupakan fungsi aktivasi yang mengubah *input*-nya menjadi *output* yang terletak pada rentang 0 sampai 1. *Sigmoid* masih banyak digunakan sebagai fungsi aktivasi pada unit *output*, ketika ingin menafsirkan *output* sebagai probabilitas untuk masalah klasifikasi biner. Namun, *sigmoid* sebagian besar telah digantikan oleh ReLU yang lebih sederhana dan lebih mudah dilatih di *hidden layer*. Fungsi *sigmoid* ditunjukkan pada persamaan (3.3).

$$sigmoid(x) = \frac{1}{1 + \exp(-x)} \quad (3.2)$$

3. TanH

Seperti fungsi *sigmoid*, fungsi tanh (*hyperbolic tangent*) juga menekan *input*-nya, mengubahnya menjadi elemen pada interval antara -1 dan 1, ditunjukkan pada persamaan (3.3). Meskipun bentuk fungsinya mirip dengan fungsi *sigmoid*, fungsi tanh menunjukkan titik simetri tentang asal usul sistem koordinat.

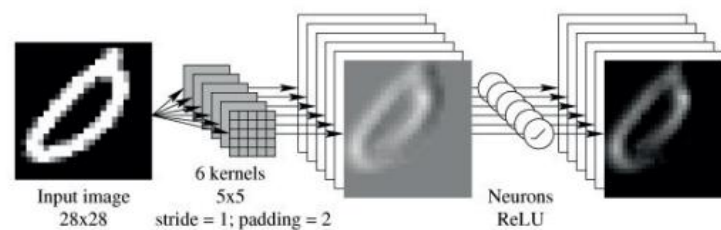
$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \quad (3.3)$$

3.7 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan salah satu jenis dari *neural network* (jaringan saraf tiruan) yang menunjukkan kinerja tinggi pada tugas visual, termasuk *image classification*, *image segmentation*, *object detection*, *face recognition* dan lain-lain. CNN dirancang untuk memproses data dalam bentuk multi *array*, misalnya, citra berwarna yang terdiri dari tiga larik 2D yang berisi intensitas piksel dalam tiga saluran warna. CNN menggunakan filter *convolutional* untuk mengekstrak informasi dari gambar, lapisan awal mendeteksi tepi, lapisan selanjutnya mendeteksi bagian dari objek, dan lapisan selanjutnya dapat mendeteksi objek lengkap, seperti wajah atau bentuk geometris kompleks lainnya (Bezdan & Bačanin, 2019).

3.7.1 Convolution Layer (Conv. Layer)

Operasi *convolution* merupakan satu blok fundamental CNN. Parameter lapisan *convolution* terdiri dari satu set filter atau yang disebut sebagai kernel. Setiap filter memiliki panjang, lebar, dan kedalaman. Umumnya ukuran filter adalah 3x3, 5x5, 7x7 sedangkan untuk kedalaman sesuai dengan *input* citra, citra *grayscale* adalah 1 dan citra warna adalah 3 (RGB). Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra *input*.



Gambar 3.8 Satu lapisan konvolusi

Sumber: (Bezdan & Bačanin, 2019)

Selama *forward propagation*, setiap filter melakukan konvolusi pada setiap posisi pada citra *input*, diikuti dengan fungsi aktivasi nonlinear (*sigmoid*, *tanh*, *ReLU*, dll), *output* yang dihasilkan disebut *feature map*. *Feature map* (juga dikenal sebagai peta aktivasi), memberikan respon filter pada setiap posisi spasial. Contoh dari lapisan konvolusi yang diikuti fungsi aktivasi non linear dapat dilihat pada Gambar 3.8. Peta aktivasi ditumpuk

sepanjang kedalaman dan menghasilkan volume *output*. Volume dari *output* bergantung pada tiga hyperparameter: *depth*, *stride*, dan *padding*. Ukuran *output* (S) dapat dihitung dengan persamaan (3.4), di mana jumlah filter (n), jumlah padding (p), ukuran filter (f), dan *stride* (s).

- *Depth* dari volume *output* merepresentasikan jumlah dari filter yang digunakan pada operasi konvolusi. Masing-masing filter mempelajari sesuatu yang berbeda dari *input*, *edges* (tepi), *blobs* (gumpalan), dan warna.
- *Stride* adalah jumlah langkah filter untuk bergeser ke posisi selanjutnya pada *input*. Ketika langkahnya adalah 1 maka filter berpindah satu piksel pada satu waktu. Ketika langkahnya adalah 2 maka filter melompat 2 piksel sekaligus saat bergeser. Hal ini menghasilkan volume *output* yang lebih kecil secara spasial.
- *Padding* adalah parameter yang menentukan jumlah piksel berisi nilai nol yang ditambahkan di setiap sisi dari *input*. Hal ini digunakan dengan tujuan untuk menghindari hilangnya informasi karena operasi konvolusi.

$$S = \frac{(n + 2p - f)}{s} + 1 \quad (3.4)$$

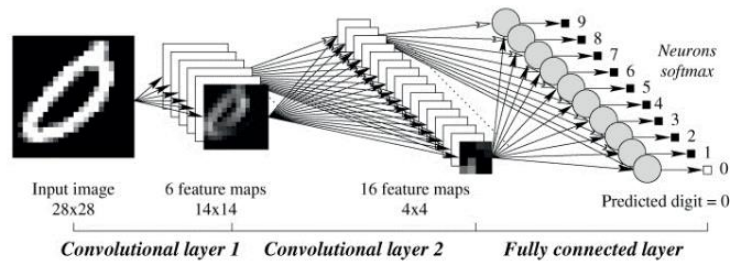
3.7.2 Pooling Layer

CNN sering menggunakan operasi *pooling layer* setelah lapisan konvolusi, fungsinya untuk memperkecil dimensi, disebut juga sebagai *subsampling* atau *downsampling*. *Hyperparameters* dari *pooling layer* mewakili ukuran dan *stride* dari filter. *Pooling layer* yang paling umum digunakan adalah dengan filter ukuran 2 dan dengan *stride* 2. Dua jenis dari *pooling layer* adalah *max pooling* dan *average pooling*, dimana nilai maksimum dan rata-rata diambil, masing-masing. *Max pooling* lebih sering digunakan daripada *average pooling*.

3.7.3 Fully Connected Layer

Setelah beberapa lapisan konvolusi dan *pooling layer*, CNN umumnya diakhiri dengan beberapa *fully connected layer*. Tensor yang dimiliki pada *output* pada lapisan ini ditransformasikan menjadi vektor dan kemudian ditambahkan beberapa lapisan *neural network*. Lapisan *fully connected* biasanya adalah beberapa lapisan terakhir dari arsitektur seperti yang ditunjukkan pada Gambar 3.9, teknik regularisasi dropout dapat diterapkan pada

fully connected untuk mencegah *overfitting*. Lapisan akhir yang sepenuhnya terhubung dalam arsitektur berisi jumlah neuron keluaran yang sama dengan jumlah kelas yang akan dikenali.



Gambar 3.9 Lapisan konvolusi dengan lapisan *fully connected*

Sumber: (Bezdan & Bačanin, 2019)

3.8 Object Detection

Object Detection (deteksi objek) adalah salah satu tugas penting pada *computer vision* yang berhubungan dengan pendeteksian objek visual dari kelas tertentu (seperti manusia, hewan, atau mobil) dalam citra digital. *Object Detection* juga merupakan dasar untuk tugas *computer vision* lainnya, seperti *instance segmentation*, *image captioning*, *object tracking*, dan lain-lain. Kebanyakan algoritma yang dikembangkan sekarang ini adalah deteksi objek berbasis *deep learning* (Zou et al., 2019).

Jenis pembuatan arsitektur deteksi objek dibagi menjadi dua kategori, *two-stage detector* dan *one-stage detector*. Kerangka *two-stage detector* membagi proses menjadi tahap *region proposal* dan tahap klasifikasi, Faster R-CNN adalah salah satu yang paling terkenal. Di sisi lain, *one-stage detector* dalam melakukan prediksi lokasi dan klasifikasi dilakukan dalam satu tahap sekaligus, YOLO dan SSD menjadi arsitektur yang terkenal (Carranza-García et al., 2021).

3.9 Image Augmentation

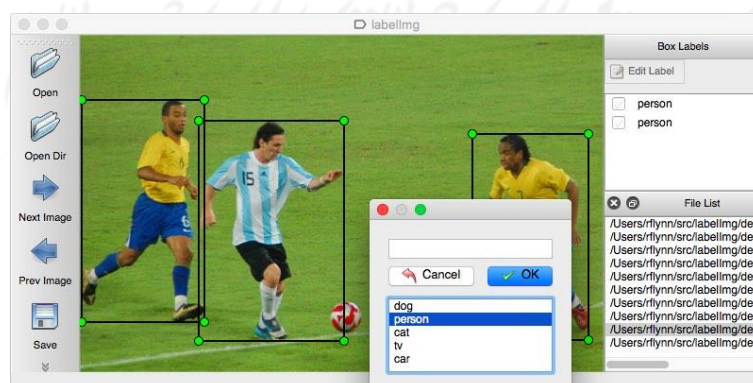
Image augmentation (augmentasi citra) adalah sebuah teknik untuk mengubah data yang dimiliki untuk membuat lebih banyak data untuk proses pelatihan model (May, 2019). Augmentasi data sangat berguna untuk mengatasi data pelatihan yang terbatas atau mahal untuk diperoleh seperti dalam aplikasi biomedis. Augmentasi citra membuat model lebih mampu menangani berbagai macam ukuran dan bentuk objek masukan. Teknik augmentasi citra untuk *deep learning* biasanya dilakukan dengan transformasi geometris, *flipping*, memodifikasi warna, *cropping*, rotasi, injeksi derau, dan menghapus bagian citra secara acak.

Solusi yang lain adalah menggunakan *Generative Adversarial Network* (GANs) untuk membuat citra sintetis untuk augmentasi citra.

3.10 Anotasi Citra

Anotasi citra merupakan sebuah tahap primer dalam pembuatan model *computer vision*. Anotasi citra dilakukan untuk mempersiapkan dataset, komponen utama dari *machine learning* dan *deep learning*. Persiapan dataset dilakukan dengan memberikan label (kelas) pada setiap citra. Dataset yang telah dianotasi kemudian digunakan untuk pelatihan model sehingga nantinya model dapat mengidentifikasi kelas tersebut dalam gambar baru yang belum pernah dilihat sebelumnya. Pekerjaan anotasi citra biasanya dilakukan secara manual oleh manusia, terkadang juga dengan bantuan program komputer. Terdapat beberapa alat (aplikasi) anotasi gambar yang gratis dan open source yaitu MakeSense.AI, *Computer Vision Annotation Tool* (CVAT) dan labelImg.

Anotasi citra untuk beberapa tugas dalam *computer vision* (*image classification*, *object detection*, dll.) membutuhkan teknik yang berbeda untuk membuat dataset yang efektif. Pada *image classification* anotasi dilakukan dengan pengelompokan citra berdasarkan kelas labelnya dengan kata lain memberikan satu label kelas untuk satu citra yang menggambarkan keseluruhan isi dari citra tersebut. Pada deteksi objek anotasi citra membutuhkan pembatas area (*bounding box*) dari masing-masing objek yang ada pada citra, sebagai penanda lokasi objek beserta dengan label kelas dari objek tersebut. Pada segmentasi citra anotasi dilakukan dengan memberikan *bounding box* sesuai dengan bentuk objek yang terlihat pada citra beserta label kelasnya, seperti yang ditunjukkan pada Gambar 3.10 berikut.



Gambar 3.10 Contoh pelabelan objek dengan labelImg

Dari beberapa kebutuhan yang berbeda pada anotasi citra maka dapat dikenal beberapa jenis atau bentuk anotasi pada citra sebagai berikut.

