Aplikasi Web Pendeteksi Jerawat Pada Wajah Menggunakan Algoritma *Deep Learning* dengan TensorFlow

1st July Arifianto
Program Studi Informatika – Program Sarjana
Universitas Islam Indonesia
Yogyakarta
17523216@students.uii.ac.id

2nd Izzati Muhimmah*
Jurusan Informatika
Universitas Islam Indonesia
Yogyakarta
izzati@uii.ac.id

Abstract*—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. Penelitian ini merupakan penelitian mengenai pengembangan aplikasi visi komputer berbasis web untuk melakukan pendeteksian jerawat pada foto wajah megnggunakan algoritma *deep learning* berbasis *convolution neural network* (CNN) dengan Tensorflow. Penelitian ini bertujuan untuk menghasilkan aplikasi yang dapat digunakan oleh kinik dermatorlogi sebagai asesmen perawatan kulit berjerawat. Segmentasi kulit wajah dilkakukan dengan algoritma pendeteksian wajah dan pendeteksian landmark wajah. Proses pelatihan model pendeteksian jerawat menggunakan metode *transfer learning* dari model SSD ResNet50 V1 FPN 640x640. Hasil dari model pelatihan memiliki nilai *sensitivity* 77,3%, *specificity* 47,3%, dan *accuracy* 63,2%. Aplikasi yang dikembangkan sudah dapat mendeteksi dan menandai jerawat yang ada pada foto wajah dengan kecepatan rata-rata 2,77 detik untuk setiap gambar.

Keywords—visi komputer, jerawat wajah, deep learning, TensorFlow

I. PENDAHULUAN

Sekarang, memiliki kulit yang bagus menjadi tren 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 (*Acne vulgaris*) adalah masalah pada kulit atau kondisi kulit yang umum terjadi. Penyebab jerawat biasanya terjadi karena penyumbatan kelenjar minyak di kulit atau infeksi bakteri. Jerawat bisa ditandai dengan kulit bersisik merah (*seborrhea*), komedo, papula, nodul, bintil, dan jarigan parut [1, 2]. Ada dua jenis jerawat, yang pertama adalah lesi non-inflamasi, dan yang lainnya adalah lesi inflamasi. Pada kondisi yang parah, jerawat besar membengkak di bawah kulit dan meradang sehingga mengakibatkan benjolan menyakitkan yang menyebabkan jarigan parut. Meski tidak berbahaya, penderita sering mengalami gangguan emosi dan sosial karena ketidaknyamanan dan rasa malu yang ditimbulkan oleh jerawat. Jerawat muncul secara umum selama masa remaja, mempengaruhi sekitar 80-90% [3].

Secara tradisional, penilaian terhadap kulit berjerawat dilakukan oleh dokter kulit di lingkungan medis. Cara perawatan kulit berjerawat diresepkan oleh dokter kulit atau produk perawatan kulit yang dijual bebas [4]. Perawatan awal untuk kulit wajah dan diagnosis jerawat adalah salah satu prosedur yang diperlukan. Metode yang umumnya dilakukan yaitu meminta dokter kulit secara manual menandai tempat jerawat pada lembar penanda sesuai dengan lokasi jerawat yang telah diamati. Hasil penandaan terkadang masih kurang akurat dan dapat menghabiskan waktu dan tenaga bagi dokter kulit untuk menentukan tingkat keparahan.

Penggunaan alat pada masalah kulit sudah digunakan oleh beberapa klinik dematologi. Kebanyakan alat analisis kulit masih membutuhkan usaha manual untuk menentukan *region of interest* (ROI) sebelum dilakukan pengamanatan dengan perangkat lunak, sebagian besar klinik menggunakan VISIA [5]. Hal ini memakan waktu dan tidak dapat diulang untuk area yang berbeda dan alat ini sangat mahal.

Berdasarkan permasalahan di atas, penelitian ini dialukuan untuk membuat suatu aplikasi untuk mendeteksi jerawat di wajah pasien secara otomatis tanpa cara manual dari foto wajah untuk menghitung atau mendiagnosis apakah area tersebut berjerawat atau tidak, sebagai asesmen perawatan kulit berjerawat. Selanjutnya hal ini akan dibahas pada bagian berikut. Bagian II, menjelaskan landasan teori yang menjadi dasar dari penelitian. Bagian III, menjelaskan tentang metode terkait yang diusulkan pada penelitian. Sedangkan bagian IV menjelaskan tentang hasil dan pembahasan dari penelitian. Selanjutnya yang terakhir bagian kesimpulan akan dibahas pada bagian V.

II. LANDASAN TEORI

A. *Penelitian Terkait*

Topik dari penelitian ini merupakan bagian dari bidang visi komputer tentang pendeteksian objek. Penelitian sebelumnya, untuk melakukan pendeteksian jerawat menggunakan rentang warna HSV atau RGB untuk mendapatkan area kulit wajah dan objek jerawat. Sistem yang dikembangkan mencoba mendeteksi jerawat berdasarkan titik-titik yang terindikasi jerawat pada wajah, kemudian sistem akan menyeleksi berdasarkan warna, bentuk, dan luas area piksel. Metode lain juga diusulkan, proses dimulai dengan masukan gambar asli, *resize* gambar, segmentasi kulit, *sharpening*, seleksi ciri (luas, bentuk, warna) dan diakhiri proses *marking* objek jerawat. Metode yang diusulkan untuk mendeteksi jerawat didapatkan nilai *sensitivity* 0,464 dan nilai *specificity* sebesar 0,971 [6, 7]. Hasil ini menunjukan bahwa metode yang digunakan masih belum terlalu bagus untuk mendeteksi jewarat. Namun, sudah sangat baik untuk mendeteksi objek bukan jerawat.

Penelitian yang dilakukan untuk pendeteksian jerawat menggunakan algoritma *machine learning* [8]. Penelitian ini menggunakan dua metode klasifikasi *machine learning*. Akurasi rata-rata klasifikasi untuk membedakan bekas luka jerawat non-inflamasi dan inflamasi untuk metode FCM dan SVM linier adalah masing-masing 80% dan 66,6%. Pada klasifikasi kulit normal dari orang yang berjerawat dengan metode FCM akurasi performanya 100%.

Penggunaan rentang warna untuk pendeteksian objek tidak bisa berjalan dengan baik apabila warna kulit pasien terlalu bervariasi dan kondisi pencahanyaan pengambilan foto yang berbeda-beda. Kami mengusulkan penggunaan algoritma pendeteksian wajah [9] dan landmark wajah [10, 11] untuk mendapatkan area wajah pasien. Sedangkan untuk mendeteksi jerawat menggunakan *deep learning* berbasis *convolution neural network* (CNN) pada TensorFlow.

B. *Jerawat (Acne Vulgaris)*

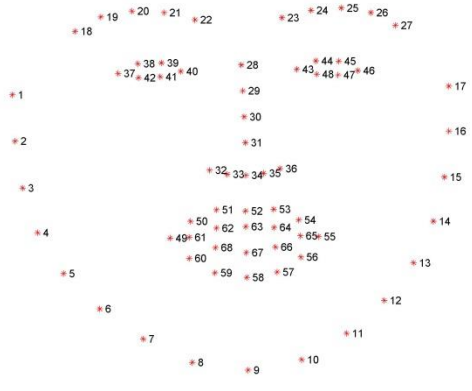
Jerawat adalah kondisi kulit yang terjadi ketika kelenjar *sebaceous* (penghasil minyak) di kulit dan di sepanjang batang rambut tersumbat dan meradang, serta terinfeksi bakteri. Jika jerawat membentuk pembengkakan yang kuat di bawah kulit, meradang dan bisa berkembang menjadi benjolan yang menyakitkan dan menyebabkan jaringan parut. Tempat munculnya jerawat biasanya terdapat di wajah, leher, dada, bahu, atau punggung [1]. Gejala yang dapat dilihat pada pasien berjerawat diantaraya sebagai berikut,

- *Whiteheads*
- Kulit berminyak meningkat
- Komedo - bintik hitam dengan pori-pori terbuka di bagian tengah
- Benjolan merah kecil
- Pembengkakan merah atau benjolan yang tampak berisi nanah (jerawat / pustula) - di wajah, dada, bahu, leher, dan / atau punggung atas
- Benjolan besar, meradang, merah, berisi cairan (nodul) di bawah kulit
- Benjolan besar, meradang, merah, lunak (kista) di bawah kulit yang bisa menjadi sebesar 2,5 cm.

Tetapi jarawat paling sering terjadi pada wajah karena berbagai faktor etiologi dari jerawat. Jerawat sendiri memiliki beberapa gejala klinis, diantaranya dibagi menjadi dua kelompok, yaitu lesi non-inflamasi dan lesi inflamasi. Lesi non-inflamasi terdiri dari komedo hitam (*blackhead comedoes*) yang terjadi karena oksidasi melanin dan komedo putih (*whitehead comedones*). Meskipun jerawat bisa terjadi di mana saja di wajah, area jerawat yang dominan adalah dahi, pipi, dan dagu.

C. *Pendeteksian Landmark Wajah*

Deteksi landmark wajah merupakan metode untuk mengekstrak fitru menonjol pada wajah. Landmark wajah mewakili titik fitur menonjol di wajah, seperti mata, hidung, dan mulut. Landmark wajah dapat digunakan untuk beragam tugas seperti pengenalan wajah, deteksi tatapan, pelacakan orang, pengenalan emosi, dan rias virtual. Sejauh ini, banyak penelitian telah dilakukan dengan tujuan mencapai ekstraksi landmark yang efisien dari gambar wajah. Proses pendeteksian dan pelacakan yang menggunakan terlalu banyak titik fitur landmark biasanya membutuhkan waktu pemrosesan yang berlebihan. Sebaliknya, apabila mengandalkan terlalu sedikit titik fitur tidak dapat menghasilkan pendeteksian yang akurat. Metode yang diusulkan telah mampu mendapatkan 68 titik fitur menonjol di wajah secara *real time*, menggunakan pustaka Dlib, dapat dilihat pada Gambar 1. Metode ini sudah dapat menyesuaikan posisi dari wajah sehingga dapat dijadikan acuan pengambilan area wajah [10, 11].



Gambar 1. Titik landmark wajah pada pustaka Dlib

D. *Tensorflow Object Detection*

TensorFlow adalah *platform open source end-to-end* untuk *machine learning* yang dibuat oleh tim Google Brain. TensorFlow memiliki perkakas, pustaka, dan sumber daya komunitas yang komprehensif dan fleksibel yang memungkinkan para peneliti dalam pengembangan *machine learning* dan tim pengembang dapat dengan mudah membangun dan menerapkan aplikasi yang didukung *machine learning*. Arsitekturnya yang fleksibel memungkinkan penerapan komputasi yang mudah di berbagai *platform* (CPU, GPU, TPU), dan dari perangkat seperti desktop, *cloud*, maupun perangkat *mobile* [12, 13].