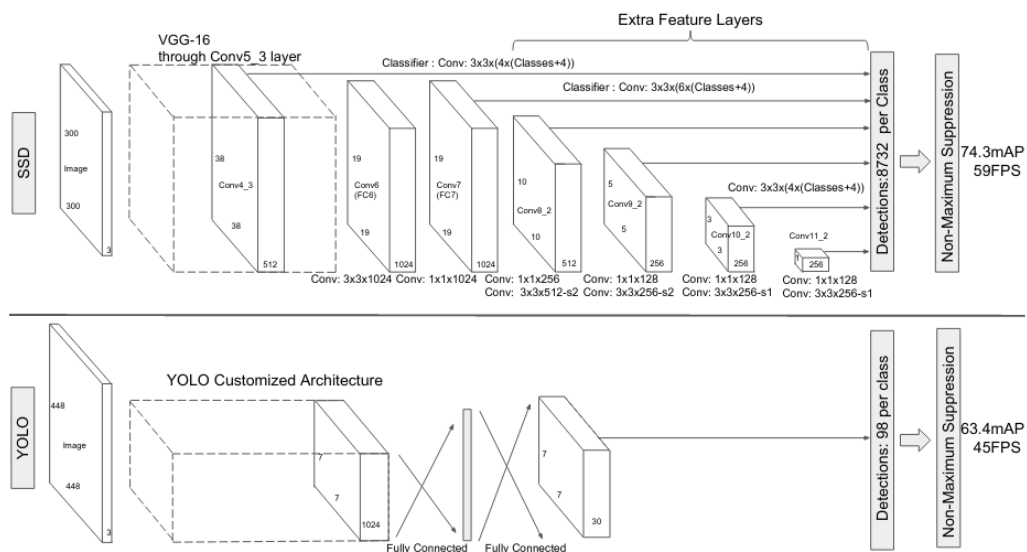
1. *Bounding box*: Kotak persegi yang mendefinisikan lokasi dari objek pada citra, dapat berbentuk dalam dua dimensi (2D) maupun tiga dimensi (3D)
2. *Polygon*: Digunakan untuk menganotasi objek yang berbentuk tidak beraturan, menandai setiap simpul dari objek yang dimaksud dan menganotasi tepinya.
3. *Landmarking*: Digunakan untuk menandai titik-titik fundamental yang menarik dalam citra, sebagai titik kunci. *Landmarking* ini sangat penting dalam tugas *face recognition*.
4. *Line* dan *Splines*: Anotasi dilakukan dengan memberikan garis lurus atau melengkung pada citra. Hal ini penting untuk pengenalan batas untuk anotasi trotoar, marka jalan, dan indikator batas lainnya.

Tahapan anotasi citra pada deteksi objek adalah sebagai berikut:

1. Menyiapkan dataset citra
2. Menentukan label kelas objek yang akan dideteksi
3. Membuat kotak yang mengelilingi objek yang akan dideteksi pada setiap citra
4. Memilih label kelas dari objek pada setiap kotak yang dibuat
5. Ekspor anotasi tersebut ke dalam format yang dibutuhkan (COCO JSON, YOLO, dll.)

3.11 Single Shot Multibox Detector (SSD)

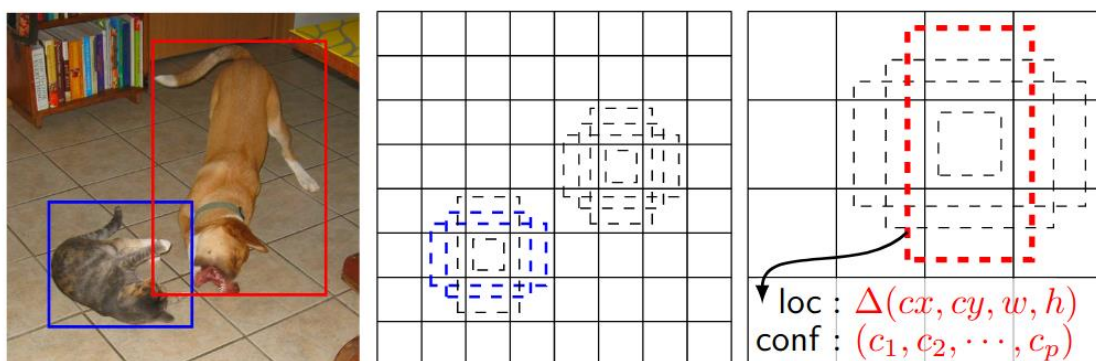
Single Shot Multibox Detector adalah salah satu arsitektur jaringan *neural network* untuk model pendeteksian objek (Liu et al., 2015). Arsitektur jaringan ini dapat dilihat pada Gambar 3.11. Arsitektur ini mencoba untuk meningkatkan performa model pendeteksian objek lebih dari arsitektur R-CNN dan penerusnya agar dapat mencapai pendeteksian secara *realtime*. Performa dari arsitektur SSD mendapatkan skor 74% mAP (*mean average precision*) dengan 59 *frame* per detik (FPS) pada dataset Pascal VOC dan COCO. SSD menggunakan VGG-16 sebagai jaringan dasar untuk ekstraksi fitur.



Gambar 3.11 Arsitektur jaringan SSD

Sumber: (Liu et al., 2015)

SSD model menggunakan lebih banyak *bounding box* pada lokasi sampel dengan skala dan rasio yang berbeda, dapat dilihat pada Gambar 3.12. SSD menggunakan tambahan lapisan konvolusi untuk memprediksi kategori objek dan *offset* lokasi pembatas. Penambahan lapisan ini dapat meningkatkan kecepatan deteksi dan mencapai akurasi tinggi walaupun dengan resolusi *input* yang relatif rendah. SSD diklaim memiliki kecepatan dan akurasi yang lebih tinggi dibandingkan dengan model pendeteksian sebelumnya yaitu YOLO dan Faster R-CNN (Liu et al., 2015).



Gambar 3.12 Penggunaan banyak *bounding box* lokasi sampel

3.12 Python

Python adalah bahasa pemrograman yang banyak digunakan, *interpreted*, berorientasi objek, dan tingkat tinggi dengan semantik dinamis, digunakan untuk pemrograman tujuan umum. Python dibuat oleh Guido Van Rossum dan pertama kali dirilis pada 20 Februari 1991. Penulisan bahasanya dan menggunakan pendekatan berorientasi objek membantu *programmer* menulis kode yang jelas dan logis untuk proyek skala kecil dan besar. Python menggunakan indentasi untuk memberitahu *interpreter* bahwa suatu kode merupakan bagian dari suatu blok program tertentu.

Python menyediakan banyak *library* untuk berbagai kebutuhan pembuatan program yang salah satunya bernama NumPy untuk pengolahan data matriks besar dan multi dimensi. Pada umumnya Python juga digunakan dalam proyek AI dan *machine learning* dengan bantuan *library* seperti TensorFlow, Keras, Pytorch, dan Scikit-learn. Selain menyediakan banyak *library* Python adalah bahasa pemrograman yang mudah dibaca dan dipelajari sehingga lebih banyak digunakan berbagai multi disiplin.

Python dapat berfungsi sebagai bahasa skrip untuk aplikasi web. Umumnya digunakan untuk membuat aplikasi web di sisi *backend*. Terdapat beberapa kerangka kerja web untuk Python seperti Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle dan Zope yang mendukung pengembang dalam desain dan pemeliharaan aplikasi yang kompleks.

3.13 Web API

Di dalam pengembangan aplikasi web terdapat dua istilah yang digunakan yaitu *frontend* dan *backend*. *Frontend* adalah bagian yang dapat dilihat pada tampilan web dan menjadi antarmuka untuk pengguna dan aplikasi sedangkan *backend* adalah sistem dibalik layar yang mengolah data dan *server*. Bagian *frontend* sering disebut juga *client-side* dan bagian *backend* sering disebut juga *server-side*. Keduanya butuh saling berkomunikasi untuk membuat fungsionalitas dari sebuah aplikasi web agar dapat beroperasi, yaitu melalui web API.

Web API merupakan antarmuka pemrograman aplikasi untuk *server* web yang menjembatani interaksi antar aplikasi, servis, dan data (Lauret, 2019). Sebuah web API terdiri dari satu atau lebih *endpoint* yang diekspos untuk menangani pesan permintaan ke aplikasi dan pesan respon dari aplikasi, biasanya diekspresikan dalam JSON atau XML yang diekspos melalui HTTP berbasis *web server*. Web API bisa memiliki akses publik atau privat, akses privat biasanya memerlukan *access token* untuk dapat mengaksesnya.

3.14 Flask

Flask merupakan sebuah *framework* web mikro yang ditulis dengan bahasa Python. Flask dibuat oleh Armin Ronacher, seorang anggota sebuah organisasi internasional penggemar Python yang bernama Poccoo, pertama kali dirilis pada bulan April 2010. Flask diklasifikasikan sebagai *framework* mikro karena tidak membutuhkan *tools* atau *library* tertentu, tidak memiliki lapisan abstraksi database, validasi formulir, atau komponen lain di mana *library* pihak ketiga telah menyediakan fungsi yang umum. Namun, Flask mendukung ekstensi untuk menambahkan fitur aplikasi seolah-olah diimplementasikan di Flask itu sendiri, contohnya adalah ekstensi pemetaan relasional objek, validasi formulir, *upload handler*, dan autentikasi.

Semua aplikasi Flask harus membuat *instance* aplikasi. *Web server* meneruskan semua permintaan yang diterima dari *client* ke objek ini untuk ditangani. *Instance* dari aplikasi perlu mengetahui kode yang mana yang perlu dijalankan untuk setiap URL yang diminta sehingga perlu menyimpan pemetaan URL ke fungsi Python. Asosiasi antara URL dan fungsi yang menanganinya disebut *route*, dilakukan dengan menambahkan *decorator* pada fungsi.

Data dikirim bolak-balik antara *client* dan *server* di bagian badan pada *request* dan *response*. Jenis format yang digunakan untuk mengkodekan data pada bagian badan terletak pada *Content-Type header*. Dua tipe format yang umumnya digunakan pada web API adalah JSON dan XML. JSON lebih cocok digunakan pada web API karena kedekatannya dengan JavaScript, bahasa skrip pada sisi *client* yang digunakan oleh *web browser* (Grinberg, 2014).

3.15 VueJS

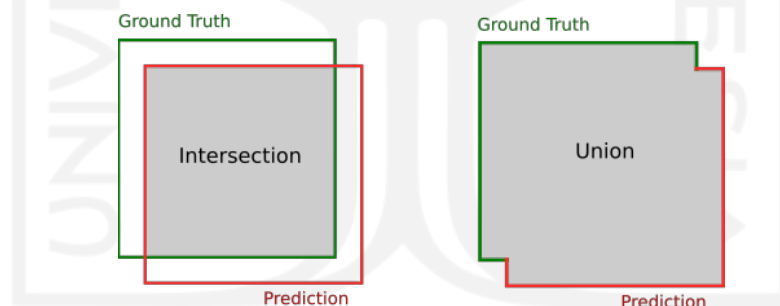
VueJs adalah *framework* JavaScript yang memudahkan pengembang untuk membuat antarmuka web yang kaya dan interaktif, dibuat oleh Evan You saat bekerja di Google dan dirilis pertama kali pada awal 2014. VueJs paling sering digunakan untuk membuat antarmuka web dan aplikasi *one-page*, bisa digunakan untuk aplikasi *desktop* dan *mobile*. VueJs berisi fungsionalitas yang memungkinkan pengembangan aplikasi web yang berfungsi penuh: memanipulasi data yang rumit dan menampilkannya pada tampilan web, menangani *routing* di sisi *client* alih-alih bergantung pada *server*, dan memungkinkan untuk mengakses *server* hanya sekali di awal untuk menampilkan aplikasi web penuh (Macrae, 2018).

VueJs mengkonsumsi dan menampilkan data dari sebuah API menggunakan Axios. Axios adalah sebuah *client* HTTP berbasis *promise* sehingga permintaan ke *server* dapat berjalan asinkron. *Promise* adalah *proxy* untuk nilai yang belum tentu diketahui saat janji

eksekusi dari kode JavaScript dijalankan. Axios membantu penanganan permintaan *client* ke API dan respon dari API (*server*).

3.16 Evaluation Matrix

Evaluation matrix digunakan untuk mengevaluasi model pendeteksian objek, yang terdiri dari *precision*, *recall* dan *mean average precision* (mAP) (Padilla et al., 2020). Dalam deteksi objek dibutuhkan proses kalsifikasi dan lokalisasi. Hasil dari eksperimen dihitung menggunakan *intersection over union* (IoU). Ilustrasi perhitungan area *intersection* dan *union* dapat dilihat pada Gambar 3.13. *Union* merupakan gabungan area prediksi ($A1$) dengan area *ground truth* ($A2$) sedangkan *intersection* merupakan perpotongan area prediksi dengan area *ground truth*. IoU merupakan pembagian antara area *intersection* dan *union* yang ditunjukkan pada persamaan (3.4). Proposal regional dianggap positif jika IoU melebihi ambang batas. Jika skor IoU berada di atas ambang batas dan nilai prediksi sesuai dengan nilai sebenarnya, itu diamati sebagai *true positive* (TP). Jika tidak, itu adalah *false positif* (FP). Jika kelompok objek kebenaran tidak memiliki objek prediksi yang terkait, itu dianggap sebagai *false negatif* (FN).



Gambar 3.13 Ilustrasi perhitungan *intersection* dan *union*

$$IoU = \frac{|A1 \cap A2|}{|A1 \cup A2|} \quad (3.5)$$

Precision mendeskripsikan reabilitas dari sebuah model pendeteksian objek. Dapat dihitung dari persamaan (3.5).

$$Precision = \frac{TP}{TP + FP} \quad (3.6)$$

Recall mendeskripsikan kemampuan dari model pendeteksian objek. Dihitung menggunakan persamaan (3.6).

$$Recall = \frac{TP}{TP + FN} \quad (3.7)$$

Mean average precision (mAP) banyak digunakan untuk mengevaluasi model pendeteksian objek. *Average Precision* (AP) adalah area dibawah kurva *precision* dan *recall*, sehingga nilai mAP dapat diperoleh dari menghitung rata-rata AP dari masing-masing kelas.