TensorFlow menyediakan *Object Detection API* untuk mempermudah pengembangan aplikasi *deep learning* untuk membuat perangkat lunak pendeteksian objek. TensorFlow *Object Detection API* adalah *open source framework* yang dapat digunakan untuk mengembangkan, melatih, dan menggunakan model deteksi objek. Sistem ini sudah banyak diterapkan pada berbagai produk Google antara lain pencarian gambar, deteksi wajah dan plat nomor kendaraan pada Google Streetview, Google Assistant, Waymo atau *self driving car*, dan lain-lain.

TensorFlow Object Detection juga menyediakan banyak model yang telah dilatih dengan dataset *Common Object in Context* (COCO) 2017 yang terdiri 300.000 gambar berisi 90 kategori objek. Model-model tesebut memiliki kecepatan dan tingkat akurasi masing-masing, sesuai dengan arsiktektur modelnya [14, 15]. Salah satunya adalah model SSD ResNet50 V1 FPN 640x640.

SSD ResNet50 V1 FPN 640x640 merupakan arsitektur deteksi objek FPN SSD berdasarkan ResNet-50. ResNet-50 adalah *convolution neural network (CNN)* yang memiliki kedalaman 50 lapisan, ResNet merupakan kependekan dari *residual network*. Setiap unit *residual* dalam jaringan terdiri dari dua lapisan konvolusi tanpa lapisan penyatuan, normalisasi *batch*, aktivasi ReLU, dan kernel 3x3 [20]. Satu tahapan pada model tesebut menggunakan *Feature Pyramid Network* (FPN) *backbone* pada arsitekturnya [21]. *Input* dari model SSD ResNet50 V1 FPN 640x640 berupa *tensor* berukuran 1x640x640x3, ukuran tersbut menunjukan nilai *batch size* berjumlah 1 dengan gambar RGB berukuran 640x640. *Output* dari model berupa nama kelas objek, nilai probabilitas, koordinat kotak deteksi, dan jumlah kotak deteksi yang diprediksi.

E. *Aplikasi Web*

Aplikasi web menjadi populer karena kemudahan tersedianya aplikasi klien untuk mengaksesnya. Kemampuan untuk memperbarui dan memelihara aplikasi web tanpa harus mendistribusikan dan menginstalasi perangkat lunak pada kemungkinan ribuan komputer klien merupakan alasan kunci popularitasnya. Aplikasi web yang umum misalnya *webmail*, toko ritel, lelang *online*, wiki, forum, maupun *weblog* [16].

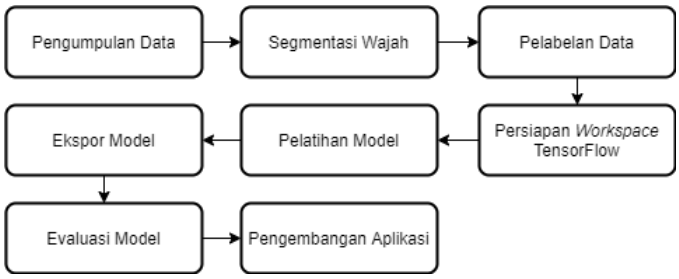
Banyak keuntungan yang diberikan oleh aplikasi berbasis web daripada aplikasi berbasis desktop, sehingga aplikasi berbasis web telah diadopsi oleh perusahaan sebagai bagian dari strategi teknologi informasinya, karena beberapa alasan:

- Akses informasi mudah
- *Setup* server lebih mudah
- Informasi mudah didistribusikan
- Bebas *platform*, informasi dapat disajikan oleh *browser* web pada berbagai jenis sistem operasi karena adanya standar dokumen berbagai tipe data dapat disajikan

Terdapat beberapa jenis arsitektur aplikasi web secara umum yang digunakan, salah satunya adalah *single-page application* (SPA). Ini merupakan aplikasi yang bekerja di dalam *browser* yang tidak membutuhkan *reload* halaman saat setiap kali terjadi interaksi pada aplikasi. Terpisah menjadi dua bagian, *frontend* dan *backend*. Dalam SPA, semua kode HTML, JavaScript, dan CSS yang diperlukan diambil oleh *browser* dengan pemuatan halaman tunggal. Permintaan ke server biasanya menghasilkan data mentah seperti XML, JSON atau HTML baru. Setelah data diterima, data diterjemahkan menggunakan JavaScript untuk memperbaharui area dari DOM [17].

III. METODOLOGI

Tahapan penelitian ini meliputi pengumpulan data, segmentasi wajah, pelabelan data, persiapan *workspace* TensorFlow, pelatihan model, ekspor model, evaluasi model dan pengembangan aplikasi. Tahapan pada penelitian dapat dilihat pada Gambar 2.



Gambar 2. Tahapan penelitian

A. *Pengumpulan Data*

Pengumpulan gambar pada penelitian ini dilakukan menggunakan kamera *smartphone*. Gambar berupa foto wajah orang-orang yang memiliki masalah jerawat pada wajah. Sudut pengambilan foto pada area wajah dari sisi depan, serong kanan dan serong kiri. Kondisi yang perlu dipastikan dalam pengambilan foto diataranya,

- wajah tanpa *makeup*,
- pencahayaan cukup,

- latar belakang kontras dengan warna kulit,
- tidak *backlight*,
- fokus ke area wajah,
- tidak menggunakan filter,
- dan, kamera beresolusi minimal 8MP.

Peneliti juga menggunakan gambar yang dikumpulkan dari Google dengan kata kunci ‘wajah berjerawat’ untuk memperbanyak data.

B. Segmentasi Wajah

Deteksi wajah dilakukan untuk mengambil ROI dari gambar berupa area kulit wajah. Algoritma pendeteksian wajah dan landmark wajah digunakan pada tahap ini untuk mendapatkan area kulit wajah secara otomatis. Hal ini dilakukan dengan menggunakan pustaka OpenCV dan Dlib.