3.17 TensorFlow

TensorFlow merupakan software *library* yang dikembangkan oleh Google Brain, dirilis pertama kali pada tahun 2015 dengan lisensi Apache 2.0. Google menggunakan TensorFlow dalam proses riset dan produksi layanan mereka. TensorFlow terkenal dalam pengembangan *machine learning*, *deep learning* dan *artificial intelligence*.

TensorFlow dapat dijalankan lintas platform. Kelebihan Tensorflow dapat berjalan di banyak CPU dan GPU yang mempermudah pengembang dalam pengembangan *deep learning*. TensorFlow tersedia pada perangkat Linux, macOS, Windows dan perangkat mobile termasuk Android dan iOS. Selain itu TensorFlow dapat digunakan pada aplikasi *desktop*, *mobile*, *server* atau web.

3.18 TensorFlow Serving

TensorFlow Serving adalah sistem penyedia yang fleksibel dan berperforma tinggi untuk model *machine learning* yang dirancang untuk lingkungan produksi. Sistem ini memudahkan penerapan algoritma atau percobaan baru dan menyediakan integrasi langsung dengan model TensorFlow.

TensorFlow Serving cocok untuk menyediakan infrastruktur untuk meng-*hosting* model di *server* yang menyediakan API untuk skalabilitas. *Client* dapat menggunakan HTTP untuk meneruskan permintaan ke *server* dengan muatan data. Data kemudian diteruskan ke model yang menjalankan inferensi, mendapatkan hasilnya, dan mengembalikannya ke *client*. Dengan infrastruktur ini memungkinkan pengembang untuk bereksperimen dengan versi model yang berbeda.

TensorFlow Serving hanya tersedia untuk Ubuntu, jadi jika sistem operasi yang digunakan bukan Ubuntu maka perlu sebuah mesin virtual Ubuntu atau sebuah Docker *container* untuk menjalankannya. Cara *setup* di Ubuntu yang perlu dilakukan adalah menambahkan repository dari *package*-nya dan menginstalnya dengan perintah Gambar 3.14. Kemudian menginstal TensorFlow Serving API di sisi *client* yaitu berupa *library* Python dengan perintah Gambar 3.15. Setelah terinstal maka TensorFlow Serving siap digunakan untuk menjalankan model TensorFlow.

```
apt install tensorflow-model-server
```

Gambar 3.14 Perintah instalasi TensorFlow Serving pada Ubuntu

```
pip install tensorflow-serving-api==2.4.1
```

Gambar 3.15 Perintah instalasi TensorFlow Serving API

3.19 Google Colaboratory

Google Colaboratory atau Google Colab merupakan produk yang diciptakan oleh Google Research. Google Colab adalah layanan *jupyter notebook* berbasis *cloud* yang memungkinkan peneliti untuk melatih model *machine learning* dan *deep learning* pada CPU, GPU dan TPU. Layanan ini membantu peneliti yang memiliki keterbatasan perangkat untuk pengembangan AI secara gratis.

Colab menyediakan penggunaan GPU dan TPU gratis. Seperti yang kita tahu bahwa kemampuan komputasi CPU untuk melatih model *deep learning* membutuhkan waktu yang sangat lama. Oleh karena itu, GPU atau TPU sangat dibutuhkan untuk itu, dapat melatih model dalam waktu yang lebih cepat. Misalkan melatih suatu model membutuhkan waktu berjam-jam dengan CPU namun apabila menggunakan GPU atau TPU hanya membutuhkan waktu beberapa menit atau detik. Spesifikasi *server* Google Colab yang dapat digunakan oleh setiap pengguna dapat dilihat pada Gambar 3.16. Colab memberikan waktu maksimal penggunaan GPU atau TPU selama 12 jam, apabila melebihi itu secara otomatis dihentikan.

CPU	GPU	TPU
Intel Xeon Processor with two cores @ 2.30 GHz and 13 GB RAM	Up to Tesla K80 with 12 GB of GDDR5 VRAM, Intel Xeon Processor with two cores @ 2.20 GHz and 13 GB RAM	Cloud TPU with 180 teraflops of computation, Intel Xeon Processor with two cores @ 2.30 GHz and 13 GB RAM

Gambar 3.16 Spesifikasi *server* Google Colab



BAB IV METODOLOGI PENELITIAN

4.1 Populasi dan Sampel Penelitian

Populasi yang digunakan dalam penelitian ini adalah gambar atau foto orang yang memiliki masalah jerawat. Sedangkan sampel yang digunakan pada penelitian ini adalah foto wajah orang yang memiliki masalah jerawat pada wajah dengan jumlah gambar 200.

4.2 Variabel dan Definisi Operasional Penelitian

Data gambar dibagi menjadi 2 kelompok yaitu data *train* dan data *test*, di mana perbandingan porsi 80% untuk data *train* dan 20% untuk data *test* (Goodfellow et al., 2015). Data *train* berisi 160 gambar dan data *test* berisi 40 gambar.

4.3 Jenis dan Sumber Data

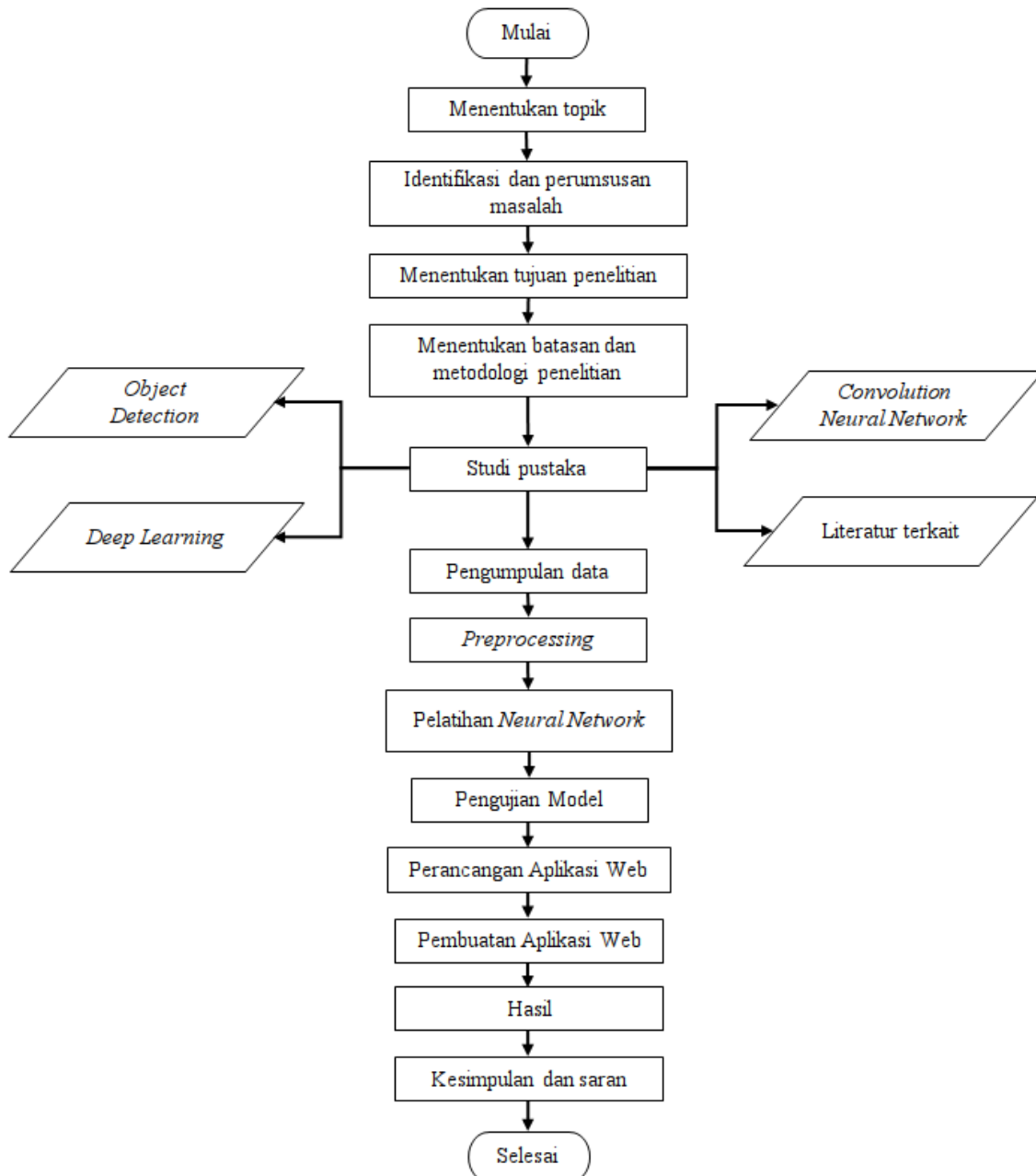
Jenis data yang digunakan dalam penelitian ini adalah data primer dan sekunder. Data primer menggunakan data yang dikumpulkan dari DermNet NZ dan dari Google Images. Data sekunder dari penelitian ini bersumber dari penelitian (Darmawan et al., 2020) yang sudah diberi anotasi oleh dokter kulit.

4.4 Metode Analisis Data

Software yang digunakan pada penelitian ini yaitu Google Chrome, Python 3.8, Google Colab, TensorFlow versi 2.4 dan OpenCV. Metode analisis data yang dilakukan dalam penelitian ini adalah metode *transfer learning* dari model *deep learning* berbasis *Convolutional Neural Network* yang digunakan untuk mendeteksi dan mengklasifikasi objek jerawat pada foto wajah.

4.5 Tahapan Penelitian

Tahapan atau langkah yang digunakan pada penelitian ini digambarkan melalui Gambar 4.1.



Gambar 4.1 Tahapan penelitian

4.6 Evaluasi Hasil

Penelitian ini menghitung performa dari model dengan menggunakan evaluasi matrik yang terdiri dari *precision*, *recall* dan *mean average precision* (mAP). *Precision* mendeskripsikan reabilitas dari model pendeteksian objek. *Recall* mendeskripsikan kemampuan sebuah model pendeteksian objek. Akurasi mendeskripsikan seberapa dekat hasil pendeteksian objek dari model dengan nilai aslinya. *Mean Average Precision* (mAP) digunakan untuk mengevaluasi keseluruhan model pendeteksian objek.

BAB V

ANALISIS DAN PEMBAHASAN

5.1 Pengumpulan Dataset

Peneliti mengumpulkan dan membuat dataset berupa gambar foto wajah orang berjerawat yang dikumpulkan dari DermNet NZ dan Google Image sebagai data primer dan mengambil dari penelitian (Darmawan et al., 2020) sebagai data sekunder. Pencarian data dilakukan dengan menggunakan kata kunci “wajah berjerawat” dan “*facial acne*”. Tidak semua hasil pencarian diambil, hanya gambar yang memenuhi kriteria. Kriteria gambar yang diambil adalah gambar yang hanya mencakup area wajah keseluruhan (*close up*) dan memiliki ukuran $>350\text{px}$. Sementara data dari DermNet NZ berupa foto kasus jerawat yang ada pada area-area bagian wajah. Sampel data gambar yang digunakan dapat dilihat pada Gambar 5.1.



Gambar 5.1 Sampel dataset

5.2 Preprocessing citra

Preprocessing citra dilakukan untuk mempersiapkan dataset sebelum masuk ke proses pelatihan model. Tahapan yang dilakukan adalah segmentasi wajah, *resize* citra, pelabelan citra, pembagian citra dan augmentasi citra. Tahapan tersebut dilakukan secara berurutan.

5.2.1 Segmentasi Wajah

Segmentasi wajah dilakukan dengan tujuan menghilangkan dan meminimalisir *background* yang tidak berguna. Segmentasi wajah dilakukan dengan menggunakan program Python dan OpenCV. Program tersebut melakukan pendeteksian wajah dan ekstraksi landmark wajah. Setelah landmark wajah didapatkan kemudian dilakukan pengambilan area wajah berdasarkan landmark wajah terluar sehingga dapat didapatkan gambar seperti Gambar 5.2. Proses deteksi wajah menggunakan *Haar cascades* OpenCV dengan pertimbangan deteksi dapat dilakukan dengan cepat dan ringan walaupun dengan perangkat dengan sumberdaya terbatas. Sedangkan untuk melakukan pendeteksian landmark wajah menggunakan Dlib dengan model pendeteksian 85 titik landmark pada wajah. Titik-titik tersebut mencakup mulut, alis kanan, alis kiri, mata kanan, mata kiri, hidung, dan rahang pada bidang 2D. Area wajah didapatkan dari titik-titik terluar yang dilakukan penyatuan titik menggunakan *convex hull* sehingga dapat dilakukan *cropping* pada area tersebut.



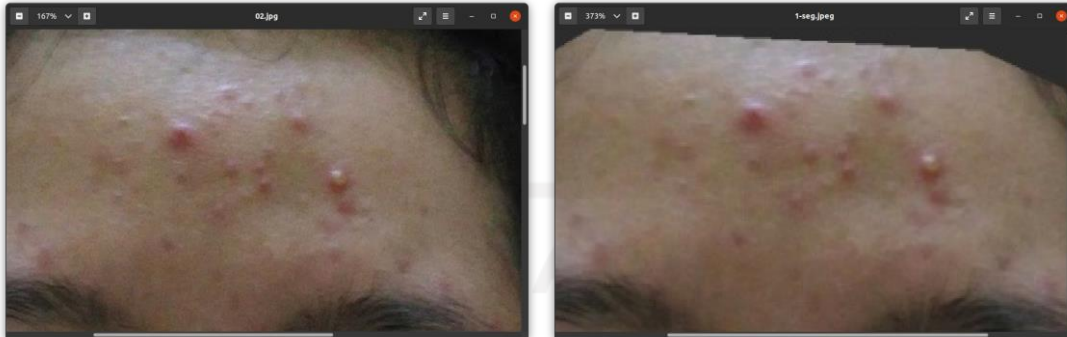
Gambar 5.2 Proses segmentasi wajah dengan landmark wajah

5.2.2 Resize Citra

Citra *input* memiliki ukuran yang bervariasi sehingga perlu dilakukan standarisasi ukuran. Ukuran citra yang terlalu besar dapat membebani proses komputasi karena kompleksitas yang terlalu tinggi sedangkan ukuran citra yang terlalu kecil dapat membuat proses lebih cepat namun dapat mengakibatkan hilangnya informasi dari citra. Oleh karena itu, penentuan ukuran citra mengikuti ukuran objek yang ingin dideteksi.

Gambar dari hasil segmentasi wajah kemudian dilakukan *resize* citra dengan ukuran 500px. Gambar dengan ukuran tersebut bukan citra yang besar dan secara visual manusia objek jerawat masih dapat terlihat dengan jelas. Seperti yang dapat dilihat pada Gambar 5.3,

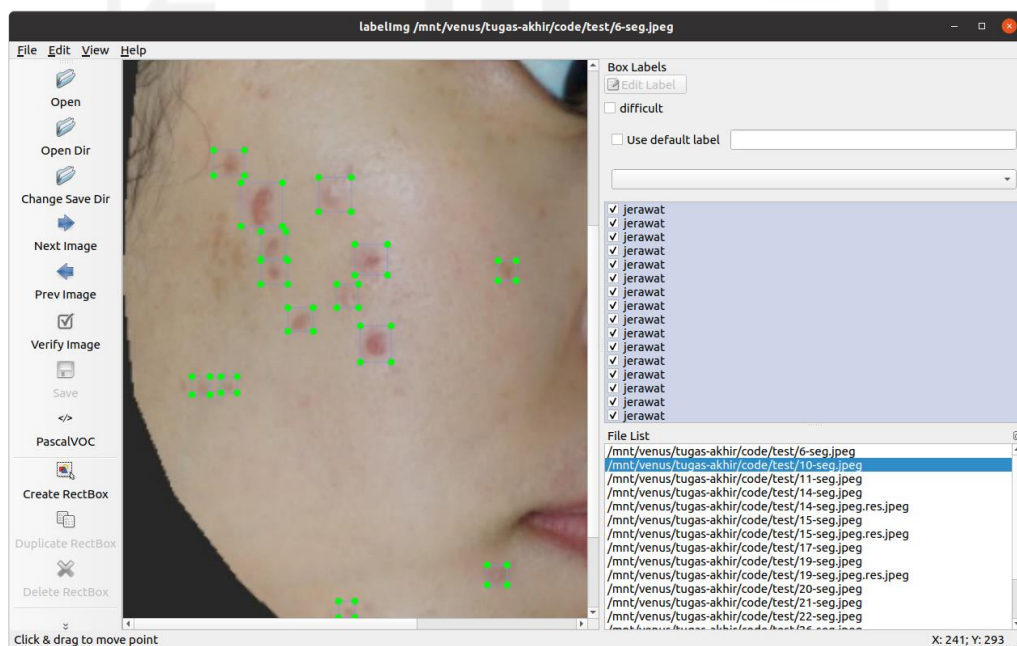
foto sebelah kiri adalah foto citra asli dengan ukuran 1000px dan foto sebelah kanan foto dengan ukuran 500px. Proses *resize* citra dilakukan secara otomatis dengan menggunakan program Python dan OpenCV.



Gambar 5.3 Contoh resolusi gambar jerawat setelah *resize*

5.2.3 Pelabelan Citra

Pelabelan citra atau anotasi gambar dilakukan untuk mengidentifikasi objek yang ingin dideteksi pada citra. Pelabelan dilakukan dengan menggunakan *software* anotasi gambar yang bernama labelImg. Pelabelan dilakukan dengan memberi tanda area kotak pada objek beserta dengan nama objeknya (*class*). Hasil dari pelabelan ini berupa berkas XML dengan format PASCAL VOC untuk setiap gambar. Contoh anotasi jerawat menggunakan labelImg dapat dilihat pada Gambar 5.4.



Gambar 5.4 Contoh pelabelan objek jerawat dengan labelImg

5.2.4 Pembagian Citra

Pembagian citra dilakukan untuk memisahkan data gambar untuk proses pelatihan dan pengujian. Perbandingan porsi untuk masing-masing adalah 80% untuk pelatihan dan 20% untuk pengujian (Goodfellow et al., 2015). Perbandingan tersebut merupakan perbandingan yang umum digunakan, di mana 80% dari data digunakan untuk melatih *neural network* dan sisanya untuk pengujian. Gambar hasil pelabelan kemudian dipisah kedalam folder yang berbeda, *train* dan *test*. Masing-masing folder berisi berkas gambar beserta dengan berkas XML-nya.

5.2.5 Augmentasi Citra

Augmentasi citra bertujuan untuk menambah variasi dari data *train*. Sebagai salah satu cara untuk mengatasi jumlah data *train* yang sedikit. Augmentasi citra dilakukan dengan membuat citra baru dari citra asli. Citra asli dimodifikasi dengan cara mengubah nilai *brightness*, penambahan *noise* dan rotasi. Harapannya apabila augmentasi citra dilakukan, model pendeteksian yang dihasilkan dari pelatihan dapat menghasilkan prediksi yang lebih akurat.

Augmentasi citra dilakukan secara otomatis menggunakan *bounding box augmentation*. Program ditulis menggunakan Python dan *library* bernama *imgaug*. Augmentasi ini dilakukan setelah pelabelan data selesai dan hanya untuk data *train* saja. Program dapat menyesuaikan lokasi objek yang telah diberi label sehingga tidak perlu melakukan pelabelan ulang pada citra baru yang terbentuk. Program ini melakukan augmentasi berdasarkan jenis-jenis augmentasi yang didefinisikan pada program sehingga program hanya perlu mengambil citra asli, memproses, dan menyimpan citra baru ke folder yang ditentukan. Pada Tabel 5.1 merupakan jenis augmentasi yang digunakan.

Tabel 5.1 Tabel jenis augmentasi citra

Jenis	Keterangan
<i>Blur</i> (Average Blur, Motion Blur)	Mengaburkan gambar dengan nilai rata-rata atau memburamkan gambar dengan cara memalsukan gerakan kamera atau objek.
<i>Arithmetic</i> (Add, Additive Gaussian Noise)	Menambahkan nilai ke semua piksel dalam gambar dengan retang -40 s/d 40 atau menambahkan <i>noise</i> sampel dari distribusi

	Gaussian.
<i>Contrast</i> (Gamma Contrast, Linear Contrast)	Menyesuaikan kontras gambar dengan menskalakan nilai piksel ke $255*((v/255)**\gamma)$ dengan nilai rentang γ 0.5 s/d 1.0 atau menyesuaikan kontras dengan menskalakan setiap piksel ke $127 + \alpha*(v-127)$ dengan rentang 0.4 s/d 1.5
<i>Color</i> (Brightness, Saturation)	Menambahkan nilai kecerahan gambar dengan rentang -40 s/d 40 atau mengalikan nilai saturasi gambar dengan nilai acak dengan rentang 0.5 s/d 1.5.
<i>Geometric</i> (Flip Horizontal, Rotation)	Membalik gambar secara horizontal dan memutar citra antara -30 s/d 30.

Dari kelima proses augmentasi yang didefinisikan, dapat membuat jumlah gambar asli menjadi lima kali lipat. Proses augmentasi berjalan dengan cara mengambil random pilihan jenis operasi dan mengambil nilai random dari rentang nilai yang telah ditentukan.

5.3 Konfigurasi Google Colab

Proses pelatihan model *deep learning* dilakukan di Google Colab. Google Colab memberikan fasilitas pemrosesan GPU secara gratis selama maksimal 12 jam. Fasilitas ini sangat mendukung untuk proses pelatihan model *deep learning* yang membutuhkan *hardware* dengan kemampuan komputasi yang cukup besar. Proses pelatihan menggunakan komputer pribadi dengan spesifikasi standar tidak mampu untuk menjalankan proses pelatihan tersebut.

5.3.1 Mengunggah Dataset ke Google Drive

Google Colab harus bisa mengakses dataset dan program yang diperlukan. Berkas-berkas tersebut kemudian diunggah ke Google Drive yang dapat terhubung secara langsung dengan Google Colab. Berkas diunggah pada folder yang sudah ditentukan untuk pelatihan model (*workspace*), hal ini mempermudah dalam proses *mounting* alamat Google Drive ke Google Colab sehingga nantinya ketika memerlukan suatu berkas tidak perlu berpindah alamat folder yang berbeda.

5.3.2 Menghubungkan Google Colab dan Google Drive

Walaupun Google Colab dan Google Drive merupakan layanan milik Google, untuk menghubungkan keduanya pengguna perlu memberikan hak akses. Hak akses tersebut ditanyakan ketika proses *mounting* alamat folder ke Google Colabs. Hak akses yang diminta berupa kode token otorisasi. Google akan menanyakan izin untuk mengakses alamat Google Drive, jika pengguna memberikan izin maka Google akan memberikan kode hak aksesnya sehingga Google Colabs dapat mengakses berkas-berkas didalamnya.

5.3.3 Instalasi *Library* Python

Lembar kerja Google Colab berupa *jupyter notebook* dengan bahasa pemrograman Python. Sebelum melanjutkan ke proses pelatihan model, perlu dipastikan terlebih dahulu kebutuhan *library* yang digunakan sudah terinstal agar tidak terjadi *error* program. Pada Tabel 5.2 berikut ini adalah daftar *library* Python yang digunakan.

Tabel 5.2 Daftar *library* Python yang digunakan

Nama <i>library</i>	Keterangan
Tensorflow versi 2.4	<i>Library</i> untuk pelatihan model <i>deep learning</i>
Numpy	<i>Library</i> pengolahan array dan matriks multidimensi
OpenCV	<i>Library</i> untuk pengolahan gambar atau <i>computer vision</i>
TensorFlow Object Detection API	<i>Library</i> untuk pelatihan model <i>deep learning</i> untuk pendeteksian objek.

5.4 Membuat TensorFlow Record

Sebelum melakukan pelatihan model pada TensorFlow, data *train* dan data *test* yang dimiliki perlu diubah ke dalam format TFRecord. Hasil dari pelabelan citra berupa berkas XML kemudian diubah ke dalam format CSV dan TFRecord. Selain itu, diperlukan juga berkas *label map* yang mendefinisikan objek-objek di dalamnya.

5.4.1 Membuat *Label Map*

Label map dibutuhkan untuk mendefinisikan setiap nama kelas atau label ke dalam format *integer*. Semua label yang ada pada proses pelabelan harus didefinisikan pada berkas ini. *Label map* disimpan dalam format “.pbtxt” yang digunakan untuk konfigurasi pelatihan model. Berkas ini digunakan baik diproses pelatihan maupun pendeteksian.

Struktur dari berkas ini berupa daftar *item* yang di dalamnya berisi *id* dan *name*. Atribut *id* merupakan nilai *integer* yang digunakan untuk memetakan *output* pendeteksian model dan atribut *name* merupakan nama objeknya. Nama dari objek harus sama persis dengan label yang disematkan pada gambar yang telah dianotasi. Semua kelas objek pada dataset harus didefinisikan pada berkas ini. Gambar 5.5 merupakan isi dari *label map* yang digunakan.

```
item {
  id: 1,
  name: 'jerawat'
}
```

Gambar 5.5 Isi dari *label map*

5.4.2 Mengubah Berkas XML ke CSV

Data gambar dan label XML-nya kemudian diubah kedalam berkas berformat CSV. Berkas CSV ini berfungsi untuk mencatat semua gambar beserta dengan koordinat setiap objek di dalamnya. Gambar 5.6 dan Gambar 5.7 merupakan perintah pada Google Colab untuk menjalankan program yang mengubah semua berkas XML ke dalam satu berkas CSV, masing-masing untuk data *test* dan *train*.

```
# Script untuk data train
!python xml_to_csv.py -i images/train -o annotations/train_label.csv
```

Gambar 5.6 Perintah untuk mengubah XML ke CSV data *train*

```
# Script untuk data train
!python xml_to_csv.py -i images/test -o
annotations/test_label.csv
```

Gambar 5.7 Perintah untuk mengubah XML ke CSV data *test*

Program “xml_to_csv.py” mencatat semua label yang ada pada semua berkas XML ke dalam satu berkas CSV. Parameter yang perlu dipersiapkan untuk menjalankan program ini adalah alamat folder dari dataset (*input*) dan alamat penyimpanan berkas CSV (*output*). Dapat dilihat dari perintah Gambar 5.6, alamat folder untuk data *train* adalah “images/train” dan alamat penyimpanan berkas CSV adalah “annotations/train_label.csv”. Begitu juga untuk

data *test*, alamat folder untuk data *test* adalah “images/test” dan alamat penyimpanan berkas CSV adalah “annotations/test_label.csv”. Isi dari berkas CSV tersebut adalah daftar semua objek yang ada pada dataset. Tabel 5.3 Format isi berkas CSV berupa baris dan kolom.