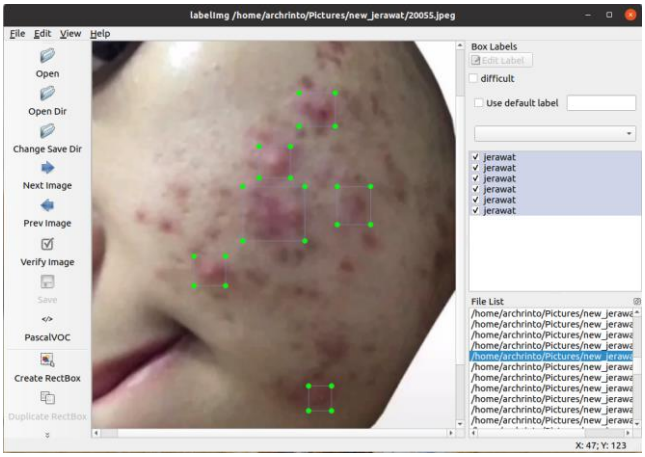
Hasil dari pendeteksian wajah menggunakan pustaka OpenCV yang menerapkan *Haar Cascades Classifier* berupa *array* koordinat *x* dan *y* titik awal lokasi piksel wajah beserta *lebar* dan *tinggi* areanya. Hasil tersebut kemudian digunakan untuk mengambil gambar area wajah dengan melakukan *cropping* sehinnga didapatkan gambar baru yang hanya berisi wajah. Gambar tersebut masih menyisakan latar belakang dari gambar lama sehingga perlu dilakukan segmentasi untuk area kulit wajah.

Segmentasi kulit wajah dilakukan dengan melakukan pendeteksian landmark wajah. Hasil dari pendeteksian menggunakan pustaka Dlib berupa 68 titik landmark wajah. Titik-titik tersebut kemudian diambil nomor 1 sampai 17 yang merupakan landmark yang mewakili tepian dari wajah sehing-ga dapat digunakan sebagai acuan untuk mendapatkan area kulit wajah, lihat pada Gambar 1. Dari titik-titik tersebut kemudian dikonversi menjadi garis batas dengan menggunakan metode *Convex Hull*. Garis batas yang didapat kemudian dipakai untuk membuat *mask* gambar, berupa *array* bertipe data *boolean* berukuran *n* x *m* x *1*. Area di dalam garis batas bernilai *true* yang mewakili area kulit wajah sedangkan di luar garis batas bernilai *false* yang mewakili area yang dianggap latar belakang. Proses *masking* kemudian dilkukan, mengahsilkan area kulit wajah yang tersegmentasi.

C. Pelabelan Data

Pelabelan merupakan kegiatan memberikan label pada suatu dataset. Metode yang sering digunakan untuk melakukan pelabelan adalah secara manual dengan menggunakan bantuan tenaga ahli. Tahap ini dilakukan untuk memberikan lebel pada objek-objek jerawat yang terdapat pada gambar wajah. Penandaan pada gambar berupa koordinat area (*x, y, lebar* dan *tinggi*) yang mewakili lokasi kumpulan piksel dari objek. Setiap penandaan objek diberi label ‘jerawat’ sebagai nama dari kelas objek. Pelabelan dilakukan untuk masing-masing file gambar kulit wajah yang sudah tersegmentasi.

Alat yang digunakan untuk pelabelan data yaitu aplikasi yang bernama LabelImg, alat anotasi gambar grafis. LabelImg ditulis dengan bahasa Python dan menggunakan Qt sebagai antarmuka grafisnya. Anotasi disimpan sebagai file XML dalam format PASCAL VOC, format yang digunakan oleh ImageNet. Selain itu, alat ini juga mendukung format YOLO dan CreateML.



Gambar 3. Pelabelan objek dengan LabelImg

D. Persiapan workspace TensorFlow

TensorFlow memiliki spesifikasi ruang kerja yang perlu dipersiapkan untuk membuat model pendeteksian objek. Persiapan ini meliputi persiapan dataset, pembuatan *LabelMap*, pembuatan *TensorFlow Record*, dan konfigurasi *pipeline* pelatihan model. Data yang sudah diberi label kemudian dilakukan pemisahan data untuk mempersiapkan data pelatihan dan data pengujian, masing-masing 80% dan 20% dari dataset yang dimiliki. Pembuatan *LabelMap* dilakukan untuk memetakan setiap label atau kelas objek yang ada pada dataset. *LabelMap* ini digunakan dalam proses pelatihan dan pendeteksian. Dataset pelatihan dan pengujian kemudian diubah ke dalam format yang disebut *TFRecord*. *TensorFlow Record* atau *TFRecord* merupakan format sederhana untuk menyimpan urutan catatan biner. Konfigurasi *pipline* pelatihan model dilakukan untuk menyesuaikan sumber daya yang dimiliki [18].

E. Pelatihan Model

Pelatihan model adalah proses untuk mendapatkan model pendeteksian jerawat pada wajah. Pelatihan model menggunakan metode *transfer learning*. *Transfer learning* merupakan istilah pada *machine learning* yang mengacu pada pemanfaatan pengetahuan dari suatu bidang ke bidang yang lain, merupakan salah satu solusi apabila memiliki data yang sedikit. Model yang sebelumnya dilatih digunakan untuk mengestrak fitur-fitur dari lapisan bawah jaringan saraf tiruan. Kemudian dari fitur-fitur tersebut dimanfaatkan untuk melatih model baru dengan tujuan pendeteksian jerawat. *Transfer learning* dilakukan dari model SSD ResNet50 V1 FPN 640x640. Model ini merupakan model *deep learining* berbasis

convolution neural network (CNN) yang menggunakan arsitektur ResNet-50.