Tabel 5.3 Format isi berkas CSV

Nama Kolom	Keterangan
Filename	alamat berkas gambar
Width	ukuran lebar gambar (piksel)
Height	ukuran tinggi gambar (piksel)
Class	Nama atau kelas objek
Xmin	Koordinat x sudut kiri atas dari <i>bounding box</i> objek
Ymin	Koordinat y sudut kiri atas dari <i>bounding box</i> objek
Xmax	Koordinat x sudut kanan bawah dari <i>bounding box</i> objek
Ymax	Koordinat y sudut kanan bawah dari <i>bounding box</i> objek

5.4.3 TFRecord Data Train

Data *train* yang sudah dipersiapkan sebelumnya kemudian diubah ke dalam format TFRecord. Gambar 5.8 merupakan perintah yang digunakan di dalam Google Colab untuk menjalankan program yang disediakan *TensorFlow Object Detection API* yang berfungsi mengubah format dataset *train* ke dalam TFRecord. Parameter yang perlu dipersiapkan berupa alamat berkas CSV, alamat folder data, dan alamat penyimpanan berkas TFRecord dari data *train*. Hasil dari program tersebut adalah suatu berkas bernama “train.record”.

```
# Script untuk data train
!python generate_tfrecord.py --csv_input=images/train_labels.csv
--image_dir=images/train --output_path=train.record
```

Gambar 5.8 Perintah untuk membuat TFRecord data *train*

5.4.4 TFRecord Data Test

Data *test* yang sudah dipersiapkan sebelumnya kemudian diubah ke dalam format TFRecord. Gambar 5.9 merupakan perintah yang digunakan di dalam Google Colab untuk menjalankan program yang disediakan *TensorFlow Object Detection API* yang berfungsi mengubah format dataset *test* ke dalam TFRecord. Parameter yang perlu dipersiapkan berupa

alamat berkas CSV, alamat folder data, dan alamat penyimpanan berkas TFRecord dari data *test*. Hasil dari program tersebut adalah suatu berkas bernama “test.record”.

```
# Script untuk data tes
!python generate_tfrecord.py --csv_input=images/test_labels.csv
--image_dir=images/test --output_path=test.record
```

Gambar 5.9 Perintah untuk membuat TFRecord data *test*

5.5 Pelatihan Model

Pelatihan model merupakan proses utama untuk mendapatkan model pendeteksian objek. Pelatihan model dilakukan menggunakan Google Colab yang menyediakan fitur GPU secara gratis untuk pelatihan model selama maksimal 12 jam. Tahap pelatihan model meliputi konfigurasi *pipeline*, proses pelatihan model dan ekspor model.

5.5.1 Konfigurasi Pipeline Pelatihan Deteksi Objek

Konfigurasi *pipeline* pelatihan berada pada berkas “pipeline.config”. Secara garis besar konfigurasi *pipeline* terbagi menjadi lima bagian yaitu model, *training_config*, *eval_config*, *train_input_config*, dan *eval_input_config*. Model deteksi objek umumnya terdapat beberapa konfigurasi untuk proses pelatihan, Tabel 5.4 menunjukkan konfigurasi yang dilakukan.

Tabel 5.4 Konfigurasi *pipeline* pelatihan model model

Konfigurasi	Isi	keterangan
num_classes	1	jumlah kelas objek yang akan dideteksi oleh model
batch_size	8	Jumlah sampel data yang harus dieksekusi untuk sekali <i>forward prop</i>
fine_tune_checkpoint	models/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8/checkpoint/ckpt-0	letak berkas <i>check point</i> dari model yang sudah dilatih sebelumnya
fine_tune_checkpoint_type	detection	jenis model yang digunakan
label_map_path	annotations/label_map.pbtxt	alamat <i>label map</i>
input_path (<i>train</i>)	annotations/train.record	alamat TFRecord data <i>train</i>

input_path (test)	annotations/test.record	alamat TFRecord data <i>test</i>
-------------------	-------------------------	----------------------------------

5.5.2 Pelatihan Model Deteksi Objek

Pelatihan model adalah tahapan yang utama, model dilatih untuk mempelajari pola dari gambar agar dapat mengenali objek di dalamnya. Tujuannya agar model dapat mendeteksi objek dengan akurasi yang tinggi, di mana program TensorFlow melakukan pelatihan model terhadap dataset jerawat. Pada proses ini model dilatih untuk mempelajari ciri-ciri dari objek jerawat dengan melakukan penyesuaian bobot dan bias pada setiap neuron di dalam lapisan *neural network* agar dapat mendeteksi jerawat pada gambar melalui proses *forward* dan *backward propagation*. Gambar 5.10 merupakan perintah yang digunakan untuk menjalankan proses *training* model di Google Colab.

```
!python model_main_tf2.py --
model_dir=models/my_ssd_resnet50_v1_fpn --
pipeline_config_path=models/my_ssd_resnet50_v1_fpn/pipeline.conf
ig --checkpoint_dir=models/my_ssd_resnet50_v1_fpn --
```

Gambar 5.10 Perintah untuk menjalankan pelatihan model

Proses pelatihan berhenti setelah menyelesaikan jumlah *steps* yang didefinisikan di dalam berkas konfigurasi. Selama proses berjalan nilai *total loss* dari setiap *step* dapat dipantau dari *log* yang ditampilkan di Google Colab. Nilai *total loss* ini menunjukkan perkembangan kemampuan model untuk mendeteksi objek selama proses *training*. Normalnya nilai *total loss* terus mengecil dengan bertambahnya *step* yang dilewati. Hasil dari tiga percobaan pelatihan dengan jumlah *step* yang berbeda dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil percobaan pelatihan dengan jumlah *step* yang berbeda

Jumlah <i>Step</i>	Waktu Pelatihan	<i>Total Loss</i>	<i>Average Precision</i>	<i>Recall</i>
3500	~ 2 jam	15,3%	30%	20%
12000	~ 8 jam	13,3%	42,3%	30,6%
25000	~ 16 jam	2,2%	42,1%	32%

Dari percobaan yang dilakukan terhadap jumlah *step* yang berbeda berpengaruh terhadap nilai *total loss*, *average precision*, dan *recall*. Semakin banyak jumlah *steps* yang digunakan membuat nilai *total loss* semakin menurun sedangkan nilai *average precision* dan *recall* semakin besar. Ini menunjukkan bahwa semakin banyak model dilatih membuat model pendeteksian semakin bagus untuk mempelajari pola objek dari dataset pelatihan. Namun, waktu yang diperlukan juga semakin lama, di mana setiap *step* memerlukan waktu sekitar 4 menit.

5.5.3 Ekspor Model

Selama proses pelatihan berjalan, TensorFlow membuat berkas *checkpoint* secara otomatis berupa *graph tensor*. *Checkpoint* merupakan informasi dari proses pelatihan yang sedang atau telah berjalan. Setelah proses pelatihan selesai maka *checkpoint* ini diekspor menjadi model yang siap digunakan untuk melakukan pendeteksian. Perintah untuk mengekspor model ditunjukkan pada Gambar 5.11.

```
!python exporter_main_v2.py --  
trained_checkpoint_dir=models/my_ssd_resnet50_v1_fpn --  
pipeline_config_path=models/my_ssd_resnet50_v1_fpn/pipeline.conf  
ig --output_directory exported-models
```

Gambar 5.11 Perintah untuk ekspor model

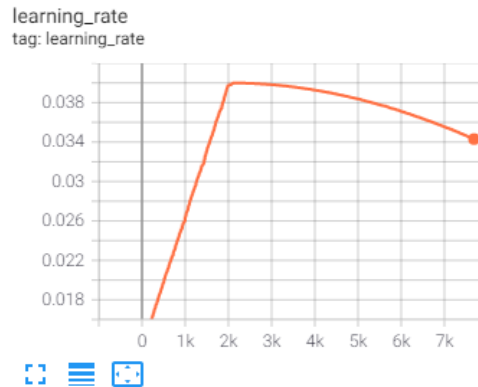
5.6 Model Hasil Pelatihan

Model pendeteksian didapatkan setelah melakukan proses pelatihan model. Model *deep learning* ini yang nantinya digunakan untuk membuat aplikasi pendeteksian jerawat. Sebelum menggunakan model tersebut perlu dilakukan pengujian untuk memastikan model sudah dapat mendeteksi jerawat dari beberapa gambar uji. Jika masih banyak objek yang tidak terdeteksi maka perlu mengevaluasi ulang dataset pelatihan dan melakukan pelatihan ulang, sampai mendapatkan model pendeteksian yang diinginkan. Data hasil pelatihan berupa grafik *learning rate*, grafik *total loss* dan model pendeteksian objek.

5.6.1 Learning Rate

Learning rate merupakan salah satu parameter pelatihan untuk menghitung nilai koreksi bobot pada waktu proses pelatihan berjalan. Nilai *learning rate* ini berada pada *range* nol (0)

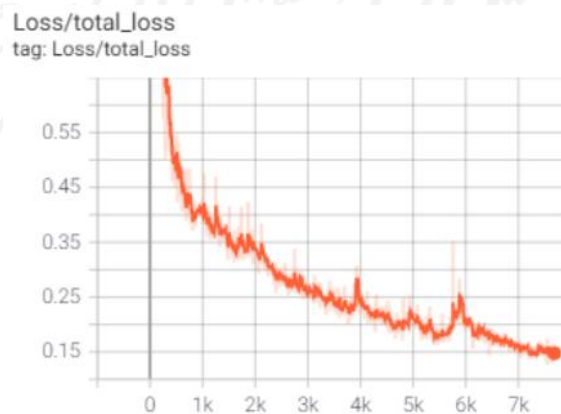
sampai (1). Semakin besar nilai *learning rate*, maka proses pelatihan akan berjalan semakin cepat. Semakin besar *learning rate*, maka ketelitian jaringan akan semakin berkurang, tetapi berlaku sebaliknya, apabila *learning rate*-nya semakin kecil, maka ketelitian jaringan akan semakin besar atau bertambah dengan konsekuensi proses pelatihan akan memakan waktu yang semakin lama. Grafik dari *learning rate* dapat dilihat pada Gambar 5.12.



Gambar 5.12 Grafik *learning rate* pelatihan model

5.6.2 Total Loss

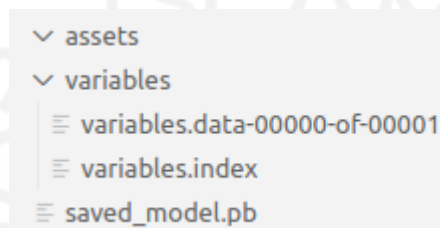
Total loss merupakan gabungan nilai dari *localization loss* dan *classification loss*, jarak antara nilai sebenarnya dengan hasil prediksi model. *Localization loss* merupakan nilai *error* yang dihasilkan dari prediksi lokasi objek dengan lokasi sebenarnya, nilai dari perhitungan IoU. Sedangkan *classification loss* merupakan nilai *error* yang dihasilkan dari prediksi kelas objek. Grafik total loss dari model dapat dilihat pada Gambar 5.13.



Gambar 5.13 Grafik *total loss* pelatihan model

5.6.3 Model

Hasil akhir dari proses pelatihan adalah model yang sudah dapat digunakan untuk melakukan pendeteksian objek. Struktur dari model yang dihasilkan dapat dilihat pada Gambar 5.14. Folder *asset* menyimpan file yang digunakan TensorFlow *graph*, folder *variables* berisi *checkpoint* dari model, dan file *save_model.pb* menyimpan program atau model TensorFlow. Berkas ini merupakan berkas model yang dihasilkan dari proses pelatihan menggunakan *TensorFlow Object Detection API* dengan Google Colab.



Gambar 5.14 Struktur model *object detection*

5.6.4 Pengujian Model

Pengujian model pada tahap ini dilakukan untuk memastikan model hasil *training* apakah sudah dapat mendeteksi jerawat atau belum. Dari visualisasi yang ditampilkan pada Gambar 5.15 menunjukkan bahwa model sudah dapat mendeteksi lesi jerawat dengan baik. Objek jerawat yang terdeteksi ditunjukkan oleh kotak-kotak berwarna hijau yang menandai lokasi dari jerawat. Model deteksi objek selain mengembalikan koordinat lokasi-lokasi piksel juga mengembalikan nilai probabilitasnya, ditampilkan pada Gambar 5.15 berupa nilai persentase. Nilai tersebut nantinya digunakan untuk menentukan *threshold* apakah objek jerawat akan ditampilkan atau tidak. Model hasil *training* sudah dapat digunakan untuk pembuatan aplikasi pendeteksian jerawat.