Proses pelatihan dipantau untuk mengetahui perkembangan dari model yang dilatih, dapat dilihat dari nilai *total loss.* Nilai *total loss* merupakan hasil dari fungsi kerugian yang mewakili ketidaktepatan dari pendeteksian dalam klasifikasi dan lokalisasi, merupakan gabungan dari kerugian klasifikasi dan lokalisasi. Nilai kerugian dalam mengklasifikasikan objek diwakili oleh nilai *classification loss* dan nilai kerugian dalam menandai kotak pembatas lokasi objek diwakili oleh nilai *localization loss*. TensorFlow menghitung nilai *total loss* tersebut dari pengecekan performa model dengan menggunakan data pengujian selama proses pelatihan [19].

TensorFlow menyediakan fitur untuk memantau dan memvisualkan laporan perkembangan pelatihan model melalui TensorBoard. Model pendeteksian yang bagus paling tidak mencapai nilai *total loss* mendekati 2 atau secara idealnya 1 atau kurang dari 1 [18]. Nilai *total loss* yang lebih rendah tentunya lebih baik, namun nilai *total loss* yang sangat rendah perlu dihindari karena model tersebut mungkin *overfitting* pada dataset, yang berarti bahwa model tersebut akan berkinerja buruk ketika diterapkan pada gambar di luar dataset. Lamanya proses pelatihan bergantung pada performa dari PC atau server yang digunakan [18].

F. Ekspor Model

Ekspor model dilakukan untuk mengambil model yang sudah selesai dari proses pelatihan. Grafik inferensi dari model diekstrak yang nantinya digunakan untuk melakukan deteksi objek.

G. Evaluasi Model

Model hasil pelatihan kemudian dievaluasi dengan membandingkan hasil pendeteksian model dengan penandaan yang dilakuakn pakar. Evaluasi ini dilakukan untuk mengetahui hasil pendeteksian apakah sudah sesuai dengan hasil yang diperoleh dari ahli atau belum. Evaluasi dilakukan menggunakan *Single Decision Threshold*. Terdapat empat istilah yang digunakan, yaitu:

1) *True Positive (TP)*: Jika nilai sebenarnya dan nilai prediksi menghasilkan hasil positif.

2) *True Negative (TN)*: Jika nilai sebenarnya dan nilai prediksi menghasilkan hasil negative.

3) *False Positive (FP):* Jika nilai sebenarnya negative, tetapi menghasilkan hasil positif.

4) *False Negative (FN):* Jika nilai sebenarnya positif, tetapi menghasilkan hasil negative.

Single Decision Threshold kemudian dihitung untuk mencari nilai *sensitivity*, *specificity* dan *accuracy*. *Sensitivity* adalah parameter untuk mengukur persentase data positif yang diidentifikasi dengan benar. *Specificity* adalah parameter untuk mengukur persentase data negatif yang diidentifikasi dengan benar. *Accuracy* adalah nilai yang menggambarkan seberapa akurat sistem dapat mendeteksi data dengan benar [22]. Masing-masing nilai dihitung menggunakan Persamaan 3.1, 3.2, 3.3.

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.1}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

H. Pengembangan Aplikasi

Pengembangan aplikasi merupakan bagian dari upaya penerapan aplikasi untuk pendeteksian jerawat agar dapat digunakan oleh pengguna. Langkah-langkah yang diterapkan untuk mengembangkan aplikasi dalam penelitian ini adalah sebagai berikut:

1) *Identifikasi kebutuhan aplikasi*: Menganalisis kebutuhan aplikasi dari sisi kebutuhan program, menjelaskan proses-proses yang dilakukan oleh program aplikasi yang dibutuhkan oleh penguna.

2) *Perancangan aplikasi:* Tahapan yang menjelaskan bagaimana aplikasi akan dibuat. Rancangan sistem berupa diagram arsitektur aplikasi, diagram alir (*flowchart*) aplikasi dan rancangan antarmuka (*interface*).

3) *Implementasi:* Tahap pembuatan aplikasi, susuai rancangan yang telah dibuat. Bagian *frontend* aplikasi menggunakan Vue.js sebagai antarmuka pengguna dan bagian *backend* aplikasi menggunakan Python dengan *framework* Flask sebagai *web service*.

IV. HASIL DAN PEMBAHASAN

A. Pengumpulan Data

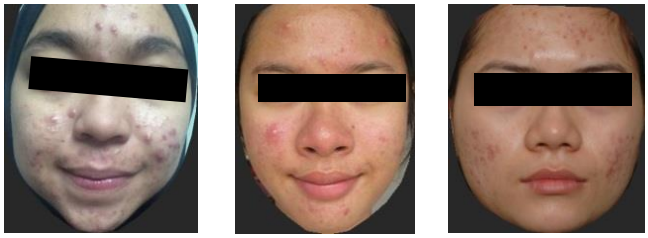
Pengumpulan data yang dikakukan menggunakan *smartphone* mendapatkan hasil sejumlah lima belas gambar dari lima orang reponden. Pengumpulan data dari Google sejumlah tiga puluh lima gambar wajah orang yang mengalami masalah jerawat. Total gambar yang terkumpul sejumlah lima puluh gambar. Gambar 4 menujukan contoh pengambilan gambar dari sisi serong kiri, sisi depan dan sisi serong kanan.

(a) Serong Kiri (b) Depan (c) Serong kanan

Gambar 4. Contoh pengambilan gambar

B. Pendeteksian Jerawat

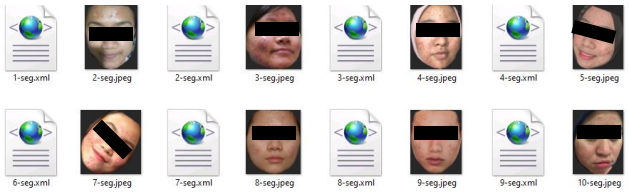
Gambar wajah orang yang memiliki masalah kulit jerawat yang telah dikumpulkan kemudian masuk ke tahap segmentasi kulit wajah untuk mendapatkan ROI. Gambar yang telah tersegmentasi kemudian disimpan dengan ukuran lebar 400 piksel dengan latar belakang gelap untuk memisahkan area kulit wajah pada gambar. Proses ini dilakukan otomatis menggunakan program, menghasilkan file-file gambar kulit wajah yang sudah tersegementasi.



Gambar 5. Segmentasi wajah

Gambar kulit wajah yang tersegmentasi kemudian masuk ke tahap pelabelan objek jerawat. Pelabelan dilakukan dengan cara menandai area-area spesifik pada citra yang merupakan objek jerawat, dapat dilihat pada Gambar 3. Anotasi menggunakan format PASCAL VOC. Hasil dari proses pelabelan ini berupa file XML yang berisi daftar koordinat area untuk masing-masing gambar, dapat dilihat pada Gambar 6.

Setelah dataset jerawat diperoleh, dari sekian banyak data tersebut dipisahkan menjadi dataset pembelajaran dan dataset pengujian. Masing-masing dataset dipisahkan pada *folder* yang berbeda.

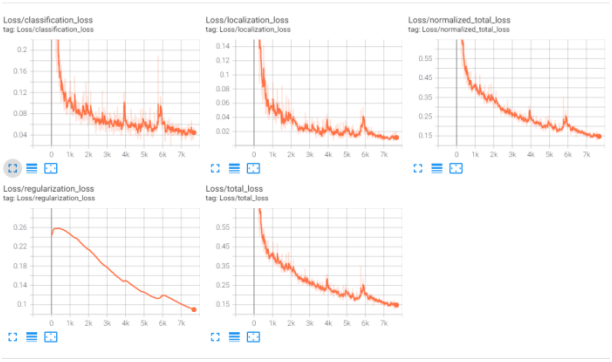


Gambar 6. Hasil pelabelan dengan LabelImg

TensorFlow memilki banyak model yang sebelumnya dilatih untuk pendeteksian objek. Mempertimbangkan kinerja dan efisiensi komputasi, peneliti memilih menggunakan model SSD ResNet50 V1 FPN 640x640, karena memberikan *trade-off* yang relatif baik antara kinerja dan kecepatan.

Langkah selanjutnya adalah menyiapkan *workspace* TensorFlow untuk pelatihan model pendeteksian objek. Persiapan meliputi pembuatan *LabelMap*, pembuatan *TFRecord*, dan konfigurasi *pipeline* pelatihan model. Pembuatan *LabelMap* memetakan isi dari dataset yang dimiliki, file berisi daftar kelas-kelas objek. Konversi dataset ke dalam format *TFRecord* dilakukan menggunakan program Python yang menghasilkan file *train.record* dan *test.record* untuk masing-masing dataset. Kemudian konfigurasi *pipeline* pelatihan model yang meliputi pengaturan jumlah kelas objek, jumlah *batch size*, pengaturan evaluasi, pengaturan lokasi dataset, lokasi *LabelMap*, dan lokasi *checkpoint* dari model.

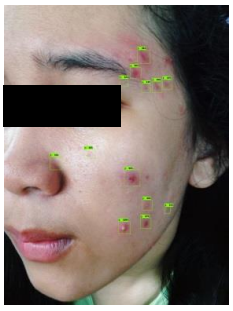
Setelah konfigurasi *pipeline* selesai kemudian dilanjutkan proses pelatihan model. Selama proses pelatihan nilai *total loss* dari model terus dipantau. Hasil dari proses pelatihan model ditunjukan pada grafik Gambar 11. Grafik dari *classification loss* menunjukan angka 0.04 dan *localization loss* menunjukan angka 0.01 sehinnga *total loss* bernilai 0.147. Capaian ini belum begitu menggembirakan karena seharusnya nilai dari *total loss* paling tidak diantara 1 sampai dengan 2 [18]. Hal ini dapat mengakibatkan model *overfitting* terhadap dataset yang dimiliki.



Gambar 7. Grafik pada TensorBoard

Evaluasi model yang sudah dilatih dilakukan menggunakan *Single Decision Threshold*. Perhitungan dilakukan dengan membandingkan hasil pendeteksian model dengan penandaan oleh pakar pada 20 gambar data pengujian. Hasil perhitungan diperoleh nilai *sensitivity*, *specificity*, dan *accuracy* dari model adalah 77,2%, 47,3% dan 63,2%. Dengan nilai *sensitivity* 77,2% berarti model sudah dapat melakukan pendeteksian jerawat cukup baik. Namun, dari nilai *specificity* 47,3% menunjukka masih perlu perbaikan dalam mendeteksi yang objek bukan jerawat. Nilai akurasi sebesar 63.2% mengindikasikan bahwa model masih memilki akurasi yang rendah untuk pendeteksian jerawat. Secara visual perbandigan pendeteksian model dan pakar dapat dilihat pada Gambar 8.