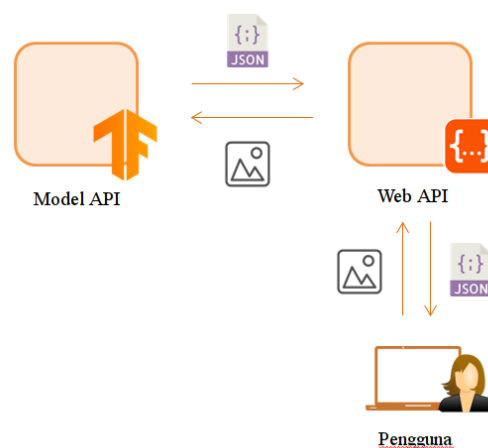


Gambar 5.15 Contoh hasil pendeteksian jerawat

5.7 Desain Aplikasi

Desain dari aplikasi meliputi rancangan alur data dan rancangan antarmuka pengguna. Aplikasi didesain untuk menerima permintaan pendeteksian jerawat melalui antarmuka web. Pengguna memungkinkan untuk mengunggah foto dan menerima lokasi-lokasi dan jumlah jerawat yang berhasil dideteksi aplikasi.

5.7.1 Alur data



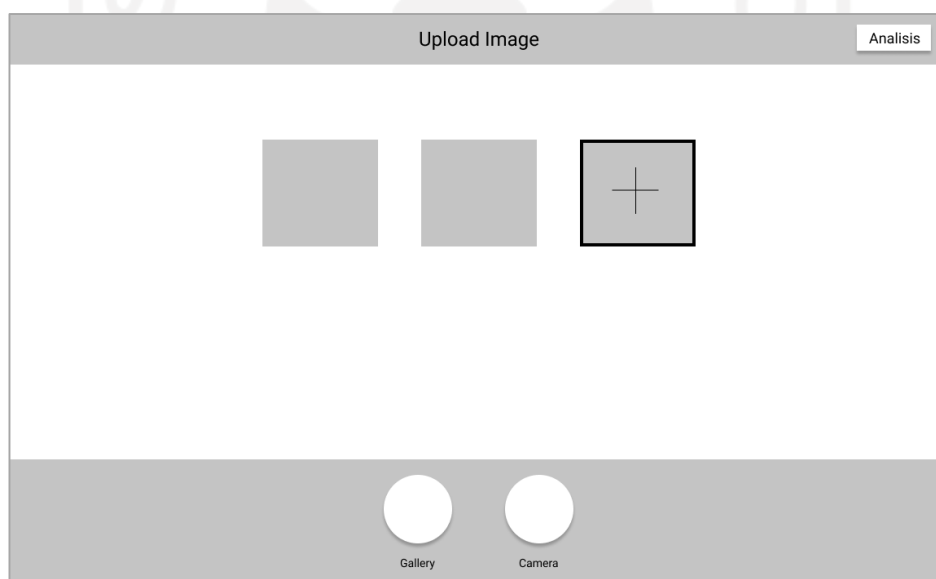
Gambar 5.16 Alur data aplikasi

Pertukaran data pada aplikasi menggunakan web API. Proses alur data dapat dilihat pada Gambar 5.16. Data yang dikirimkan dari antarmuka pengguna berupa gambar atau foto melalui *form upload*. Kemudian Gambar tersebut diterima oleh web API yang menjembatani proses pendeteksian objek. Setelah permintaan diterima, gambar dikirimkan ke *endpoint* model pendeteksian objek (model API). Hasil dari pendeteksian kemudian dikembalikan ke

web API untuk diolah. Data yang diterima berupa daftar koordinat $xmin$, $ymin$, $xmax$, $ymax$ dan $score$. Data tersebut ditata ulang ke dalam format JSON yang lebih simpel dan mudah dibaca. Setelah itu dikirimkan antarmuka aplikasi dalam format JSON untuk ditampilkan pada gambar.

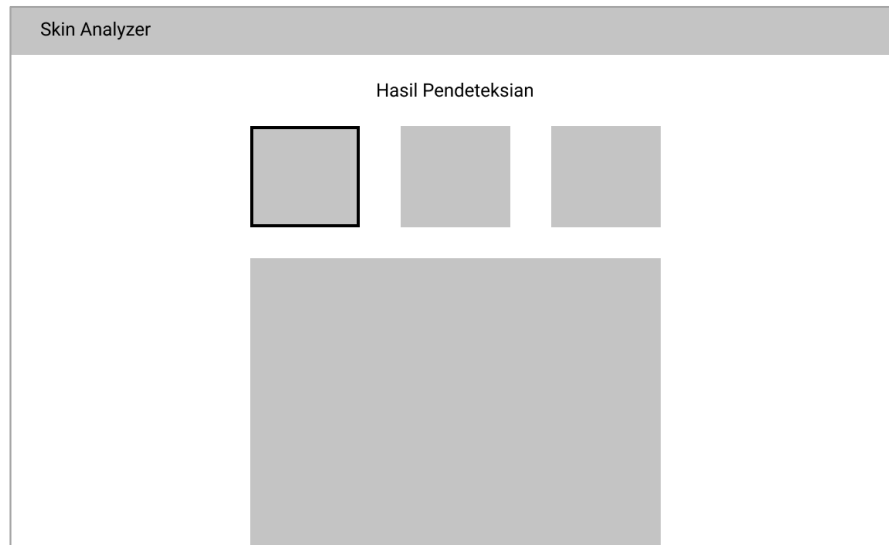
5.7.2 Desain Antarmuka Aplikasi

Desain antarmuka aplikasi diperuntukan untuk aplikasi web. Antarmuka aplikasi digunakan oleh pengguna untuk melakukan pendeteksian jerawat dengan cara mengunggah foto. Setelah proses pendeteksian selesai, aplikasi akan menampilkan lokasi-lokasi jerawat yang terdeteksi pada foto dan jumlahnya. Aplikasi didesain untuk dapat melakukan pendeteksian lebih dari satu foto sekaligus. Gambar 5.17 merupakan rancangan tampilan *form upload*. Sumber foto dapat bersumber dari kamera langsung atau *upload* berkas dari komputer.



Gambar 5.17 Rancangan antarmuka *upload* gambar

Setelah data hasil pendeteksian diterima dari web API, data tersebut kemudian ditampilkan ke halaman hasil. Jerawat yang terdeteksi pada gambar diberi tanda lingkaran yang mengelilingi objeknya. Apabila gambar lebih dari satu maka gambar yang lain ditampilkan pada kolom bagian atas dalam ukuran yang lebih kecil, seperti desain antarmuka Gambar 5.18.



Gambar 5.18 Rancangan antarmuka hasil pendeteksian objek

5.8 Pengembangan Aplikasi

Pengembangan aplikasi meliputi pengembangan Web API (*backend*) dan antarmuka aplikasi (*frontend*). Web API bertugas untuk menerima permintaan pendeteksian dari antarmuka aplikasi dan mengirimkan hasil pendeteksian dari API model pendeteksian objek. Web API dikembangkan dengan menggunakan Python dan Flask (*web framework*). Antarmuka aplikasi bertugas untuk mempermudah pengguna melakukan pengambilan foto atau gambar dan menampilkan hasil pendeteksian jerawat pada foto-foto yang diunggah. Antarmuka aplikasi dikembangkan menggunakan VueJs.

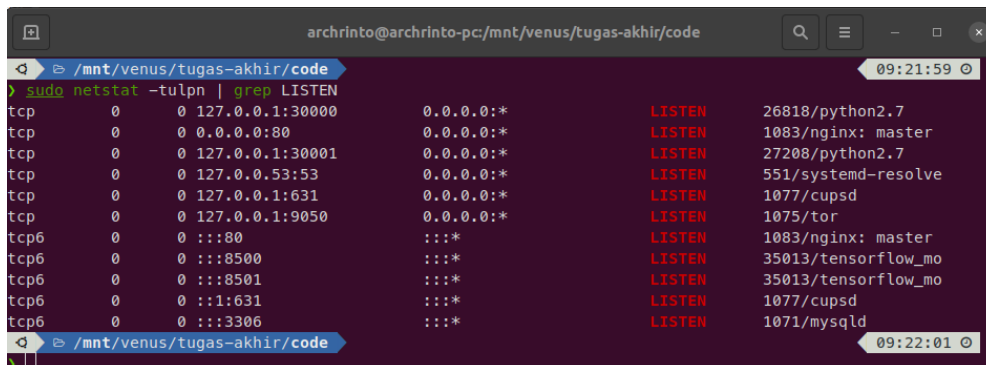
5.8.1 Menjalankan model pendeteksian

Model pendeteksian yang sudah didapatkan dari proses *training* kemudian dijalankan menggunakan TensorFlow Serving. Tujuannya adalah untuk memisahkan aplikasi utama dengan model pendeteksian agar dapat berjalan secara paralel. Setelah dijalankan maka model dapat diakses melalui protocol HTTP sebagai API model pendeteksian jerawat. Perintah untuk menjalankan model dengan TensorFlow Serving dapat dilihat pada Gambar 5.19.

```
tensorflow_model_server --port=8500 --rest_api_port=8501 --
model_name=acne_model --model_base_path=/acne_model
```

Gambar 5.19 Perintah untuk menjalankan model dengan TensorFlow Serving

Perintah pada Gambar 5.19 menunjukkan model pendeteksian dijalankan pada *port* 8501. Untuk memastikan bahwa model sudah berjalan, maka perlu pengecekan pada *port* yang terbuka pada sistem operasi. Pada Ubuntu atau Linux pengecekan menggunakan *netstat*. Dapat dilihat pada Gambar 5.20 terdapat *port* 8501 yang sudah berjalan.



```

archrinto@archrinto-pc:/mnt/venus/tugas-akhir/code
> sudo netstat -tulpn | grep LISTEN
tcp        0      0 127.0.0.1:30000      0.0.0.0:*           LISTEN      26818/python2.7
tcp        0      0 0.0.0.0:80          0.0.0.0:*           LISTEN      1083/nginx: master
tcp        0      0 127.0.0.1:30001     0.0.0.0:*           LISTEN      27208/python2.7
tcp        0      0 127.0.0.53:53       0.0.0.0:*           LISTEN      551/systemd-resolve
tcp        0      0 127.0.0.1:631      0.0.0.0:*           LISTEN      1077/cupsd
tcp        0      0 127.0.0.1:9050     0.0.0.0:*           LISTEN      1075/tor
tcp6       0      0 :::80              :::*                 LISTEN      1083/nginx: master
tcp6       0      0 :::8500            :::*                 LISTEN      35013/tensorflow_mo
tcp6       0      0 :::8501            :::*                 LISTEN      35013/tensorflow_mo
tcp6       0      0 :::1:631           :::*                 LISTEN      1077/cupsd
tcp6       0      0 :::3306            :::*                 LISTEN      1071/mysqld

```

Gambar 5.20 Hasil pengecekan *port* yang terbuka dengan *netstat*

5.8.2 Pembuatan Web API

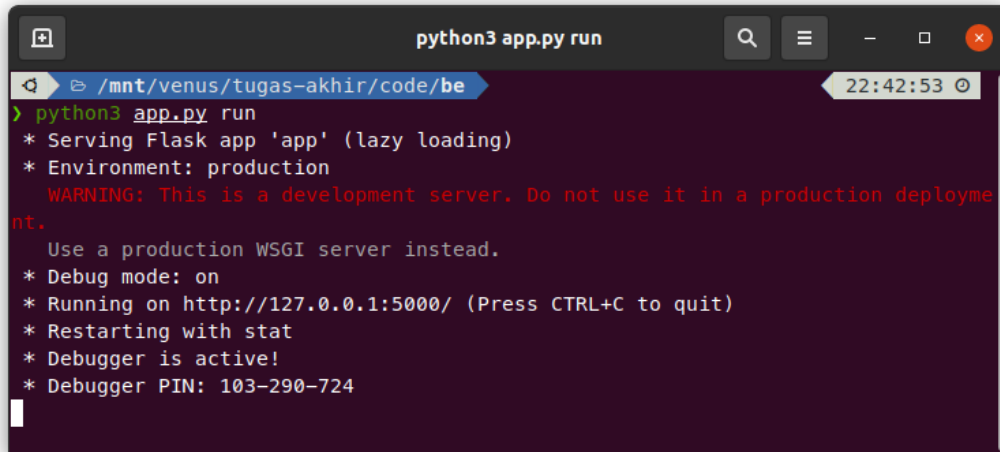
Pengembangan Web API menggunakan Python dan Flask. Web API memiliki *endpoint* untuk menerima permintaan pendeteksian, berupa berkas gambar RGB dari *form upload*. Berkas gambar yang diterima berupa *blob* sehingga perlu diubah ke format *tensor* dengan menggunakan OpenCV dan NumPy agar dapat diolah. Kemudian gambar tersebut masuk ke proses pendeteksian wajah. Apabila wajah terdeteksi, proses selanjutnya adalah mengekstrak titik landmark sebagai acuan untuk melakukan segmentasi wajah. Hasil segmentasi tersebut kemudian dikirimkan ke *endpoint* model pendeteksian jerawat. Pada kasus tertentu, apabila wajah tidak terdeteksi karena pengambilan gambar hanya berupa potongan wajah maka gambar langsung diteruskan ke model pendeteksian.

Hasil dari pendeteksian dari model kemudian ditata ulang menjadi format JSON yang lebih sederhana. Hasil ini yang kemudian dikirimkan ke antarmuka pengguna untuk ditampilkan pada gambar hasil deteksi.

Web API agar dapat diakses oleh antarmuka pengguna harus dijalankan terlebih dahulu. Web API dijalankan menggunakan perintah Gambar 5.21. Setelah web API berhasil dijalankan maka akan muncul tampilan seperti Gambar 5.22 yang menunjukkan bahwa Web API berjalan pada alamat <http://127.0.0.1:5000/>.

```
python3 app.py run
```

Gambar 5.21 Perintah untuk menjalankan web API

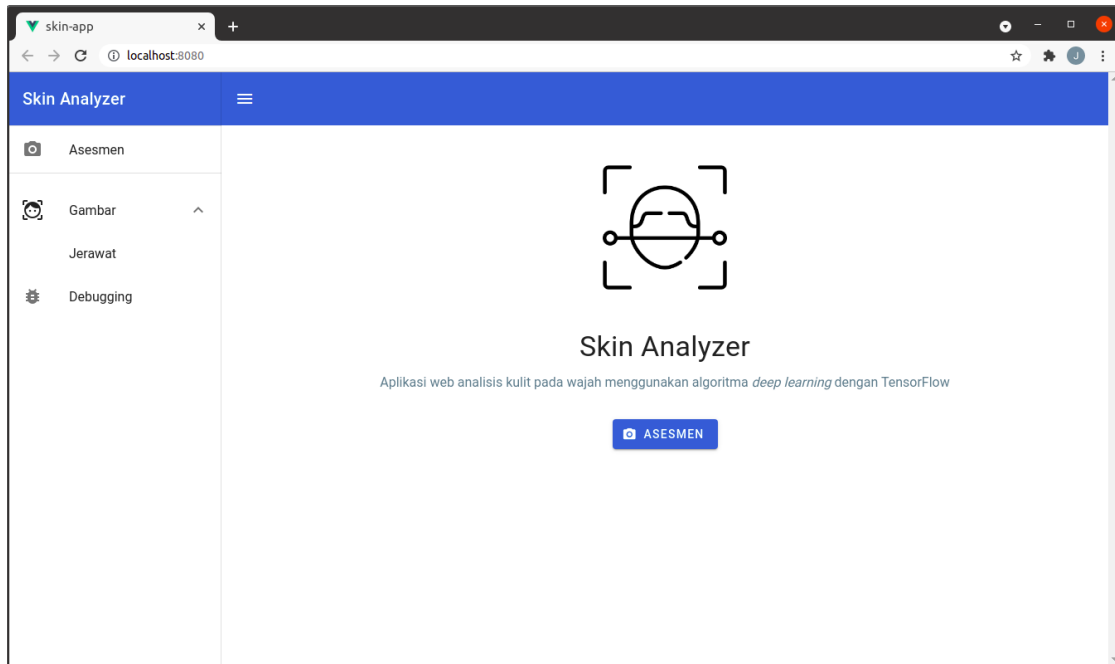


```
python3 app.py run
> python3 app.py run
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 103-290-724
```

Gambar 5.22 Tampilan pada terminal ketika web API berjalan

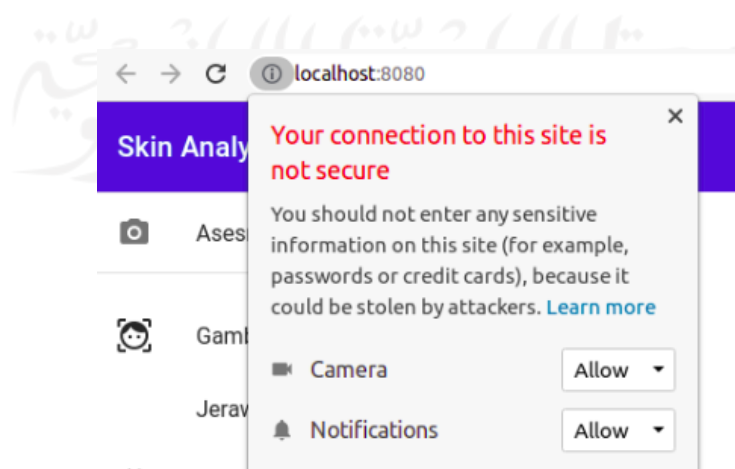
5.8.3 Pembuatan Antarmuka Web

Dari desain antarmuka yang telah dibuat kemudian ditulis menjadi halaman web menggunakan kode HTML dan CSS, komponen-komponen standar yang tersedia pada VueJs. Beberapa fungsionalitas pada halaman dibuat menggunakan program JavaScript yaitu VueJs untuk menangani interaksi pengguna pada *browser*, diantaranya untuk menampilkan pilihan sumber foto, mengambil foto dari kamera, *upload* foto dari komputer dan menampilkan hasil pendeteksian. Halaman utama dari antarmuka aplikasi dapat dilihat pada Gambar 5.23.

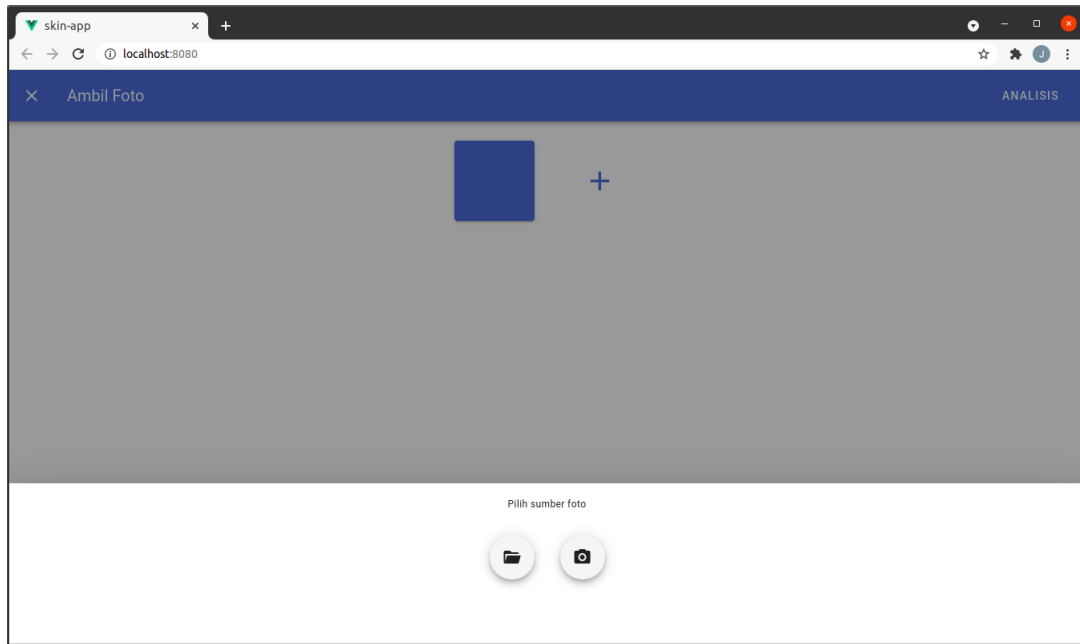


Gambar 5.23 Halaman depan aplikasi

Pada halaman *upload* foto, ketika pengguna menekan tombol “Tambah Foto” maka akan muncul dua pilihan, *File Manager* atau *Camera*. Halaman *upload* foto dapat dilihat pada Gambar 5.25 Jika tombol *File Manager* yang ditekan maka menampilkan pilihan berkas dari sistem operasi sedangkan jika tombol *Camera* yang ditekan maka aplikasi akan mengakses kamera dari perangkat. Pengguna harus memastikan bahwa izin untuk mengakses kamera pada *browser* telah diizinkan, jika tidak aplikasi tidak dapat menampilkan gambar dari kamera. Contoh hak akses kamera di *browser* Google Chrome dapat dilihat pada Gambar 5.24.

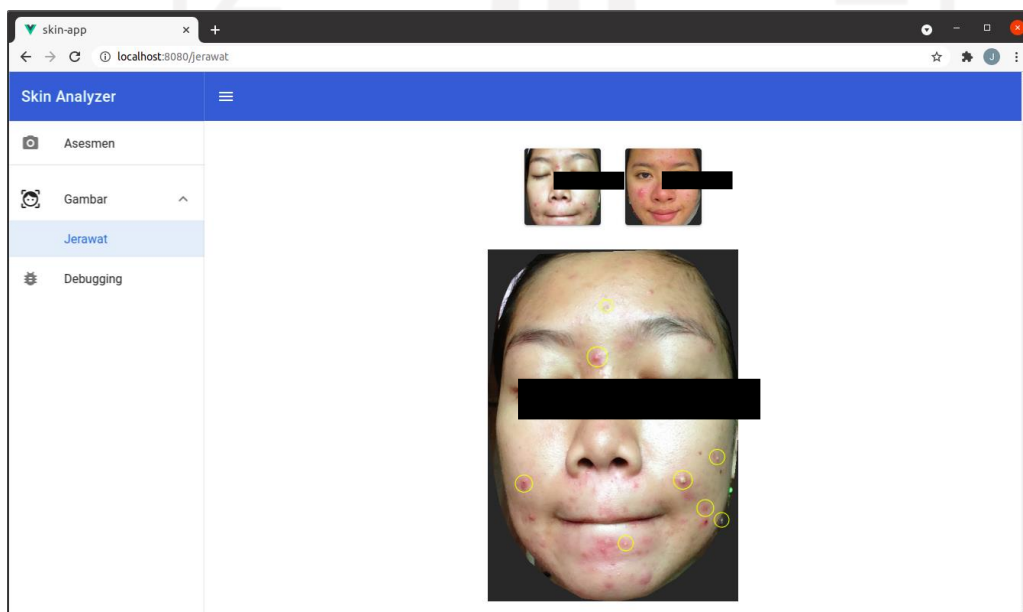


Gambar 5.24 Hak akses kamera pada Google Chrome



Gambar 5.25 Tampilan halaman unggah gambar

Apabila pengguna telah mengunggah foto maka akan ada tombol “Analisis” untuk melakukan pendeteksian. Setelah proses pendeteksian selesai maka akan ditampilkan foto yang sudah diberi tanda lingkaran pada objek jerawat yang terdeteksi pada masing-masing foto yang diunggah. Contoh tampilan hasil pendeteksian jerawat dapat dilihat pada Gambar 5.26.

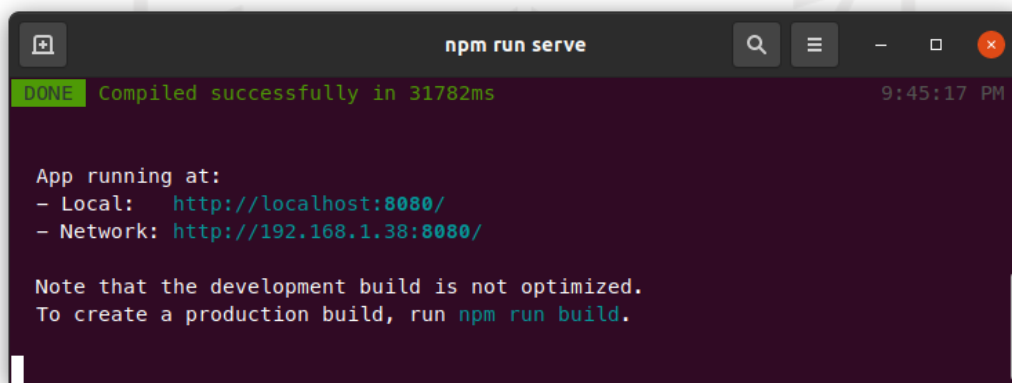


Gambar 5.26 Tampilan halaman hasil pendeteksian jerawat

Antarmuka pengguna berjalan terpisah dan alamat URL berbeda dengan Web API. Aplikasi diakses menggunakan alamat URL dari antarmuka pengguna ini. Antarmuka pengguna dijalankan dengan menggunakan perintah yang ditunjukkan Gambar 5.27. Perintah tersebut adalah perintah untuk melakukan kompilasi program sehingga dapat berjalan pada *server* (*runtime environment* Node.js). Dapat dilihat pada Gambar 5.28 antarmuka web sudah berjalan pada URL <http://localhost:8080/> (komputer lokal).

```
npm run serve
```

Gambar 5.27 Perintah untuk menjalankan antarmuka aplikasi



```
npm run serve
DONE Compiled successfully in 31782ms 9:45:17 PM
App running at:
- Local: http://localhost:8080/
- Network: http://192.168.1.38:8080/
Note that the development build is not optimized.
To create a production build, run npm run build.
```

Gambar 5.28 Tampilan terminal ketika antarmuka aplikasi berjalan

5.9 Evaluasi Hasil

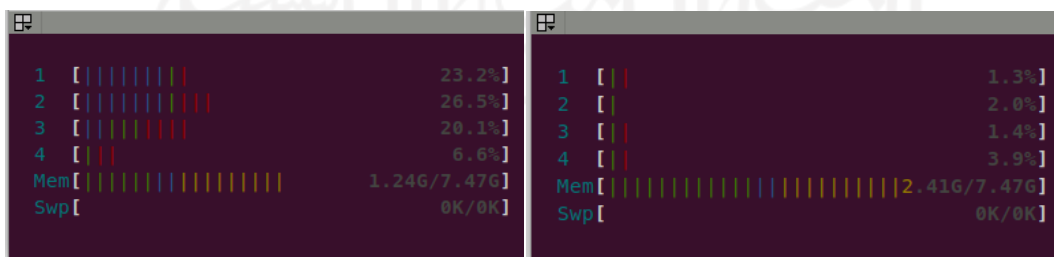
Model deteksi objek menggunakan *deep learning* membutuhkan dataset yang telah diberi anotasi berupa *bounding box* dan nama objeknya. Dari data yang terkumpul terhitung masih terbatas sehingga perlu diperbanyak untuk menambah variasi data. Maka dari itu proses augmentasi citra dilakukan, data yang awalnya berjumlah 160 untuk data *training* setelah dilakukan augmentasi menjadi 800 gambar. Namun jumlah itu masih kurang karena menurut (Goodfellow et al., 2015) data yang diperlukan paling tidak 5.000 data gambar untuk mencapai model pendeteksian yang optimal.