(a)



(b)

(c) (d)

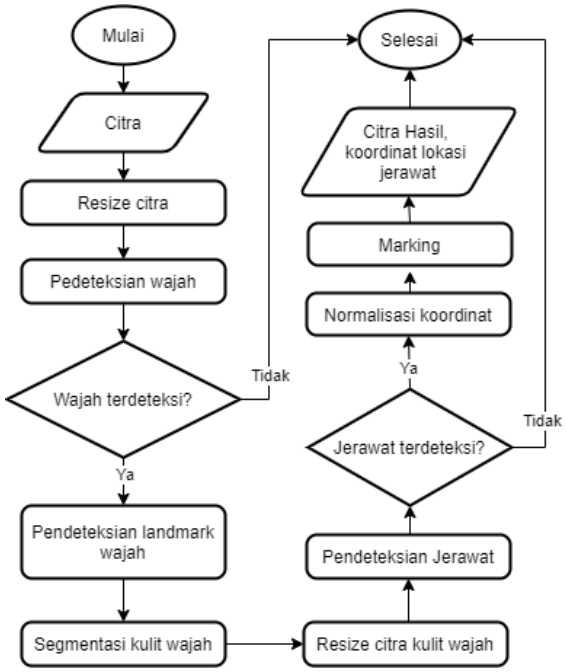
Gambar 8. Perbandingan hasil penandaan pakar dengan pendeteksian model. (a) dan (c) adalah penandaan pakar, sedangkan (b) dan (c) adalah deteksi dari model.

C. Pengembangan Aplikasi

Kebutuhan aplikasi meliputi kebutuhan program dan antarmuka aplikasi. Kebutuhan didasari oleh kebutuhan pengguna aplikasi yaitu untuk asesmen perawatan kulit berjerawat di klinik dermatologi. Berdasarkan identifikasi didapatkan kebutuhan umum aplikasi sebagai berikut,

- Aplikasi harus mampu menggungah gambar atau mengambil foto dari kamera
- Aplikasi harus mampu melakukan segmentasi wajah
- Aplikasi harus mampu melakukan pendeteksian jerawat
- Aplikasi harus mampu mengirimkan dan menampilkan data hasil pendeteksian jerawat

Langkah selanjutnya adalah perancangan aplikasi. Perancangan arsitektur aplikasi menggunakan arsitektur *single-page application* (SPA). Aplikasi terbagi menjadi dua bagian yaitu *frontend* sebagai antarmuka di sisi pengguna dan *backend* di sisi server sebagai *web service*. Pertukaran data antara *frontend* dan *backend* menggunakan HTTP *methods* dan data berformat JSON. Perancangan alur program aplikasi dapat dilihat pada Gambar 9.



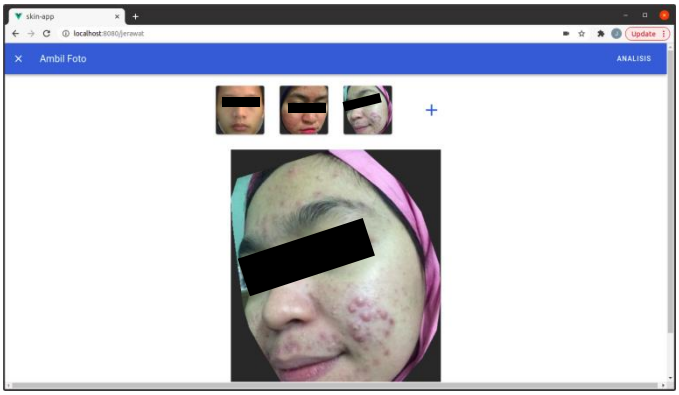
Gambar 9. Alur program aplikasi

Rancangan antarmuka dilakukan untuk merancang tampilan aplikasi yang nantinya digunakan pengguna. Pada aplikasi ini terdapat beberapa tampilan yaitu:

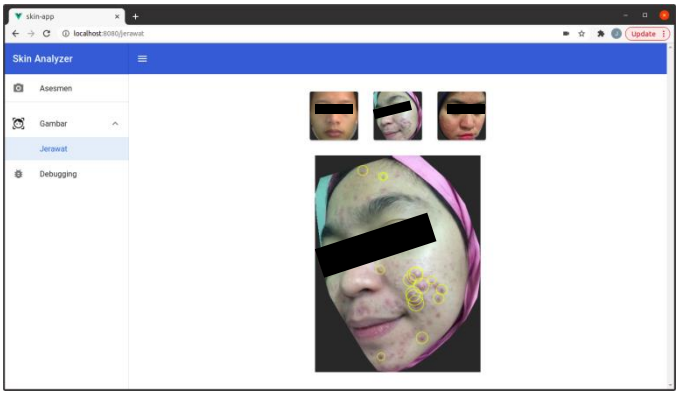
- Tampilan pengambilan gambar. Tampilan ini berfungsi untuk menggambil dan menampilkan gambar dari kamera pengguna. Pengambilan gambar dilakukan tiga kali untuk sisi wajah depan, serong kiri dan serong kanan.
- Tampilan hasil deteksi. Tampilan ini berfunsgi untuk menampilkan hasil pendeteksian jerawat dari foto yang diambil.

Langkah selanjutnya adalah implementasi. Implementasi dimulai dari sisi *frontend* yaitu pembuatan antarmuka pengguna. Antarmuka dibuat menggunakan *framework* Javascript yaitu Vue.js dengan Vuetify sebagai penyedia komponen *user interface*. Pada saat pengambilan gambar aplikasi menyediakan pilihan sumber yaitu dari kamera atau file.

Implementasi alur program aplikasi kemudian dilakukan di sisi *backend* sebagai *web service*. Implementasi dilakukan menggunakan bahasa pemrograman Python dengan *framework* Flask. Di sisi *backend* aplikasi menyediakan *endpoint* untuk melakukan pendeteksian jerawat, *input* berupa foto atau gambar dan *output* berupa koordinat lokasi jerawat berformat JSON.