Proses pelatihan model deteksi objek dengan *deep learning* membutuhkan komputasi *hardware* yang tinggi. Percobaan menjalankan pelatihan model dengan TensorFlow pada laptop yang memiliki kapasitas CPU Intel i5 2,5Ghz, RAM 8GB dan GPU Nvidia 1GB tidak mampu berjalan, di awal proses berjalan muncul pesan “*out of memory*”, walaupun *batch*

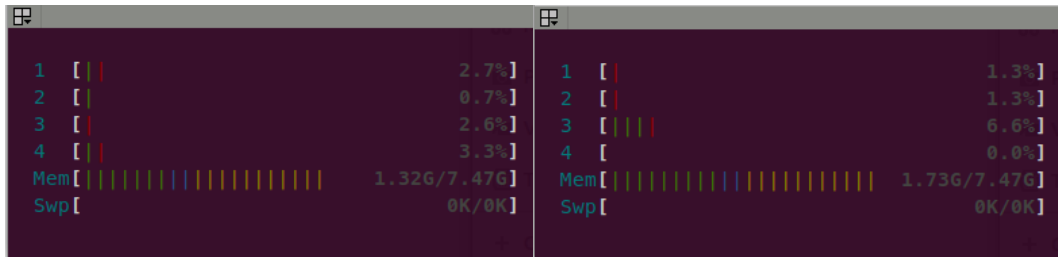
diatur menjadi paling kecil. Kemudian pelatihan model dipindahkan ke Google Colab dengan kapasitas CPU Intel Xeon 2,3 GHz, RAM 13GB, GPU Nvidia Tesla K80 12GB.

Hasil model pendeteksian menggunakan *deep learning* dengan model SSD ResNet50 V1 FPN 640x640 mendapatkan hasil yang memuaskan. Hal ini ditunjukkan dari nilai *total loss* yang sudah dapat mencapai 0,135, nilai mAP sebesar 42.1%, dan AR (*average recall*) sebesar 32%. Jika dibandingkan dengan anotasi yang dilakukan oleh dokter kulit secara manual, model pendeteksian memperoleh nilai *recall (sensitivity)* sebesar 77,2% dan *precision* 70,3%. Hal ini menunjukkan bahwa dari dataset pengujian yang dimiliki, model mampu mendeteksi jerawat cukup baik walaupun ada beberapa objek bukan jerawat yang masih terdeteksi.

Implementasi model pendeteksian menggunakan model *deep learning* pada web dapat berjalan dengan baik. Model dijalankan menggunakan TensorFlow Serving mengkonsumsi memori perangkat lebih sedikit dibandingkan dengan langsung membukanya dengan *library* TensorFlow pada web API. Perbandingannya dapat dilihat pada Gambar 5.29 dan Gambar 5.30. Dari situ dapat dilihat bahwa menjalankan model dengan TensorFlow Serving membutuhkan memori yang lebih sedikit yaitu sebesar 0,41G sedangkan tanpa menggunakan TensorFlow Serving sebesar 1,17G. Perbandingan rata-rata kecepatan pendeteksian antara keduanya yaitu 2,53 detik dengan TensorFlow Serving dan 2,71 detik apabila tidak, menggunakan TensorFlow Serving membutuhkan waktu *inference* yang lebih sedikit. TensorFlow tidak hanya berisi program untuk menjalankan model namun juga program untuk pelatihan model, ini menjadi indikasi mengapa mengkonsumsi memori yang lebih besar karena program yang tidak dibutuhkan tetap dibuka walaupun kebutuhannya hanya untuk menjalankan model saja.



Gambar 5.29 Perbandingan konsumsi memori tanpa menggunakan TensorFlow Serving



Gambar 5.30 Perbandingan konsumsi memori menggunakan TensorFlow Serving

Proses pendeteksian dengan aplikasi dapat berjalan dalam hitungan detik. Dari uji coba pendeteksian menggunakan aplikasi, mulai dari proses *upload* gambar sampai menampilkan hasilnya, rata-rata pendeteksian memerlukan waktu 2,77 detik per gambar. Apabila gambar yang diunggah memiliki ukuran gambar yang semakin besar maka proses *upload* juga semakin lama.

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan uji coba dan hasil analisis yang telah dilakukan, maka dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Model pendeteksian menggunakan model *deep learning* dapat dilakukan tanpa perlu mengekstrak ciri dari objek secara manual karena di dalam model *deep learning* berbasis CNN ekstraksi ciri dilakukan melalui proses konvolusi. Pelatihan model memerlukan dataset gambar yang sudah diberi anotasi berupa *bounding box* beserta nama objeknya.
2. Hasil pendeteksian jerawat menggunakan *deep learning* dinilai dapat berjalan dengan baik. Jerawat pada wajah dapat dideteksi dengan nilai mAP dari model sebesar 42.2% dan AR (*average recall*) sebesar 32%. Jika dibandingkan dengan anotasi yang dilakukan oleh dokter kulit secara manual, model pendeteksian memperoleh nilai *recall (sensitivity)* sebesar 77,2% dan *precision* 70,3%.
3. Model pendeteksian objek dengan *deep learning* dapat diterapkan pada aplikasi web dengan membuat layanan berupa web API untuk mengakses modelnya. Model dapat dijalankan secara terpisah dengan menggunakan TensorFlow Serving yang membutuhkan memori dan waktu *inference* yang lebih sedikit.
4. Aplikasi pendeteksian jerawat sudah dapat mendeteksi dan menghitung jerawat pada wajah dari citra digital yang diunggah pengguna. Pengguna cukup mengunggah foto pada aplikasi dan mendapatkan hasil pendeteksian jerawatnya.

6.2 Saran

Berdasarkan penelitian yang telah dilakukan, maka dapat diberikan beberapa saran sebagai berikut:

1. Jumlah dataset yang digunakan untuk pelatihan model *deep learning* perlu diperbanyak untuk meningkatkan kualitas dari pendeteksian jerawat karena jika merujuk pada (Goodfellow et al., 2015), data yang diperlukan paling tidak 5.000 data yang telah diberi label sedangkan pada penelitian ini hanya baru terkumpul 200 data atau 840 data jika dengan augmentasi.

2. Apabila ingin mengunggah aplikasi ke *cloud* maka pastikan *server* memiliki GPU dan RAM yang tinggi > 2GB agar dapat menjalankan model TensorFlow.
3. Model pendeteksian jerawat dapat dikembangkan lebih lanjut untuk aplikasi berbasis mobile seperti Android dan iOS dengan menggunakan web API yang telah dibuat.



DAFTAR PUSTAKA

- Alamdari, N., Tavakolian, K., Alhashim, M., FAAD, M., & Fazel-Rezai, R. (2016). *Detection and Classification of Acne Lesions in Acne Patients: A Mobile Application*.
- Andriy, B. (2019). *The Hundred-Page Machine Learning*.
- Bezdan, T., & Bačanin, N. (2019). *Convolutional Neural Network Layers and Architectures*. 445–451. <https://doi.org/10.15308/sinteza-2019-445-451>
- Carranza-García, M., Torres-Mateo, J., Lara-Benítez, P., & García-Gutiérrez, J. (2021). On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*, *13*(1), 1–23. <https://doi.org/10.3390/rs13010089>
- Chantharaphaichit, T., Uyyanonvara, B., Sinthanayothin, C., & Nishihara, A. (2015). Automatic Acne Detection for Medical Treatment. *2015 6th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES) Automatic*.
- Darmawan, A., Rositasari, A., & Muhimmah, I. (2020). The Identification System of Acne Type on Indonesian People's Face Image. *IOP Conference Series: Materials Science and Engineering*, *803*(1). <https://doi.org/10.1088/1757-899X/803/1/012028>
- Goodfellow, I., Bengio, Y., & Courville, A. (2015). *Deep Learning*.
- Grinberg, M. (2014). *Flask Web Development: Developing Web Applications with Python*.
- Hayashi, N., Akamatsu, H., Kawashima, M., Ito, M., Otsuki, M., Kawashima, M., Hayashi, N., Tsuboi, R., Nakagawa, H., Watanabe, S., Matsunaga, K., Akamatsu, H., Miyachi, Y., Furukawa, F., Iwatsuki, K., Kubota, Y., Tokura, Y., & Furue, M. (2008). Establishment of grading criteria for acne severity. *Journal of Dermatology*, *35*(5), 255–260. <https://doi.org/10.1111/j.1346-8138.2008.00462.x>
- Junayed, M. S., Jeny, A. A., Atik, S. T., & Neehal, N. (2019). AcneNet - A Deep CNN Based Classification. In *2019 12th International Conference on Information & Communication Technology and System (ICTS)*.
- Kusumadewi, S. (2003). *Artificial Intelligence (Teknik dan Aplikasinya)*. Graha Ilmu.
- Lauret, A. (2019). *The Design of Web APIs* (J. Stout (ed.)). Manning Publications Co.
- Liu, W., Angelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2015). *SSD: Single Shot MultiBox Detector*. https://doi.org/10.1007/978-3-319-46448-0_2
- Lucky, A. W., Barber, B. L., Girman, C. J., Williams, J., Ratterman, J., Waldstreicher, J.,

- Cincinnati, D., Ohio, R., & Jersey, N. (1996). *A multirater validation of acne lesion counting study to assess the reliability.*
- Macrae, C. (2018). *Vue.js: Up and Running : Building Accessible and Performant Web Apps.*
- Masterson, K. N. (2018). Acne Basics. *Journal of the Dermatology Nurses' Association*, 10(1S), S2–S10. <https://doi.org/10.1097/jdn.0000000000000361>
- May, P. (2019). *Improved Image Augmentation for Convolutional Neural Networks by Copyout and CopyPairing.* <http://arxiv.org/abs/1909.00390>
- Padilla, R., Netto, S. L., Da Silva, E. A. B., & Netto, S. L. (2020). *A Survey on Performance Metrics for Object-Detection Algorithms.* <https://doi.org/10.1109/IWSSIP48289.2020>
- Patnaik, S. K., Sidhu, M. S., Gehlot, Y., Sharma, B., & Muthu, P. (2018). Automated skin disease identification using deep learning algorithm. *Biomedical and Pharmacology Journal*, 11(3), 1429–1436. <https://doi.org/10.13005/bpj/1507>
- Prasetyo, D., Muhimmah, I., & Kurniawardhani, A. (2018). *PENGOLAHAN CITRA PADA FOTO.*
- Ramli, R., Malik, A. S., Hani, A. F. M., & Jamil, A. (2012). Acne analysis, grading and computational assessment methods: An overview. In *Skin Research and Technology* (Vol. 18, Issue 1, pp. 1–14). <https://doi.org/10.1111/j.1600-0846.2011.00542.x>
- Rashataprucksa, K., Chuangchaichatchavarn, C., Triukose, S., Nitinawarat, S., Pongprutthipan, M., & Piromsopa, K. (2020). Acne detection with deep neural networks. *PervasiveHealth: Pervasive Computing Technologies for Healthcare*, 53–56. <https://doi.org/10.1145/3421558.3421566>
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). *Dive into Deep Learning.*
- Zhao, T., Zhang, H., & Spoelstra, J. (2019). *A Computer Vision Application for Assessing Facial Acne Severity from Selfie Images.* <https://youtu.be/7tqJsms0viI>
- Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). *Object Detection in 20 Years: A Survey.* <http://arxiv.org/abs/1905.05055>

LAMPIRAN