(a) Tampilan pengambilan gambar



(b) Tampilan hasil pendeteksian

Gambar 10. Implementasi Aplikasi

Pengembangan aplikasi menggunakan *web service* ini dapat dimanfaatkan untuk pengembangan aplikasi mobile, dengan mengkonsumsi *endpoint* yang ada. Arsitektur aplikasi dirancang untuk mendukung pengembangan aplikasi kedepannya, pendeteksian masalah kuli yang lain seperti keriput, kemerahan, dan kuli kusam. Hal tersebut dapat dilakukan dengan menambahkan *endpoint* baru pada sisi *backend*.

Aplikasi yang dikembangkan sudah dapat menggungah file foto atau dari kamera pengguna. Pendeteksian jerawat pada aplikasi rata-rata memerlukan waktu 2,77 detik pada setiap gambar.

V. KESIMPULAN

Aplikasi web untuk melakukan pendeteksian jerawat menggunakan algoritma *deep learning* pada TensorFlow sudah dapat mendekeksi jerawat pada foto wajah. Area kulit wajah diperoleh dengan efektif menggunakan pendeteksian wajah dikombinasikan dengan pendeteksian landmark wajah. Model pendeteksian jerawat diperoleh dari hasil *transfer learning* dari model SSD ResNet50 V1 FPN 640x640 yang dilatih dengan dataset jerawat, menghasilkan nilai *sensitivity* sebesar 77.3%, *specificity* sebesar 47.3%, dan *accuracy* sebesar 63.2%. Penggunaan model tersebut pada aplikasi sudah cukup baik untuk mendeteksi jerawat di area kulit wajah. Aplikasi dalam melakukan pendeteksian jerawat memerlukan waktu rata-rata 2,77 detik untuk setiap gambar. Model pendeteksian memiliki potensi untuk dilakukan peningkatan akurasi setelah aplikasi dirilis, aplikasi dapat mengumpulkan lebih banyak foto dari pengguna. Pelabelan gambar baru kemudian bisa dilakukan kembali, ini bisa digunakan untuk mengulang pelatihan model.

DAFTAR PUSTAKA

[1] health24. (2020). What is acne? Health24. https://www.news24.com/health24/Medical/Acne/About-acne/Acne-20120721.

[2] Suva, M., "A Brief Review on Acne Vulgaris: Pathogenesis, Diagnosis, and Treatment" , Research & Reviews: Journal of Pharmacology. Vol. 4, Issue 3,2015.

[3] MediLexicon International. (2017). Acne: Causes, treatment, and tips. Medical News Today. https://www.medicalnewstoday.com/articles/107146.

[4] Strauss, John S., Daniel P. Krowchuk, James J. Leyden, Anne W. Lucky, Alan R. Shalita, Elaine C. Siegfried, Diane M. Thiboutot et al. "Guidelines of care for acne vulgaris management." Journal of the American Academy of Dermatology 56, no. 4 (2007): 651-663

[5] VISIA Skin Analysis, from https://www.canfieldsci.com/imaging-systems/visia-complexion-analysis/

[6] Prasetyo, D., Muhimmah, I., & Kurniawardhani, A. (2018). Aplikasi Pendeteksi Jerawat Di Wajah Dengan Menggunakan Teknik Pengolahan Citra Pada Foto.

[7] 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

[8] Alamdari, N., Tavakolian, K., Alhashim, M., & Fazel-Rezai, R. (2016). Detection and classification of acne lesions in acne patients: A mobile application. IEEE International Conference on Electro Information Technology, 2016-August, 739–743. https://doi.org/10.1109/EIT.2016.7535331

[9] Imran, W. T., & Lawi, A. (2020). Face image detection using haar cascade classifier. 1–5.

[10] Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1867–1874. https://doi.org/10.1109/CVPR.2014.241

[11] King, D. (2014). Real-Time Face Pose Estimation. dlib C++ Library. http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html.

[12] Tensorflow. Tensorflow. (n.d) https://www.tensorflow.org/

[13] What is TensorFlow? How it Works? Introduction & Architecture. Guru99. (n.d.). https://www.guru99.com/what-is-tensorflow.html.

[14] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January, 3296–3305. https://doi.org/10.1109/CVPR.2017.351.

[15] Tensorflow Object Detection API. (n.d.). Retrieved from https://github.com/tensorflow/models/tree/master/research/object_detection.

[16] Aplikasi web. Aplikasi web - Wikipedia bahasa Indonesia, ensiklopedia bebas. (2020). https://id.wikipedia.org/wiki/Aplikasi_web.

[17] What is Web Application Architecture? Components, Models, and Types. Hackr.io. (2021). https://hackr.io/blog/web-application-architecture-definition-models-types-and-more.

[18] TensorFlow 2 Object Detection API tutorial documentation. (2020). https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest.

[19] Morgunov, A. (2021). TensorFlow Object Detection API: Best Practices to Training, Evaluation & Deployment. neptune.ai. https://neptune.ai/blog/tensorflow-object-detection-api-best-practices-to-training-evaluation-deployment.

[20] Adam, Jason. (20 Desember 2019). ResNet-50 API. https://jason-adam.github.io/resnet50/.

[21] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2020). Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2), 318–327. https://doi.org/10.1109/TPAMI.2018.2858826

[22] Muhimmah, I., Abriyani, F., & Kurniawardhani, A. (2019). Automatic wrinkles detection on face image. IOP Conference Series: Materials Science and Engineering, 482(1). https://doi.org/10.1088/1757-899X/482/1/012026